UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MASTER OF SCIENCE IN NEUROTECHNOLOGY

MASTER´S THESIS

DESIGN, IMPLEMENTATION AND ANALYSIS OF A HUMAN-ROBOT TRUST METHODOLOGY USING ELECTROENCEPHALOGRAPHY AND ARTIFICIAL INTELLIGENCE MODELS

MARIA ROLLANO CORROTO

2025

Master of Science in Neurotechnology

MASTER´S THESIS

Title: Design, Implementation and Analysis of a Human-Robot Trust Methodology Using Electroencephalography and Artificial Intelligence Models

Author: María Rollano Corroto

Technical Supervisor: Laura Melgar

Academic Supervisor: Javier Bajo

Department: Artificial Intelligence

EXAMINATION COMMITTEE

President: --------------------------------

Member: ---------------------------------------

Secretary: ---------------------------------

Substitute: -----------------------------------

The above-named members of the Examination Committee agree to grant the qualification of :

Date of Defense: ---------------------------------

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MASTER OF SCIENCE IN NEUROTECHNOLOGY

MASTER´S THESIS

DESIGN, IMPLEMENTATION AND ANALYSIS OF A HUMAN-ROBOT TRUST METHODOLOGY USING ELECTROENCEPHALOGRAPHY AND ARTIFICIAL INTELLIGENCE MODELS

MARIA ROLLANO CORROTO

2025

Resumen

La confianza desempeña un papel clave en la forma en que las personas interactúan con sistemas automatizados, pero medirla de forma objetiva y en tiempo real sigue siendo un reto. Los métodos tradicionales, como cuestionarios post-tarea, no logran captar cambios rápidos y momentáneos que se producen durante la interacción. Esta tesis estudia el uso de señales de electroencefalografía (EEG) combinadas con técnicas de inteligencia artificial (IA) como alternativa para clasificar niveles de confianza en tareas de cooperación humano-robot.

Con este objetivo, el trabajo busca mejorar el estado del arte tomando como punto de partida un estudio previo que empleó datos de EEG para evaluar modelos clásicos de aprendizaje automático (como KNN, SVM y Naive Bayes) y modelos de aprendizaje profundo (como redes neuronales convolucionales y Vision Transformers) en este mismo contexto de interacción. Una aportación destacada de dicho estudio fue el uso de modelos de aprendizaje profundo diseñados para capturar la estructura espacial de las señales EEG, un enfoque aún poco explorado en este ámbito.

Partiendo del mismo conjunto de datos y protocolo experimental, esta tesis reproduce la metodología original y la amplía mediante la evaluación de nuevas configuraciones. En particular, se comparan entradas con una o varias características, segmentación por ventanas (slice-wise) frente a segmentación por ensayo completo (trial-wise), y validación intra-sujeto frente a inter-sujeto. Estas comparaciones permiten validar el enfoque previo y ofrecen una visión más completa sobre cómo decisiones de modelado —como segmentación temporal, variabilidad entre sujetos o selección de características— afectan la robustez y capacidad de generalización de los sistemas de reconocimiento de confianza basados en EEG.

Los resultados mostraron que se obtenía mayor precisión al utilizar una única característica por canal, mientras que añadir más características solía reducir el rendimiento, probablemente por introducir información redundante o ruidosa. La segmentación por ventanas también ofreció mejores resultados que la segmentación por ensayo completo, posiblemente debido a solapamientos temporales que generan correlaciones entre los datos de entrenamiento y prueba. Esto puede facilitar la clasificación, aunque también conducir a una sobreestimación del rendimiento. Por último, la validación intra-sujeto superó a la inter-sujeto, lo que refleja la dificultad de generalizar entre individuos.

La combinación —una característica, segmentación por ventanas y validación intra-sujeto— logró la mayor precisión utilizando XGBoost, con una media del 79,6%. Los modelos clásicos como SVM y Random Forest ofrecieron un rendimiento notable, superando ligeramente a los de aprendizaje profundo, con CNN y ViT alcanzando precisiones del 75,5% y 75,1%, respectivamente.

Estos resultados evidencian el valor del EEG para clasificar la confianza en cooperación humano-robot, aunque también plantean desafíos que deben resolverse antes de aplicar estos enfoques en escenarios reales. Los modelos clásicos ofrecieron un rendimiento sólido. No obstante, los de aprendizaje profundo podrían mostrar su verdadero potencial con conjuntos de datos más amplios y diversos. A futuro, conviene investigar cómo mejorar la generalización entre individuos, por ejemplo, mediante técnicas de transferencia, y diseñar modelos más interpretables que puedan integrarse en sistemas adaptativos y fiables. El código ha sido desarrollado por la autora y está disponible en: https://github.com/mrollano/Thesis_MariaRollano.git.

Palabras clave

Electroencefalografía (EEG), Inteligencia Artificial (IA), Clasificación del nivel de confianza, Interacción Humano–Robot, Aprendizaje Automático (ML), Aprendizaje Profundo (DL), Redes Neuronales Convolucionales (CNN), Vision Transformer (ViT), XGBoost

Summary

Trust plays a fundamental role in how people interact with automated systems, yet measuring it objectively and in real time remains a significant challenge. Traditional assessment methods, such as post-task questionnaires, fail to capture rapid, moment-to-moment changes that occur during interaction. This thesis explores the use of electroencephalography (EEG) signals combined with artificial intelligence (AI) techniques as an alternative approach to recognize trust levels during human–robot cooperation tasks.

To this end, the study aims to improve the state of the art by taking a previously published approach as a foundation. The original work used EEG data and evaluated traditional machine learning models (e.g., KNN, SVM, and Naive Bayes) and deep learning models (e.g., Convolutional Neural Network and Vision Transformer) to classify trust levels during human–robot interaction. A key innovation of the paper lies in the use of deep learning models designed to capture the spatial structure of EEG signals, an aspect still relatively novel in trust recognition. Using the same dataset and experimental protocol, this thesis reproduces the referenced methodology and extends it by evaluating additional configurations. Specifically, it compares single-feature versus multi-feature inputs, slice-wise versus trial-wise segmentation, and within-subject versus between-subject validation. These comparisons validate previous work but also provide a broader perspective on how modeling choices, including temporal segmentation, subject variability, and feature selection, influence the robustness and generalizability of trust recognition systems based on EEG signals.

This setup was tested, and results showed that the highest classification accuracy was consistently achieved when using one characteristic per channel as input. Adding more features typically led to reduced performance, likely due to the introduction of redundant or noisy information. Furthermore, slice-wise segmentation significantly outperformed trial-wise, possibly due to overlapping time windows that introduce correlations between training and test data, making classification easier but potentially overestimating performance. The within-subject evaluation also yielded considerably better results than the between-subject, highlighting the difficulty of generalizing models across individuals. Taken together, the best configuration — one characteristic, slice-wise segmentation and within-subject validation — achieved the highest accuracy using XGBoost classifier, reaching an average of 79.6%. Classical models like Support Vector Machine and Random Forest also performed strongly, slightly outperforming the deep learning models, with CNN and ViT achieving accuracies of 75.5% and 75.1%, respectively.

These findings show that EEG signals can be a useful tool for recognizing trust in human–robot cooperation, but also highlight challenges that must be tackled before applying these approaches in real-world settings. Interestingly, classical models performed particularly well, in most cases surpassing deep learning methods. However, this does not mean that deep learning lacks potential; its full capabilities may only become evident when tested on much larger and more diverse datasets. Looking ahead, future research should explore ways to improve generalization across individuals, for example by applying transfer learning techniques, and make models more interpretable to support their use in adaptive, trustworthy systems. All code used in this thesis was developed by the author, based on and extending previous work. The full repository is available at: https://github.com/mrollano/Thesis_MariaRollano.git

Keywords

Electroencephalography (EEG), Artificial Intelligence (AI), Trust Recognition, Human–Robot Interaction, Machine Learning, Deep Learning, Convolutional Neural Networks (CNN), Vision Transformer (ViT), XGBoost

Index

1 Introduction and Objectives **1**

1.1 Structure of the Document 2

2 State of the Art **3**

2.1 EEG and Trust 3

2.2 EEG, Trust and Artificial Intelligence Models 4

2.3 Open Datasets 6

3 Development **9**

3.1 Data Acquisition and Structure 9

3.2 Signal Preprocessing 11

3.2.1 Common Average Reference 12

3.2.2 Band-Pass Filter 13

3.2.3 Notch Filter 14

3.2.4 Artifact Removal via Independent Component Analysis 15

3.2.5 Epoch Extraction 16

3.2.6 Baseline Removal 17

3.3 Experimental Evaluation Framework 18

3.3.1 Types of Approaches 18
3.3.2 Feature Extraction 20
3.4 Artificial Intelligence Models for EEG Classification 24
3.4.1 K-Nearest Neighbors 24
3.4.2 Support Vector Machine 25
3.4.3 Naive Bayes 26
3.4.4 Random Forest 27
3.4.5 Extreme Gradient Boosting 27
3.4.6 Convolutional Neural Network 28
3.4.7 Vision Transformer 33

4 Results and Discussion **38**
4.1 Performance of Proposed Classification Models 39
4.2 Comparison With Reference Paper Results 44
4.3 Discussion and Interpretation of Results 47

5 Conclusions and Future Works **52**

6 Ethical, Economic, Societal and Environmental Aspects **54**
6.1 Introduction 54
6.2 Description of Relevant Impacts Related to the Project 54
6.2.1 Ethical Considerations 54
6.2.2 Economic Implications 55
6.2.3 Social Impact 55
6.2.4 Environmental Impact 55
6.2.5 Relevant Stakeholders 56

6.3 Detailed Analysis of One of the Principal Impacts: Ethical Considerations . . . 56

6.4 Conclusions . . . 58

7 Financial Budget **59**

Bibliography **61**

Annexes **67**

A Hyperparameter Search . . . 67

List ot Figures

3.1 Overview of the preprocessing workflow. 12

3.2 PSD for all EEG channels, computed using EEGLAB. The spectral power is primarily concentrated below 60 Hz, supporting the selected band-pass filtering range. 13

3.3 Frequency and phase response of the FIR band-pass filter (0.5–60 Hz) used in the preprocessing pipeline, generated with EEGLAB. 14

3.4 ICLabel classification of independent components. Each scalp topography corresponds to a component labeled with its most likely class and associated confidence score, generated with EEGLAB. 16

3.5 Overview of the six experimental configurations evaluated in this study, combining different EEG feature representations (one vs. multiple characteristics) with modeling paradigms (within-subject vs. between-subject) and data partitioning strategies (slice-wise vs. trial-wise). 18

3.6 Illustration of a convolutional layer with multi-channel input and multiple filters. Each filter generates a corresponding feature map by performing convolution operations over local patches of the input. Reproduced from [1]. . . 29

3.7 Schematic representation of a CNN workflow. The input is processed through convolutional and pooling layers to extract features, followed by flattening and fully connected layers for classification. The final output is a probability distribution over the target classes. Extracted from [2]. 31

3.8 (a) Mapping of EEG channels into a 9×9 spatial grid based on electrode positions. (b) CNN input representation and feature flow. Extracted from [3]. . 32

3.9 ViT architecture overview. The input image is divided into fixed-size patches, which are flattened and linearly projected. A positional encoding is added to each patch embedding, and a learnable classification token is prepended. The resulting sequence is processed by a stack of Transformer encoder layers. The output corresponding to the classification token is used for final prediction. Adapted from [4]. 34

3.10 Diagram of the MSA mechanism. Queries (Q), keys (K), and values (V) are derived through separate linear projections. Multiple attention heads compute scaled dot-product attention in parallel. The outputs are then concatenated and passed through a final linear layer. Reproduced from [5]. 35

4.1 Distribution of perceived trust scores across different levels of robot performance and task complexity, extracted from [3]. 38

4.2 Distribution of the top 20 most important features per subject using the XGBoost model. Results correspond to the within-subject, multiple characteristics, slice-wise approach and are categorized by feature type. 48

4.3 Distribution of the top 20 most important features per subject using the Random Forest model. Results correspond to the within-subject, multiple characteristics, slice-wise approach and are categorized by feature type. 49

4.4 Topographic distribution of average feature importance across EEG channels using only Band Differential Entropy (BDE) features extracted from the XGBoost model. Results correspond to the within-subject, slice-wise, multiple-characteristics configuration. 50

4.5 Comparison of model performance under the within-subject, slice-wise, single-feature configuration. 51

List of Tables

2.1 Summary of studies using machine learning for EEG-based trust prediction. The accuracy of the best model reported in each study is indicated in parentheses. EEG features: EDA = Electrodermal Activity, BVP = Blood Volume Pulse, HR = Heart Rate, SKT = Skin Temperature, BR = Blinking Rate, BD = Blinking Duration, HRV = Heart Rate Variability, ST = Skin Temperature, GSR = Galvanic Skin Response, EEG = Electroencephalogram, fNIRS = Functional Near-Infrared Spectroscopy, ECG = Electrocardiogram, RSP = Respiration. 5

3.1 Sample of post-trial trust ratings for participant Sub01. 10

3.2 Hyperparameter search configuration for each AI model evaluated in this study. 37

4.1 KNN model performance across different evaluation paradigms and feature configurations. 40

4.2 SVM model performance across different evaluation paradigms and feature configurations. 41

4.3 NB model performance across different evaluation paradigms and feature configurations. 41

4.4 Random Forest model performance across different evaluation paradigms and feature configurations. 42

4.5 XGBoost model performance across different evaluation paradigms and feature configurations. 42

4.6 CNN model performance across different evaluation paradigms and feature configurations. 43

4.7 Transformer model performance across different evaluation paradigms and feature configurations. 43

4.8 Comparison of accuracy between this work and the reference study, under slice-wise and trial-wise validation for the within-subject setting using a single characteristic. 44

4.9 Comparison of F1-score between this work and the reference study, under slice-wise and trial-wise validation for the within-subject setting using a single characteristic. 45

7.1 Estimated budget for the project 60

7.2 Best hyperparameters per subject using KNN on within-subject, one characteristic, slice-wise approach. 67

7.3 Best hyperparameters per subject using KNN on within-subject, one characteristic, trial-wise approach. 68

7.4 Best hyperparameters per subject using KNN on within-subject, multiple characteristics, slice-wise approach. 68

7.5 Best hyperparameters per subject using KNN on within-subject, multiple characteristics, trial-wise approach. 69

7.6 Best hyperparameters per test subject using KNN on between-subject, one characteristic. 69

7.7 Best hyperparameters per test subject using KNN on between-subject, multiple characteristics. 70

7.8 Best hyperparameters per subject using SVM on within-subject, one characteristic, slice-wise approach. 70

7.9 Best hyperparameters per subject using SVM on within-subject, one characteristic, trial-wise approach. 71

7.10 Best hyperparameters per subject using SVM on within-subject, multiple characteristics, slice-wise approach. 71

7.11 Best hyperparameters per subject using SVM on within-subject, multiple characteristics, trial-wise approach. 72

7.12 Best hyperparameters per subject using NB on within-subject, one characteristic, slice-wise approach. 72

7.13 Best hyperparameters per subject using NB on within-subject, one characteristic, trial-wise approach. 73

7.14 Best hyperparameters per subject using NB on within-subject, multiple characteristics, slice-wise approach. 73

7.15 Best hyperparameters per subject using NB on within-subject, multiple characteristics, trial-wise approach. 74

7.16 Best hyperparameters per test subject using NB on between-subject, one characteristics. 74

7.17 Best hyperparameters per test subject using NB on between-subject, multiple characteristics. 75

7.18 Best hyperparameters per subject using RF on within-subject, one characteristic, slice-wise approach. 75

7.19 Best hyperparameters per subject using RF on within-subject, one characteristic, trial-wise approach. 76

7.20 Best hyperparameters per subject using RF on within-subject, multiple characteristics, slice-wise approach. 76

7.21 Best hyperparameters per subject using RF on within-subject, multiple characteristics, trial-wise approach. 77

7.22 Best hyperparameters per test subject using RF on between-subject, one characteristic. 77

7.23 Best hyperparameters per test subject using RF on between-subject, multiple characteristics. 78

7.24 Best hyperparameters per subject using XGBoost on within-subject, one characteristic, slice-wise approach. 78

7.25 Best hyperparameters per subject using XGBoost on within-subject, one characteristic, trial-wise approach. 79

7.26 Best hyperparameters per subject using XGBoost on within-subject, multiple characteristics, slice-wise approach. 79

7.27 Best hyperparameters per subject using XGBoost on within-subject, multiple characteristics, trial-wise approach. 80

7.28 Best hyperparameters per test subject using XGBoost on between-subject, one characteristic. 80

7.29 Best hyperparameters per test subject using XGBoost on between-subject, multiple characteristics. 81

Glossary

AC – Alternating Current

AI – Artificial Intelligence

BD – Blinking Duration

BDE – Band Differential Entropy

BR – Breathing Rate

BVP – Blood Volume Pulse

CAR – Common Average Reference

CNN – Convolutional Neural Network

CV – Cross-Validation

DC – Direct Current

DE – Differential Entropy

DL – Deep Learning

ECG – Electrocardiogram

EDA – Electrodermal Activity

EEG – Electroencephalography

ERN – Error-Related Negativity

F1 – F1 Score

FIR – Finite Impulse Response

fNIRS – Functional Near-Infrared Spectroscopy

GELU – Gaussian Error Linear Unit

GSR – Galvanic Skin Response

HAR – High-Ability Robot

HR – Heart Rate

HRV – Heart Rate Variability

ICA – Independent Component Analysis

KNN – K-Nearest Neighbors

LAR – Low-Ability Robot

LOSO – Leave-One-Subject-Out

MAR – Medium-Ability Robot

MDMT – Multidimensional Measure of Trust

MHA – Multi-Head Attention

ML – Machine Learning

MLP – Multilayer Perceptron

MSA – Multi-Head Self-Attention

NB – Naive Bayes

Pe – Error Positivity

PSD – Power Spectral Density

RBF – Radial Basis Function

ReLU – Rectified Linear Unit

RF – Random Forest

RSP – Respiration

SA – Self-Attention

SAE – Simulated Conditionally Automated Environment

SE – Sample Entropy

SELU – Scaled Exponential Linear Unit

SKT – Skin Temperature

ST – Skin Temperature

SVM – Support Vector Machine

ViT – Vision Transformer

XGBoost – Extreme Gradient Boosting

Introduction and Objectives

Over the past few years, the integration of intelligent systems into daily life has grown rapidly, particularly through the use of collaborative robots [6]. As these technologies become increasingly present in healthcare, homes, and professional settings, understanding how people perceive and trust them has become essential.

However, trust is not something that can be directly seen or easily measured. It depends on many factors, such as the complexity of the task or even past human experiences. Traditional methods often rely on questionnaires to measure it; however, these are limited when it comes to capturing real-time changes or fast mental processes that occur during interaction [7, p. 4].

This thesis explored an alternative approach by using electroencephalography (EEG) signals to classify human trust levels in human–robot collaboration scenarios. Unlike traditional methods, EEG offers a dynamic, real-time technique for assessing cognitive states [8], making it particularly valuable for identifying patterns related to trust. By monitoring brain activity, EEG is able to detect the subtle, real-time fluctuations that questionnaires may miss, providing a more immediate and accurate representation of trust during interaction.

To explore this idea, this work relied on the EEGTrust dataset, introduced in a study [3] where participants collaborated with robots of varying ability levels across tasks of different difficulty. These conditions were designed to influence the participants' perception of the robot, and trust levels were subsequently measured using post-trial questionnaires and classified using EEG data. The EEGTrust dataset is publicly available at [9].

The main objective of this project was to carefully reproduce and understand the methodology proposed in the original study, in order to validate its results and check whether similar outcomes could be obtained. Beyond replication, the project also aimed to explore different approaches and test a wider range of machine learning and deep learning models for trust classification.

The results were then analyzed and compared to understand how different models and strategies performed. The aim was not only to evaluate trust classification accuracy, but also to reflect on what these findings reveal about how trust might be detected from brain activity in practical human–robot interaction scenarios. In doing so, the project hoped to offer insights that may support future research and contribute to ongoing efforts in the scientific community to develop more effective and human-aware robotic systems.

1.1 Structure of the Document

To capture the entire process, from reviewing the current state of the field to developing new models and analyzing the resulting conclusions, the document is structured to reflect this progression.

To begin with, Section 2 presents the state of the art, starting with a review of previous research on trust in human–robot interaction and the use of EEG signals to model it. It also examines recent developments in artificial intelligence applied to EEG data and introduces relevant public datasets that support this line of work.

With this background in place, Section 3 focuses on the development phase. It begins with a detailed overview of the EEGTrust dataset, including its structure and experimental setup. The section then outlines the preprocessing steps applied to the EEG signals, describes the machine learning and deep learning models used, and explains the evaluation framework adopted throughout the project.

As a result of this process, Section 4 presents the results obtained from the different models under the specific evaluation conditions.

Finally, Section 5 offers an interpretation of the results, discusses the main limitations of the study, and outlines possible directions for future work.

State of the Art

The starting point for this project was a thorough review of the existing literature and available resources in the field. This initial analysis aimed to identify the main strategies used to model and predict trust from EEG data, explore how artificial intelligence techniques had been applied in this context, and examine the availability of open-access datasets that supported reproducible research in this area.

To organize the findings, this chapter is divided into three parts: (1) EEG and trust, (2) EEG, trust, and AI models, and (3) available open-access datasets.

2.1 EEG and Trust

EEG has become a widely used tool in neuroscience due to its ability to capture brain activity with high temporal resolution in a non-invasive way [10]. In the context of human-machine interaction, EEG is gaining relevance as a way to objectively measure internal mental states, including cognitive load [11], engagement [12], and, more recently, trust [13].

Trust, however, is a complex and dynamic process. It evolves based on the situation, individual differences, and the nature of the interaction itself [7]. Several studies have begun to explore how it can be inferred from EEG signals in human–machine interaction scenarios. Although a wide range of EEG features has been explored, two types of neurophysiological markers are commonly highlighted in the literature: event-related potentials and oscillatory patterns in specific frequency bands.

Regarding event-related potentials, two components have been widely studied in relation to trust: Error-Related Negativity (ERN) and Error Positivity (Pe). The ERN is typically observed as a sharp negative deflection in the EEG signal following erroneous or unexpected feedback and is believed to reflect a rapid internal monitoring process that detects mismatches between expected and actual outcomes. The Pe, on the other hand, is a later positive deflection associated with the conscious recognition and evaluation of such errors. In the context of automation and trust, de Visser et al. [14] showed that EEG components, specifically ERN and Pe, are linked to trust-related processes. Importantly, these components were triggered not only by errors made by the participants themselves but also when the errors were attributed to an automated system. This finding provides strong evidence that violations of user expectations by automation can evoke neural responses, signaling a loss of trust.

As for oscillatory activity, studies have identified consistent changes in specific frequency bands, most notably, a reduction in parietal alpha power and an increase in frontal theta activity. These patterns are associated with altered cognitive states: parietal alpha desynchronization often reflects heightened alertness or reduced relaxation, while increased frontal theta is commonly linked to enhanced cognitive control or mental effort. Zander et al. [15] interpreted these shifts as indicators of reduced user confidence or elevated cognitive monitoring during interaction with a system, conditions that could indicate reduced trust.

While these studies support the idea that trust can be inferred from brain activity patterns, the field remains far from establishing a standardized methodology. Studies often rely on different EEG features, extract them using varied preprocessing strategies, and interpret them according to task-specific or user-specific dynamics. This lack of consensus, combined with high inter-individual variability and inconsistent experimental designs, makes it difficult to draw generalizable conclusions about how trust can be detected from EEG signals.

2.2 EEG, Trust and Artificial Intelligence Models

Given the difficulties in interpreting EEG data for trust detection, recent studies have started to explore the use of artificial intelligence (AI) techniques to support this task. These methods aim to handle the complexity and variability of EEG signals, offering new ways to extract meaningful patterns linked to trust. Still, the application of AI in this area is at an early stage.

So far, only a limited number of studies have explored this combination in practice. Most have used traditional machine learning algorithms such as Support Vector Machines (SVM), Random Forest (RF), or k-Nearest Neighbors (KNN). Table 2.1 provides an overview of the main studies applying machine learning models for EEG-based trust prediction, including the interaction context, type of classification, models applied, and data features used for training the models.

Author(s) and Year	**Interaction**	**Type of Classification**	**Model(s)**	**Data Features**
Ajenaghughure et al. (2019) [16]	Game ("Who Wants to Be a Millionaire")	Binary Classification	Voting Classifier (Linear Regression, LDA, SVM, QDA, KNN, RF, Gradient Boosting) (Accuracy: 77.8%)	EEG
Loizaga et al. (2023) [17]	Prisoner's Dilemma Game with Robot	Binary Classification	KNN, SVM, RF (Accuracy: Not reported)	EEG, GSR, Pupil Dilation, Breathing rate
Li et al. (2024) [18]	High-Speed Train Driving Simulation	Multi-Class Classification	RF, BPNN (Back-propagation Neural Network), Bayesian Network (Accuracy: 93.14%)	EEG, fNIRS, ECG, EDA, RSP
Chauhan, Jang, and Jeong (2024) [19]	Human-Robot Collaboration in Construction	Regression	KNN, Gradient Boost, XGBoost, RF (Accuracy: Not reported)	EEG, EDA, ST, HRV

Table 2.1: Summary of studies using machine learning for EEG-based trust prediction. The accuracy of the best model reported in each study is indicated in parentheses. EEG features: EDA = Electrodermal Activity, BVP = Blood Volume Pulse, HR = Heart Rate, SKT = Skin Temperature, BR = Blinking Rate, BD = Blinking Duration, HRV = Heart Rate Variability, ST = Skin Temperature, GSR = Galvanic Skin Response, EEG = Electroencephalogram, fNIRS = Functional Near-Infrared Spectroscopy, ECG = Electrocardiogram, RSP = Respiration.

In contrast to these traditional methods, deep learning has emerged as a promising extension of them. However, its application in EEG-based trust prediction remains limited. To date, only one study has explored this direction [3]. The authors employed a Convolutional Neural Network (CNN) and a Vision Transformer (ViT), along with other machine learning models like KNN, SVM, and Naive Bayes (NB), all trained on band differential entropy (BDE) features extracted from four frequency bands: theta, alpha, beta, and gamma. An accuracy of 74.19% was achieved for the CNN and 74.99% for the ViT.

While deep learning methods have shown promise, another approach explores combining EEG features with contextual data through a knowledge-driven framework. Zhu et al. [20] proposed a knowledge-driven framework in a multi-robot scenario, which integrates symbolic reasoning (context and rules) with EEG-based signals to estimate trust. Although the study demonstrates qualitative improvements, such as better adaptability to changing robot behavior, it does not provide detailed performance metrics. To date, it remains the only example of this kind of approach, and no standard framework currently exists for combining symbolic and sub-symbolic methods in EEG-based trust modeling.

These studies highlight the potential of AI in EEG-based trust prediction. However, they also reveal key limitations that must be addressed when applying these methods in practice.

Among the most common limitations is the use of binary classification ("high" vs. "low" trust). While this approach simplifies the modeling process, it does not account for the complexity of trust, which can fluctuate over time and is influenced by different factors. Some studies, such as the one presented earlier by Ajenaghughure et al. (2019) in Table 2.1, have used multi-class classification to better capture the varying levels of trust (e.g., low, medium, high). However, binary classification remains prevalent, limiting the models' ability to reflect the dynamic nature of trust in real-world interactions.

Another limitation is the reliance on trust questionnaires as the ground truth. These surveys are inherently subjective and depend on the participant's personal perception of trust, which can vary between individuals. This subjectivity, along with inconsistencies in how trust is rated, can affect the accuracy and generalizability of the models.

Furthermore, the lack of consistency in the methodologies used across studies, particularly in EEG signal processing and analysis, further complicates the situation. Variability in experimental tasks, participant groups, and signal processing techniques makes it difficult to compare results or draw reliable conclusions. In addition, many studies also lack standardized validation protocols, making it unclear how the models would perform with new users or in different contexts.

This section has explored the growing role of AI in trust prediction using EEG, highlighting both its potential and the challenges that come with it. Despite the progress made, significant limitations remain, which need to be addressed in future research to improve the accuracy and applicability of these models. One crucial aspect is the availability of high-quality, open datasets that can be used to advance this field.

In this regard, the next section examines the publicly available datasets relevant to trust prediction through EEG, focusing on those that are specifically designed for human-machine interaction.

2.3 Open Datasets

To develop AI models capable of predicting trust, it is essential to identify publicly available datasets that align with the objectives of this project.

A review of the literature and open data repositories led to the identification of just two datasets providing trust-related data in human–machine interaction settings that used EEG: the EEG-Vision Dataset and the Robot Interaction Dataset.

EEG-Vision Dataset

The EEG-Vision Dataset comes from the study discussed earlier in Section 2.2 by Xu et al. (2024) [3]. The dataset was collected during a cooperative cooking game, where participants worked together with a robot chef to prepare onion soup. The experimental setup involved different types of robots: High Ability Robot (HAR), Medium Ability Robot (MAR), and Low Ability Robot (LAR), and varying task difficulties, creating distinct trust environments to study how trust evolves in these scenarios.

A total of 16 participants took part in the experiment, with each participant completing 30 cooking rounds. After each round, participants were asked to rate how much they trusted the robot during that interaction using the Multidimensional Measure of Trust (MDMT) scale.

This dataset exclusively uses EEG data, which was recorded during the task and trust rating phases. The authors provide the implementation code for the dataset, which is publicly available through their GitHub repository [9].

Robot Interaction Dataset

The Robot Interaction Dataset was introduced by Campagna and Rehm (2024) [21] as part of their study "Trust Assessment with EEG Signals in Social Human–Robot Interaction". The goal of this dataset is to analyze how human trust evolves over time during social interactions with a robot, and it supports both classification and regression tasks to model the dynamics of trust in these interactions.

The experiment involved 21 participants who wore 14-channel Emotiv EEG headsets while playing a structured guessing game with a robot. Throughout the interaction, the robot occasionally provided incorrect suggestions in order to simulate trust violations. To capture the dynamics of trust, the session was divided into five distinct phases: Trust Building, where participants became familiar with the robot; Situational Awareness, where the robot commented on the environment to demonstrate awareness; Transparency, where it explained its decisions; Trust Violation, during which it acted in an untrustworthy way; and Trust Repair, where it attempted to restore trust.

EEG signals were continuously recorded throughout the session, resulting in approximately 3.6 million EEG samples (around 173,000 per participant). The features in this dataset are temporal in nature, reflecting the dynamic changes in trust during the interaction. In addition, after each phase, participants completed the MDMT questionnaire, providing subjective trust ratings aligned with each phase.

All EEG recordings, metadata, and trust labels are publicly available in the following repository [22].

Both datasets, EEG-Vision and Robot Interaction, provide valuable trust-related data in human–machine interaction settings with EEG. While both datasets are essential for advancing trust prediction models, the EEG-Vision Dataset has been selected as the foundation for this project. Several key factors make it particularly suitable: the dataset's well-documented collection process, its balance across trust levels, and its potential for developing robust and unbiased classification models.

In addition, one of its most distinctive strengths lies in its use as the foundation for training a Vision Transformer (ViT) model, as detailed in the paper by Xu et al. (2024). This innovative approach integrates the spatial layout of EEG electrodes into the model's input, allowing it to learn both temporal patterns and spatial dependencies between channels, offering a richer and more informative representation of brain activity. Therefore, the EEG-Vision Dataset serves as the starting point for this project, providing both a solid foundation and an innovative framework for developing advanced trust prediction models.

Development

As mentioned in the previous section, the EEG-Vision dataset served as the foundation for this project. To briefly recall, this dataset was collected during a human-robot cooperation experiment involving 16 participants in a simulated cooking game. Participants completed 30 trials, each followed by a self-reported trust score. The experiment was designed to elicit different levels of trust by varying robot ability and task complexity. Specifically, the robot's ability was manipulated through pre-trained robots with varying levels: high-ability (HAR), medium-ability (MAR), and low-ability (LAR), reflecting differences in task execution capability, human intent understanding, and cooperation. Task complexity was also varied through five different layouts, with layouts 1 to 3 representing simpler tasks and layouts 4 and 5 involving more complex scenarios. These variations in robot ability and task difficulty effectively elicited different levels of trust from the participants.

This section provides a detailed overview of the full data processing pipeline used in this study. It begins by describing the structure and contents of the EEG-Vision dataset. It then outlines the preprocessing steps applied to the EEG signals to prepare them for further analysis. Finally, the section introduces the machine learning models used for trust classification and explains the different experimental approaches evaluated.

3.1 Data Acquisition and Structure

Before beginning any signal processing or model development, the dataset [9] was carefully reviewed to understand its structure and ensure that each data modality was handled appropriately. The dataset is organized into five main folders, each corresponding to a specific component: participant ratings, trajectory data, EEG recordings, face video, and pre-experiment questionnaires. For the purpose of this study, only the first three were used; face video was excluded due to its irrelevance to the analysis, and the pre-experiment questionnaires were not available.

Participant Ratings

After each experimental trial, participants completed a short questionnaire to assess their perceived level of trust in the robot during the interaction. This questionnaire, adapted from the trust scale by Xu and Dudek [23], included three items rated on a 5-point Likert scale. The items focused on three key aspects of the interaction: the robot's ability to complete the task (Ability), the quality of collaboration (Teamwork), and the overall sense of trust it inspired (Trustworthiness).

Each participant's responses were saved in an individual .csv file, containing trial-by-trial entries with metadata such as trial number, task layout (1-3 representing simpler tasks, while 4-5 more complex), robot type (1 = HAR, 2 = MAR, 3 = LAR) and the corresponding ratings. A composite trust score was calculated as the average of the three responses: ability, teamwork, and trustworthiness. Table 3.1 presents an example of this data for participant Sub01.

Participant ID	**Trial**	**Layout**	**Robot**	**Ability**	**Teamwork**	**Trustworthy**	**Average**
Sub01	1	1	1	6	4	6	5.33
Sub01	2	1	2	3	2	2	2.33
Sub01	3	1	3	1	1	1	1.00
Sub01	4	2	3	3	3	3	3.00
Sub01	5	2	2	2	2	2	2.00
Sub01	6	2	1	6	4	6	5.33
Sub01	7	3	2	3	2	2	2.33
Sub01	8	3	3	4	2	3	3.00
Sub01	9	3	1	6	4	6	5.33

Table 3.1: Sample of post-trial trust ratings for participant Sub01.

While this sample shows a clean subset of the data, the complete set of participant ratings contains occasional inconsistencies and missing trials. Although the experimental protocol aimed for 30 trials per participant (across 5 layouts, 3 robot types, and 2 repetitions for each combination), not all participants reached this number. For instance, Sub01 completed only 9 trials, while others, such as Sub02 and Sub03, completed 15. These deviations are likely due to technical issues or interruptions during recording. Some entries also included out-of-range scores, for example, values above 5 in the trust questionnaires, suggesting potential input errors or inconsistencies in the response process.

Trajectory Data

In addition to the subjective ratings, the dataset includes detailed trajectory data that captures the movements of both the human and robot agents during each trial. Stored in JSON format, these files contain frame-by-frame information about each agent's position, orientation, and any objects they were holding at that time.

The robot's behavior was controlled by a policy learned through reinforcement learning, modeled as a Markov Decision Process. This allowed the agent to develop cooperative strategies by maximizing rewards over time. The dataset also includes metadata reflecting the robot's internal decision-making, such as the action selected at each time step, the associated reward, and task-related parameters, for example, the number of ingredients required, cooking duration, or kitchen layout.

EEG Recordings

The third and most crucial component for this study is the EEG data, which were recorded continuously while participants performed the cooperative task. These signals form the basis for the trust recognition models that were developed later in the work. They were captured using a 64-channel G.TEC cap, with electrodes placed according to the international 10–20 system. Data were sampled at 250 Hz, providing high temporal resolution.

The recordings are available in two formats: participants Sub01 to Sub03 used the HDF5 format, while all others used GDF. Each EEG file is accompanied by a .txt annotation file marking the start of each trial. All trials lasted exactly 60 seconds, following the standardized experimental protocol.

Prior to model development, the raw EEG data undergo preprocessing to remove artifacts and enhance signal quality; this is detailed in the following section.

3.2 Signal Preprocessing

EEG signals inherently consist of both neural activity and various sources of noise, such as muscle artifacts, eye movements, or electrical interference. A fundamental challenge in EEG analysis lies in the fact that signal and noise are often interwoven, making it difficult to fully isolate one from the other. Consequently, each preprocessing decision inevitably reflects a compromise between preserving meaningful neural information and eliminating contaminating artifacts.

Preprocessing encompasses all procedures applied to the data after acquisition and before statistical analysis. These procedures can be categorized into three main types: organizational, selective, and transformative. Organizational steps include operations that structure the data without altering its content, such as segmenting continuous recordings into epochs. Selective preprocessing involves identifying and discarding data that are considered unreliable or artifact-contaminated, such as rejecting noisy epochs or excluding malfunctioning electrodes. Transformative steps, in contrast, actively modify the data to improve its signal-to-noise ratio, through methods such as filtering or re-referencing [24].

Taking these principles into account, and to ensure methodological consistency and reproducibility, the pipeline in this project followed the same sequence as the original study, as illustrated in Figure 3.1. Moreover, this approach is widely adopted in EEG research. All processing steps were carried out using MATLAB in combination with the EEGLAB toolbox [25].

The following subsections describe each stage of the pipeline in detail, from re-referencing and filtering to artifact rejection, epoch segmentation, and baseline correction.

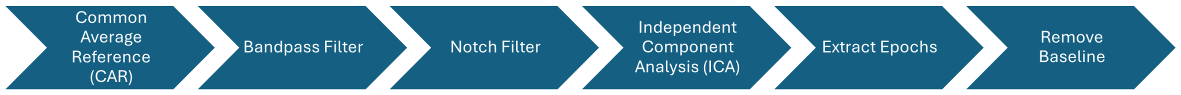
Common Average Reference (CAR)
Bandpass Filter
Notch Filter
Independent Component Analysis (ICA)
Extract Epochs
Remove Baseline

Figure 3.1: Overview of the preprocessing workflow.

3.2.1 Common Average Reference

As a first step in the preprocessing pipeline, the Common Average Reference (CAR) technique was applied to the EEG recordings. Re-referencing is a crucial stage in EEG data preparation, as the choice of reference directly affects the spatial distribution and interpretation of neural signals.

EEG signals can be recorded using various referencing schemes, typically categorized as unipolar, bipolar, or average reference. In a unipolar setup, each electrode's voltage is measured relative to a single reference electrode, often positioned at an electrically neutral site such as the mastoid or earlobe. Although this method is straightforward, it can be highly sensitive to noise introduced at the reference location.

In contrast, bipolar referencing computes the voltage difference between pairs of adjacent electrodes. This configuration enhances the detection of local signal features and can suppress certain types of noise, but it may obscure spatially distributed neural activity, limiting its utility for some analyses.

The average reference approach, as implemented in CAR, calculates the mean voltage across all electrodes at each time point and subtracts this value from each individual channel. This method relies on the assumption that the summed electrical potentials across the scalp approximate zero. By distributing reference influence equally among all electrodes, CAR reduces the impact of noise from any single channel and improves the overall signal-to-noise ratio [26].

Due to these theoretical advantages, average referencing is commonly recommended in high-density EEG configurations, where the underlying assumption of signal symmetry across the scalp is more likely to hold [27]. In this dataset, EEG was recorded using a 64-channel cap with electrodes evenly distributed across the scalp, supporting the validity of the average reference assumption and reinforcing the suitability of CAR as a preprocessing step. Applying CAR at the outset of the pipeline established a consistent and reliable baseline for all subsequent analyses.

3.2.2 Band-Pass Filter

As a second step, a band-pass filter between 0.5 and 60 Hz was applied to the EEG recordings. This filtering stage is essential for attenuating low-frequency drifts and high-frequency noise, thereby enhancing the signal-to-noise ratio.

Given that the primary goal of this study was to investigate neural correlates of trust, typically expressed within specific EEG frequency bands, a broad frequency range was preserved during filtering to retain relevant spectral components. The selected range (0.5 to 60 Hz) encompassed the delta (0.5–4 Hz), theta (4–7 Hz), alpha (8–12 Hz), beta (13–30 Hz), and low gamma (30–60 Hz) bands. Particular attention was paid to the theta and alpha bands, as they have been frequently associated with cognitive and affective processes involved in trust and social decision-making [28]. Theta activity is linked to memory encoding and emotional regulation [29], whereas alpha rhythms are related to attentional control and internally directed cognitive processing [30]. In addition, the inclusion of beta and low gamma frequencies allowed for the examination of their potential contributions to higher-order cognitive functions, such as attention modulation, action monitoring, and sensory integration, which might also influence the formation or modulation of trust [30, 31].

To empirically support this choice, the power spectral density (PSD) across all EEG channels was computed. As shown in Figure 3.2, the majority of spectral power is concentrated below 60 Hz, confirming that the selected band-pass range captures the most informative frequency components for the study.

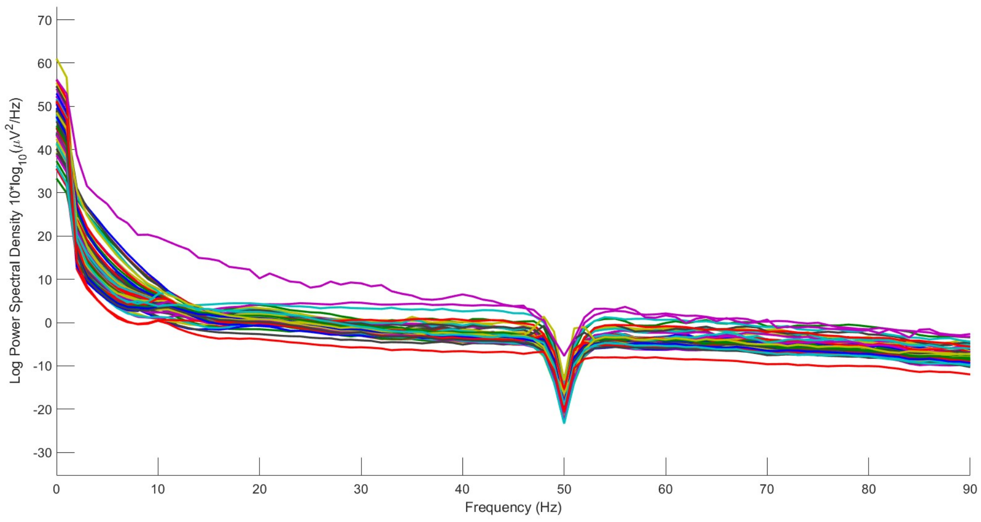

Figure 3.2: PSD for all EEG channels, computed using EEGLAB. The spectral power is primarily concentrated below 60 Hz, supporting the selected band-pass filtering range.

To preserve phase integrity within the selected frequency range, a linear-phase Finite Impulse Response (FIR) filter was applied. Linear-phase filters maintain a constant group delay across frequencies, which preserves the temporal structure of the EEG signal after filtering. This property is essential for analyses involving event-related potentials or other temporally sensitive neural markers [32]. The filter was implemented using the windowed sinc method available in EEGLAB, which employs a Hamming window to design the filter kernel. This configuration offers a favorable balance between frequency selectivity and minimal signal distortion, aligning well with the analytical objectives of the study [33]. The frequency and phase response of the implemented FIR filter is shown in Figure 3.3, confirming the preservation of passband characteristics up to 60 Hz with linear-phase behavior across the spectrum.

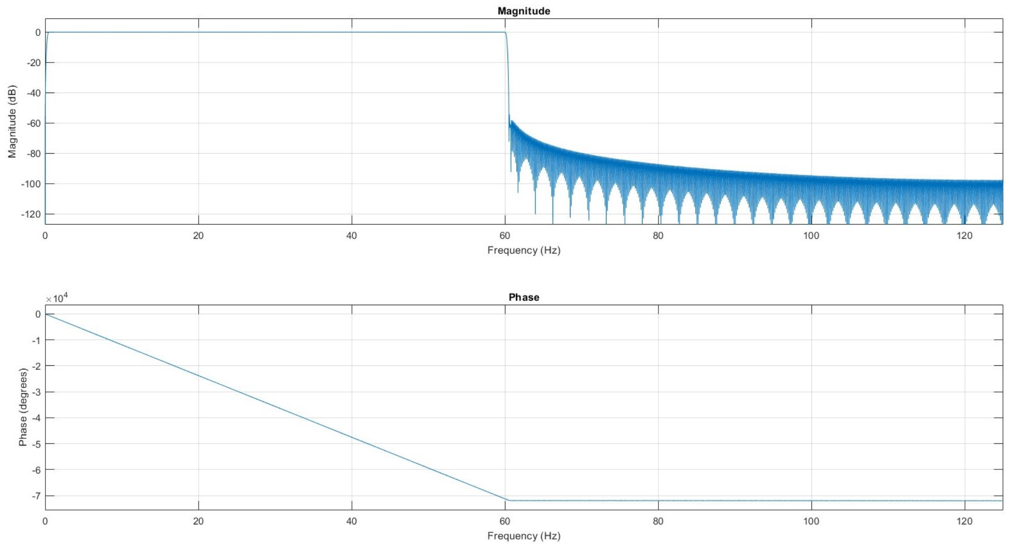

Figure 3.3: Frequency and phase response of the FIR band-pass filter (0.5–60 Hz) used in the preprocessing pipeline, generated with EEGLAB.

3.2.3 Notch Filter

In EEG recordings, power line interference is a common source of contamination, typically appearing as a sinusoidal artifact centered at 50 Hz in Europe and 60 Hz in North America. This interference originates from the alternating current (AC) electrical supply and can severely affect the quality of EEG data, particularly within frequency bands relevant to cognitive and affective processing [34]. A standard approach to mitigate this type of noise is to apply a notch filter designed to suppress the specific interference frequency while preserving neighboring spectral components. However, despite their utility, notch filters can introduce phase distortions or attenuate adjacent frequencies if not carefully implemented [35].

Visual inspection of the PSD, shown previously in Figure 3.2, reveals the absence of a prominent peak at 50 Hz, indicating that power line interference had already been attenuated prior to the analysis. The notch observed at that frequency confirms that a 50 Hz filter was applied during the original data preparation. As a result, no additional notch filtering was introduced during the preprocessing pipeline, in order to avoid redundant processing steps and minimize the risk of unnecessary signal distortion.

3.2.4 Artifact Removal via Independent Component Analysis

To separate neural signals from non-neuronal sources, Independent Component Analysis (ICA) was used as a data-driven method to isolate statistically independent components in the EEG recordings [36].

ICA operates by estimating a mixing matrix M, which characterizes how latent sources combine to produce the observed EEG signals. The sources, or components (C), are computed by inverting this matrix as follows:

$$C = M^{-1}X,$$

where X is the observed EEG data. Once the independent components are estimated, those identified as artifacts are nullified (e.g., $C_n = 0$), and the cleaned EEG signal is reconstructed using the inverse transformation:

$$X' = MC.$$

This procedure enables the targeted removal of artifacts while preserving the structure of underlying neural activity.

Following the application of CAR, the EEG signals, originally recorded using a 64-channel montage, exhibited a linear dependency among channels. This dependency reduced the effective rank of the data by one, limiting the number of extractable independent components to a maximum of 63. This rank reduction is a direct mathematical consequence of the referencing scheme, which enforces that the sum of all channel signals is zero at every time point. Attempting to extract more components than the data rank permits would lead to unstable or non-informative solutions [30].

Following ICA decomposition, artifact-related components were identified and removed using the ICLabel plugin. ICLabel is a supervised machine learning classifier trained on a large dataset of expert-labeled EEG components. For each independent component, the algorithm estimates the probability of belonging to one of seven classes: brain, muscle, eye, heart, line noise, channel noise, or other. In accordance with the ICLabel documentation and commonly accepted practices, components were removed if their class probability exceeded 0.90 for eye artifacts, 0.90 for muscle artifacts, or 0.80 for channel noise. These thresholds are recommended by the developers of ICLabel to ensure a high degree of confidence in component classification, while maintaining a balance between artifact rejection and neural signal preservation [37]. This strategy minimizes the inclusion of artifactual sources in the data without overly penalizing ambiguous components that may still contain valuable neurophysiological information. An example of the ICLabel interface, showing how components are visualized and categorized, is presented in Figure 3.4.

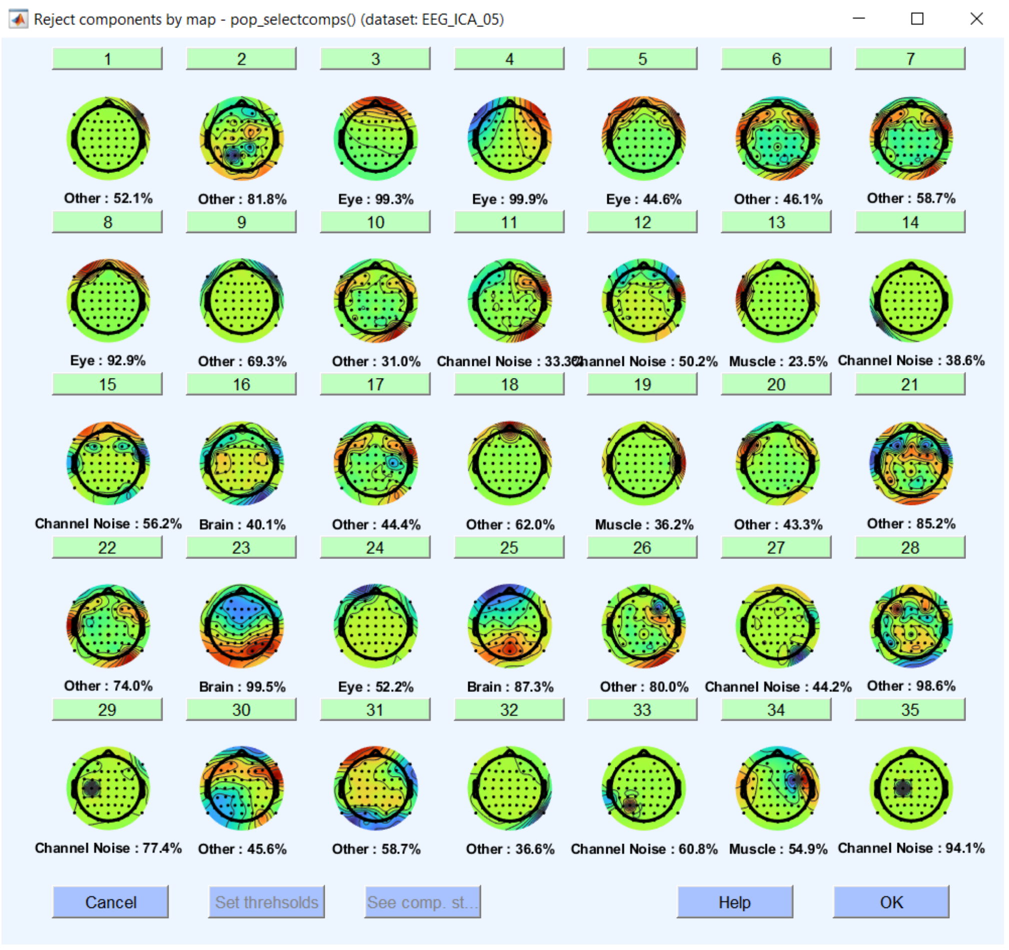

Figure 3.4: ICLabel classification of independent components. Each scalp topography corresponds to a component labeled with its most likely class and associated confidence score, generated with EEGLAB.

3.2.5 Epoch Extraction

Epoching, which refers to the segmentation of continuous EEG signals into discrete time windows, is a fundamental step in EEG analysis. This procedure isolates neural activity that is time-locked to specific events, thereby facilitating statistical comparisons and improving the signal-to-noise ratio. As noted by Cohen [30], epoching enhances temporal specificity and supports downstream processes such as baseline correction and condition-wise averaging.

During preprocessing, a mismatch was observed between the event markers embedded within the EEGLAB EEG structure, particularly in the event and urevent fields, and those listed in the external .txt annotation files. In most cases, the EEGLAB data contained more than 30 events with inconsistent labels, likely due to repeated or aborted trials being recorded differently across acquisition systems.

To resolve this inconsistency, all event markers in the EEGLAB structure were manually cross-checked against the reference annotation files for each participant. Events that could not be reliably matched were discarded, ensuring that each participant retained exactly 30 valid trials, as specified in the original experimental protocol. The only exception was Subject 6, whose EEG events could not be aligned with the corresponding annotations and was therefore excluded from further analysis.

Once the integrity of event alignment was verified, the continuous EEG recordings were segmented into 60-second epochs corresponding to each trial. In addition, a 3-second pre-stimulus interval was included in each epoch to support subsequent baseline correction procedures.

3.2.6 Baseline Removal

The final step in the preprocessing pipeline consisted of baseline correction. This procedure involves subtracting the mean voltage of a predefined pre-stimulus interval from the entire epoch, thereby adjusting for slow drifts and Direct Current (DC) offsets unrelated to neural activity. In this study, the 3-second interval preceding the onset of each trial was used as the baseline period.

Baseline correction is widely recognized as a critical step in EEG preprocessing, as it enhances the interpretability of event-related signals by removing low-frequency fluctuations that may obscure neural responses. Cohen [30] states that subtracting the average amplitude of the baseline window from each data point in the epoch ensures that observed variations are more accurately attributed to experimental events rather than to arbitrary voltage shifts.

Implementing this correction across all epochs allows for consistent comparisons across trials and participants, reinforcing the validity of subsequent analyses focused on trust-related neural dynamics.

With all preprocessing steps completed, the resulting EEG data were ready for subsequent analysis, including feature extraction and trust classification, as described in the following sections.

3.3 Experimental Evaluation Framework

3.3.1 Types of Approaches

One of the primary objectives of this thesis was to systematically evaluate different modeling strategies for classifying human trust using EEG signals. The analysis focused on how three key factors influenced classification performance: the type of feature representation, the modeling paradigm, and the data partitioning strategy. To this end, six experimental setups were designed, as illustrated in Figure 3.5. They combine two modeling paradigms (within-subject vs. between-subject), two types of feature representations (single vs. multiple EEG characteristics), and, within the within-subject paradigm, two data partitioning strategies (slice-wise vs. trial-wise).

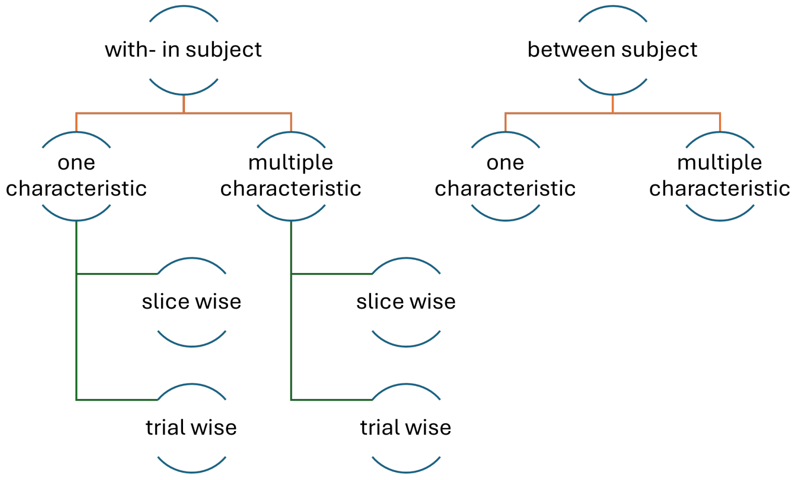

Figure 3.5: Overview of the six experimental configurations evaluated in this study, combining different EEG feature representations (one vs. multiple characteristics) with modeling paradigms (within-subject vs. between-subject) and data partitioning strategies (slice-wise vs. trial-wise).

The two modeling paradigms employed in this study differed in their focus on personalization versus generalization. In the within-subject approach, an individual model was trained and tested for each participant, capturing subject-specific neural patterns associated with trust. Conversely, the between-subject paradigm evaluated a single model's ability to generalize across individuals by training on data from multiple participants and testing on a previously unseen subject. Although more challenging, this approach better reflected real-world scenarios in which labeled data from new users may not be available.

Each strategy followed a distinct evaluation protocol tailored to its objective. In the within-subject setting, a hyperparameter search was conducted independently for each participant using 10-fold cross-validation on their training data. Once the optimal configuration had been identified per subject, the most frequently selected combination of hyperparameters across all individuals was determined. This most common configuration was then used to retrain a final model for each participant, ensuring consistency across models while still preserving subject-specific training data.

In the between-subject setting, a leave-one-subject-out (LOSO) scheme was employed: each participant was held out once as the test set, while the remaining subjects formed the training set. To prevent any data leakage, the training data were further split into internal folds to perform hyperparameter tuning. Final performance metrics were then averaged across all LOSO iterations to provide a robust estimate of the model's ability to generalize across individuals.

As part of the within-subject paradigm, two data partitioning strategies were explored. In the slice-wise approach, one-second EEG segments were randomly assigned to training and test sets, which may have resulted in segments from the same trial being split across both. The trial-wise strategy, in contrast, preserved the temporal integrity of 60-second trials by allocating entire trials to either training or testing, thereby avoiding any temporal overlap. While slice-wise splitting allowed training and test samples from the same trial to appear in both sets, potentially leading to overly optimistic results due to their temporal proximity, the trial-wise strategy enforced stricter separation by keeping trials intact, resulting in a more realistic and challenging evaluation of model performance within subjects.

In addition to evaluating different modeling paradigms and partitioning strategies, this study also investigated how the type of EEG feature representation affected classification performance. Two configurations were compared: one that relied exclusively on a single EEG feature and another that incorporated a broader set of features capturing diverse signal properties. A detailed description of the selected features, including their motivation and computation, is provided in the following section.

For clarity and consistency, the following terminology will be used throughout this thesis to refer to each experimental configuration:

- Approach 1: Within-subject paradigm with one characteristic and slice-wise partitioning.
- Approach 2: Within-subject paradigm with one characteristic and trial-wise partitioning.
- Approach 3: Within-subject paradigm with multiple characteristics and slice-wise partitioning.
- Approach 4: Within-subject paradigm with multiple characteristics and trial-wise partitioning.
- Approach 5: Between-subject paradigm with one characteristic.

- Approach 6: Between-subject paradigm with multiple characteristics.

3.3.2 Feature Extraction

To model the cognitive mechanisms underlying trust, it is crucial to extract and select meaningful EEG features. As outlined in the previous section, two feature representation strategies were adopted in this study: a single-feature approach based on Band Differential Entropy (BDE), and a multi-feature approach that combines BDE with five additional EEG-derived descriptors. This design allowed for evaluating whether feature diversity contributed to improved classification performance, or whether a single, well-supported descriptor was sufficient.

Band Differential Entropy

The single-feature approach relies on Differential Entropy (DE). When EEG signals are divided into different frequency bands, DE is computed for each band to assess how the signal behaves within that range. Specifically, when the signal remains stable and shows minimal variation over time, its variance is low, resulting in a low DE value. This indicates that signals with lower complexity exhibit more predictable patterns, and consequently, contain less differential entropy. When applied to specific frequency bands, this is referred to as Band Differential Entropy (BDE).

In the literature, BDE has demonstrated consistent performance across multiple cognitive tasks, including emotion recognition [38] and mental workload estimation [39]. Its relevance to trust modeling has also been documented, as entropy-based features have shown sensitivity to fluctuations in trust towards automated systems and robotic agents [40].

The computation of BDE in this study was carried out using the standard implementation provided by the TorchEEG library [41], which uses the NumPy and Torch frameworks to efficiently compute entropy-based features from EEG signals. They are based on the standard definition of differential entropy for a continuous random variable X with probability density function $f(x)$, which is given by:

$$h(X) = -\int f(x)\log(f(x))\,dx$$

Since directly estimating $f(x)$ can be computationally demanding, the signal is assumed to follow a Gaussian distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, under which the entropy simplifies to:

$$h(X) = \frac{1}{2}\log(2\pi e\sigma^2)$$

This expression for entropy $h(X)$ is then used to compute the DE for each frequency band of the EEG signal. This simplified form makes it possible to estimate BDE using only the standard deviation of the EEG signal within each frequency band. Since the standard deviation reflects how spread out the signal values are, higher dispersion implies greater uncertainty and, therefore, higher entropy. This means that more complex and variable brain activity results in higher BDE values.

In this work, BDE was computed for four frequency bands: theta (4–8 Hz), alpha (8–13 Hz), beta (13–30 Hz), and gamma (30–45 Hz). Although the delta band (0.5–4 Hz) was included in the preprocessing stage, it was excluded from the BDE computation to minimize the influence of low-frequency artifacts such as eye movements and muscle activity. This decision aligns with the reference paper as well as with the default configuration of the TorchEEG framework, which leaves out the delta band when estimating entropy due to its limited relevance in modeling higher-order cognitive states.

Although BDE provides a useful summary of EEG activity, it only captures one aspect of the signal's behavior, specifically the variability within each frequency band. However, trust is influenced by a variety of cognitive and emotional processes, and considering a broader range of signal characteristics could enhance the model's ability to detect patterns related to trust. For this reason, the feature set was extended to include additional aspects of the EEG signal.

The selection of complementary features was guided by their ability to capture distinct signal characteristics and their compatibility with the data structure required for the grid-based models employed (i.e., Convolutional Neural Networks and Vision Transformers). As with BDE, each feature was computed separately for four frequency bands: theta, alpha, beta, and gamma, using the TorchEEG library and the underlying Torch and NumPy frameworks.

Each selected feature is described below.

Band Power Spectral Density (PSD) measures how energy is distributed across frequency bands and is commonly used to explore cognitive and emotional states. In particular, changes in power within the alpha and beta bands have been shown to be associated with variations in user trust. For example, a recent study by Shahsavar and Choudhury [42] reported that changes in these bands were linked to fluctuations in user trust in a depression screening app.

In TorchEEG, PSD is computed using Welch's method, which estimates the average power over sliding windows. Given an EEG signal segment $x[n]$, the power spectral density $P(f)$ is calculated as:

$$P(f) = \frac{1}{L} \sum_{w=1}^{L} |\mathrm{FFT}\{x_w[n]\}|^2$$

where $x_w[n]$ is the w-th windowed segment, and L is the total number of windows. The resulting power in each predefined frequency band (theta, alpha, beta, gamma) is then extracted by integrating $P(f)$ over the band's frequency range.

As mentioned before, the selection of complementary features was guided by their ability to capture distinct signal characteristics. While the previously discussed characteristics, BDE and PSD, focus on aspects related to the frequency domain, it was decided to also include statistical measures, such as Band Skewness and Kurtosis, which provide additional insights into the distribution of the EEG signal. Moreover, there are studies that have successfully applied such metrics in similar contexts, such as one of those discussed in Section 2, where the authors of [16] used EEG signals to predict trust, employing statistical features, among others, to train their models. The explanation of these two features follows below.

Band Skewness describes the asymmetry in the amplitude distribution of EEG signals within each frequency band, providing a statistical measure of how the amplitudes deviate from a normal distribution. This measure is particularly useful for capturing the distributional characteristics of EEG signals, which can reveal patterns of asymmetry that might not be immediately apparent in simple amplitude measurements.

The TorchEEG library uses the standard statistical formula to calculate skewness:

$$\text{Skewness}(X) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \mu}{\sigma}\right)^3 \tag{3.1}$$

where x_i represents the signal amplitude, μ is the mean, σ is the standard deviation, and n is the number of time samples within the EEG segment, in this case $n = 250$.

Band Kurtosis evaluates the peakedness or flatness of the distribution of EEG signal amplitudes within each frequency band. Higher kurtosis values suggest that the signal contains more extreme deviations, such as sharp bursts of neural activity. According to the literature, these bursts can occur in response to unexpected events or performance errors [43]. In the context of our study, such variations could reflect moments when the user's confidence fluctuates due to unexpected or incorrect system behavior. This could provide valuable insights into this study.

Its calculation in TorchEEG is based on the following formula:

$$\text{Kurtosis}(x) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \mu}{\sigma}\right)^4 \tag{3.2}$$

where x represents the EEG signal, μ is the mean, σ is the standard deviation, and $n = 250$ is the number of time samples per segment.

Up to this point, features related to the frequency domain and statistical measures have been considered. To complete the feature set, two additional characteristics related to the time domain are introduced: Band Sample Entropy and Band Hjorth Parameters.

Band Sample Entropy is a non-linear measure used to quantify the irregularity and unpredictability of EEG signals over time. It evaluates how frequently similar patterns of signal values repeat within a time window, helping to understand how much the signal changes over time. Higher values indicate less predictable, more irregular brain activity. [44]

Although it may appear similar to BDE, these features capture different aspects of the signal. While BDE summarizes the overall variability within each frequency band, Sample Entropy (SE) focuses on the internal structure of the signal, assessing whether similar patterns recur over time. This makes SE useful for identifying subtle variations in how brain signals evolve over time. Since BDE has proven useful for modeling trust from EEG data, it is logical to include another feature that also measures signal irregularity, but in a different way. SE accomplishes this without assuming how the signal is distributed. Using both measures together allows for the exploration of whether they provide complementary information, potentially enhancing model performance.

Sample Entropy is computed by TorchEEG using the following definition:

$$\mathrm{SampEn}(m, r, N) = -\log\left(\frac{A}{B}\right) \tag{3.3}$$

where m is the embedding dimension, r is the tolerance (typically a fraction of the signal's standard deviation), N is the signal length, A is the number of matching patterns of length $m+1$, and B is the number of matching patterns of length m.

Band Hjorth Parameters include Activity, Mobility, and Complexity, which describe how the EEG signal behaves over time [45]. Activity measures the overall power of the signal (its variance), Mobility relates to the average frequency, and Complexity indicates how quickly the frequency changes over time.

The parameters are calculated as follows:

$$\text{Activity} = \mathrm{Var}(x(t)) \tag{3.4}$$

$$\text{Mobility} = \sqrt{\frac{\mathrm{Var}(\mathrm{d}x(t)/\mathrm{d}t)}{\mathrm{Var}(x(t))}} \tag{3.5}$$

$$\text{Complexity} = \frac{\text{Mobility}(\mathrm{d}x(t)/\mathrm{d}t)}{\text{Mobility}(x(t))} \tag{3.6}$$

Other EEG features could also provide useful information, but the main objective of this work was not to find the most informative features, but to compare different representation strategies. The selected features cover aspects of the frequency domain (DE and PSD), time domain (Sample Entropy and Hjorth Parameters), and statistical measures (skewness and kurtosis), providing a solid starting point for evaluating the potential benefits of including these types of features.

3.4 Artificial Intelligence Models for EEG Classification

The approaches discussed so far were applied using a combination of traditional machine learning algorithms and deep learning architectures. The former were selected for their interpretability and low computational cost, while the latter capture the complex temporal and spatial structure inherent in EEG signals.

The models evaluated include five traditional machine learning algorithms, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), and Extreme Gradient Boosting (XGBoost), along with two deep learning approaches: a Convolutional Neural Network (CNN) and a Vision Transformer (ViT). These two last models are particularly suited for EEG analysis, as they can learn not only temporal dynamics but also spatial dependencies between electrodes. Due to their central role in the proposed framework, both architectures are described in greater detail in the following subsections.

All traditional machine learning models were implemented in Python using the scikit-learn library [46], with the exception of XGBoost, which was implemented using its official Python package. All input features were standardized using z-score normalization prior to model fitting. This was done to account for scale sensitivity in distance-based models such as KNN and SVM, and to ensure numerical consistency and comparability across all classifiers. Although some models like Random Forest or Naive Bayes do not inherently require feature scaling, applying the same preprocessing pipeline across all models ensured a fair and consistent evaluation framework. For these algorithms, in order to optimize performance and reduce overfitting, hyperparameter tuning was performed via cross-validation. A summary of the set of hyperparameters explored for each is provided in Table 3.2 at the end of this section.

3.4.1 K-Nearest Neighbors

The KNN algorithm is a non-parametric, instance-based supervised learning method frequently used for classification tasks [47]. It assumes that similar instances exist in close proximity in the feature space and makes predictions based on the majority class among the k closest samples from the training set.

Given a new input vector $\mathbf{x}$, the algorithm calculates the distance to all training samples $\mathbf{x}_i$ and selects the k nearest neighbors based on a chosen distance metric. The most commonly used distance is the Euclidean distance, defined as:

$$d(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2} \tag{3.7}$$

where n is the number of features. The predicted class is then assigned by majority voting among the k neighbors.

The value of k is a hyperparameter that controls the algorithm's complexity and sensitivity to noise. A small k may lead to overfitting, while a large k can cause underfitting by oversmoothing decision boundaries.

3.4.2 Support Vector Machine

SVM are supervised learning models based on statistical learning theory, widely applied in binary classification tasks [48]. The core principle is to construct an optimal hyperplane that separates samples of different classes with the maximum possible margin in the feature space.

Given a labeled training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, where $\mathbf{x}_i \in \mathbb{R}^n$ represents the input features and $y_i \in \{-1, +1\}$ the corresponding class labels, the SVM aims to find a hyperplane defined by a weight vector $\mathbf{w}$ and a bias term b such that:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \quad \text{for } y_i = +1, \tag{3.8}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{for } y_i = -1. \tag{3.9}$$

These two inequalities ensure that data points from opposite classes are not only correctly classified, but also lie at least one unit away from the decision boundary. The margin, which is the distance between the closest points of each class (the support vectors), is therefore maximized. These constraints can be consolidated into a single condition, expressed as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i, \tag{3.10}$$

which formalizes the requirement that all training samples lie on the correct side of the margin.

To improve robustness in real-world settings, where data may include outliers or classes are not linearly separable, the soft-margin formulation is commonly adopted. This approach introduces slack variables that allow for some classification errors, and a regularization parameter C that controls the trade-off between maximizing the margin and minimizing misclassification.

When the data are not linearly separable in the original space, SVM can be extended using kernel functions $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ that implicitly project the inputs into a higher-dimensional feature space. This allows the algorithm to construct non-linear decision boundaries. Common kernel choices include the linear, polynomial, radial basis function (RBF), and sigmoid kernels.

For this model, in one of the experimental settings, hyperparameter optimization was skipped due to the high computational cost. In that case, the most frequently selected values from previous runs were reused. This particular case is discussed in more detail in Section 4.1.

3.4.3 Naive Bayes

NB classifiers are a family of probabilistic models derived from Bayes' theorem, which assume conditional independence between features given the class label [49]. Despite this simplifying assumption, NB classifiers have demonstrated competitive performance in various high-dimensional domains, including text classification, biomedical signal processing, and, as in this work, EEG-based classification tasks [50].

The decision rule in NB is based on estimating the posterior probability of a class y given an input feature vector $\mathbf{x}$. According to Bayes' theorem:

$$P(y \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid y)P(y)}{P(\mathbf{x})}, \tag{3.11}$$

where:

- $P(y \mid \mathbf{x})$ is the posterior probability of class y given the observation $\mathbf{x}$,
- $P(\mathbf{x} \mid y)$ is the likelihood of observing $\mathbf{x}$ under class y,
- $P(y)$ is the prior probability of class y,
- and $P(\mathbf{x})$ is the marginal probability of the input.

The core assumption of Naive Bayes is that the features x_j are conditionally independent given the class label. This allows the likelihood to be factorized as:

$$P(\mathbf{x} \mid y) = \prod_{j=1}^{n} P(x_j \mid y), \tag{3.12}$$

which greatly simplifies computation and enables efficient parameter estimation using simple statistics from the training set.

Depending on the assumed distribution of each feature, different variants of NB exist. In this study, the Gaussian Naive Bayes variant was adopted, which assumes that each feature follows a normal distribution within each class. The likelihood is then computed as:

$$P(x_j \mid y) = \frac{1}{\sqrt{2\pi\sigma_{y,j}^2}} \exp\left(-\frac{(x_j - \mu_{y,j})^2}{2\sigma_{y,j}^2}\right), \tag{3.13}$$

where $\mu_{y,j}$ and $\sigma_{y,j}^2$ represent the mean and variance of feature x_j for class y, respectively.

3.4.4 Random Forest

RF is an ensemble learning method that constructs a collection of decision trees and aggregates their outputs to perform classification or regression [51]. It belongs to the family of bagging (Bootstrap Aggregating) algorithms, designed to improve model stability and generalization by combining multiple learners trained on random subsets of the data.

Each individual tree in the forest is trained on a bootstrap sample, which is a randomly drawn subset of the training data with replacement. Additionally, at each node split, a random subset of features is selected rather than considering the full set, which helps to decorrelate the trees and reduce overfitting.

For classification, the final output is determined by majority voting across all decision trees:

$$\hat{y} = \text{mode}\{T_1(\mathbf{x}), T_2(\mathbf{x}), \ldots, T_M(\mathbf{x})\}, \tag{3.14}$$

where $T_i(\mathbf{x})$ denotes the prediction made by the i-th decision tree, and M is the total number of trees in the ensemble.

Random Forests are particularly well-suited for high-dimensional data such as EEG signals due to their ability to handle non-linear relationships, resistance to overfitting, and capacity to estimate feature importance. In addition, they are non-parametric, requiring minimal assumptions about the data distribution.

For this model, in the multi-feature setup, the model was also used to identify the most important EEG features, which were subsequently visualized and analyzed.

3.4.5 Extreme Gradient Boosting

XGBoost is a scalable and efficient implementation of the gradient boosting framework, widely used in classification and regression tasks due to its predictive performance and regularization capabilities [52]. It belongs to the family of boosting algorithms, which build an ensemble of weak learners, typically decision trees, by sequentially adding models that correct the residual errors of previous ones.

In contrast to bagging methods like Random Forest, which train trees independently, boosting constructs models in a stage-wise manner. At each iteration, a new tree is added to minimize a differentiable loss function through gradient descent. The prediction at iteration t is computed as:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t(\mathbf{x}), \tag{3.15}$$

where f_t is the newly added regression tree that fits the gradient of the loss function with respect to the current prediction $\hat{y}^{(t-1)}$.

XGBoost incorporates several enhancements over standard gradient boosting, including regularization terms to prevent overfitting, parallelized tree construction, and handling of missing values. The objective function minimized during training is defined as:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k), \tag{3.16}$$

where l is a differentiable convex loss function (e.g., log-loss for classification), and $\Omega(f_k)$ is a regularization term controlling the complexity of each tree.

Due to its ability to model complex non-linear patterns and robustness to noisy or redundant features, XGBoost is particularly effective for high-dimensional datasets such as EEG. As in Random Forest, feature importance scores were used to identify key EEG features in the multi-feature setup.

3.4.6 Convolutional Neural Network

CNNs are a class of feedforward neural networks designed to exploit the spatial structure present in data such as images or topographically organized signals [53]. Their core strength lies in their ability to learn patterns not only from feature magnitudes but also from their spatial relationships. This capability is particularly useful in the analysis of biomedical signals like EEG, where spatial proximity between electrodes corresponds to functional brain regions.

CNNs are typically composed of a sequence of convolutional layers, activation functions, pooling layers, and fully connected layers.

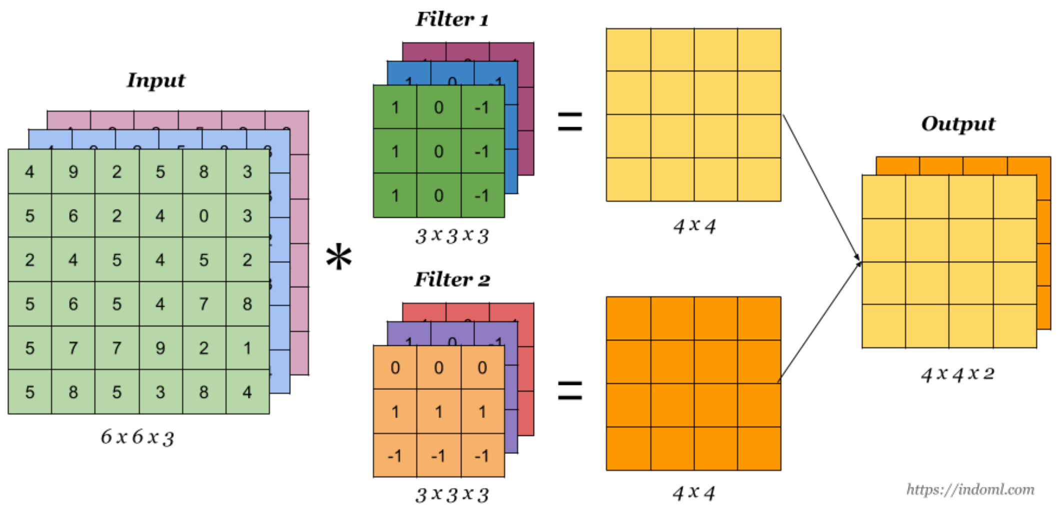

Figure 3.6: Illustration of a convolutional layer with multi-channel input and multiple filters. Each filter generates a corresponding feature map by performing convolution operations over local patches of the input. Reproduced from [1].

At the core of CNNs are the convolutional layers, which apply learnable filters (also called kernels) across the input to extract local spatial features. Each filter performs a dot product between its weights and local regions of the input, generating a feature map that highlights areas where specific patterns are detected. As illustrated in Figure 3.6, multiple filters operate in parallel to extract different types of features, such as edges, textures, or orientations. Filters are initialized randomly and optimized during training via backpropagation. In early layers, they typically capture low-level features, such as edges or corners, whereas deeper layers encode more abstract and semantic information [54].

Two key parameters influence the behavior of convolutions: padding and stride. Padding involves adding artificial borders around the input. This technique ensures that features near the edges are not lost and helps preserve spatial resolution. Common strategies include:

- Valid padding: no padding is applied, reducing the spatial dimensions of the output.
- Same padding: padding is applied to maintain the same spatial dimensions as the input.
- Zero padding: border values are set to zero, the most frequent implementation.

The stride determines the step size by which the filter moves across the input. A stride of one provides dense sampling, while higher strides downsample the input more aggressively, reducing the output resolution. By adjusting stride and padding, CNNs can control the granularity and resolution of learned features [55].

Following convolution, a non-linear activation function is applied to the feature maps. The most commonly used function is the Rectified Linear Unit (ReLU), which introduces non-linearity and helps the network learn complex input-output mappings by suppressing negative values [56].

To further reduce the computational burden and capture translational invariance, pooling layers are employed. These layers downsample the feature maps by summarizing local neighborhoods. Two common approaches are:

- Max pooling: retains the maximum value within each region.
- Average pooling: computes the mean value within each region.

After multiple layers of convolutions, activations, and pooling, the resulting feature maps are flattened into one-dimensional vectors and passed to fully connected layers. These dense layers integrate the spatial information extracted previously and perform the final classification. The output layer typically employs a softmax function to convert raw scores into a probability distribution over the target classes.

To mitigate overfitting and enhance generalization, dropout regularization is applied in the fully connected layers [57]. This technique randomly deactivates a proportion of neurons during training, forcing the model to learn more robust and distributed representations.

The entire CNN is trained using a forward-backward cycle. In the forward pass, data propagates through the network to produce predictions. In the backward pass, the error is computed using a loss function and gradients are propagated backwards to update the model weights [58].

A visual summary of this process is presented in Figure 3.7, which illustrates the overall structure of a CNN. It shows how the input is successively transformed through convolutional, activation, and pooling layers, followed by flattening and fully connected layers that ultimately produce class probabilities.

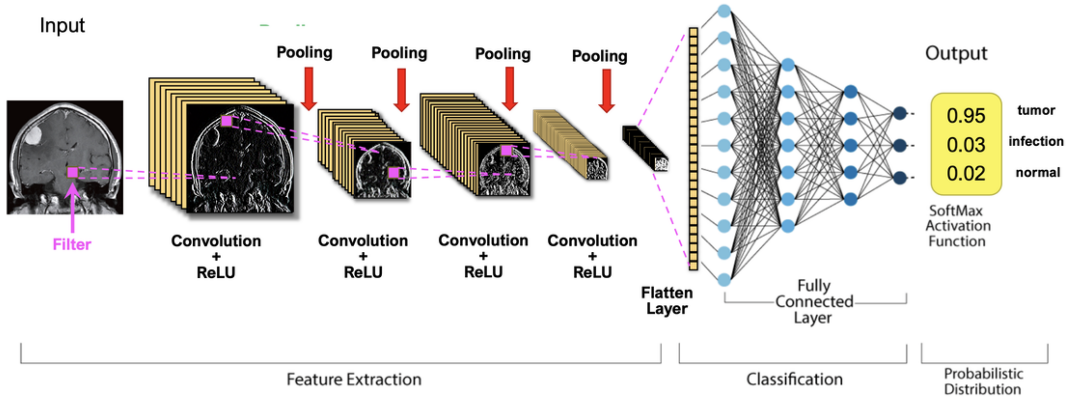

Figure 3.7: Schematic representation of a CNN workflow. The input is processed through convolutional and pooling layers to extract features, followed by flattening and fully connected layers for classification. The final output is a probability distribution over the target classes. Extracted from [2].

3.4.6.1 Spatial Grid Mapping of EEG Signals

As previously discussed, one of the core advantages of using CNNs lies in their capacity to exploit not only the magnitude of input features but also their spatial arrangement. In the context of EEG, this spatial structure is defined by the physical placement of electrodes across the scalp, where each electrode captures neural activity from a distinct cortical region. This setup is similar to how pixels work in an image; nearby pixels often carry related information. Likewise, nearby EEG electrodes capture related brain activity.

To preserve the spatial structure of the EEG data, each segment is first represented as a sequence of feature vectors $[s_1, s_2, \ldots, s_n]^\top \in \mathbb{R}^{n \times d_f}$, where n denotes the number of electrodes and d_f the number of features extracted per channel.

Rather than treating this sequence as an unordered list, the feature vectors are spatially arranged on a grid that reflects the physical layout of the electrodes according to the international 10–20 system. This arrangement is defined by a transformation matrix that specifies the position of each electrode on the grid, allowing the spatial topology of brain activity to be preserved. Locations without corresponding electrodes are zero-padded to maintain uniform dimensions.

For this project, this concept was realized as follows. Since the dataset employed uses a 64-channel EEG montage, the same 9×9 transformation matrix described in the reference study was adopted. This ensures compatibility with the original electrode configuration used during data collection, which is essential for accurately replicating the experimental setup. The transformation matrix used for spatial mapping is shown below. Each cell corresponds to an electrode label, and empty positions, marked with dashes, are assigned a value of zero in the processed data.

$$T = \begin{bmatrix} - & - & \mathrm{FP1} & - & \mathrm{FPZ} & - & \mathrm{FP2} & - & - \\ \mathrm{F9} & \mathrm{AF7} & \mathrm{AF3} & - & - & - & \mathrm{AF4} & \mathrm{AF8} & \mathrm{F10} \\ \mathrm{F7} & \mathrm{F5} & \mathrm{F3} & \mathrm{F1} & \mathrm{FZ} & \mathrm{F2} & \mathrm{F4} & \mathrm{F6} & \mathrm{F8} \\ \mathrm{FT7} & \mathrm{FC5} & \mathrm{FC3} & \mathrm{FC1} & \mathrm{FCZ} & \mathrm{FC2} & \mathrm{FC4} & \mathrm{FC6} & \mathrm{FT8} \\ \mathrm{T7} & \mathrm{C5} & \mathrm{C3} & \mathrm{C1} & \mathrm{CZ} & \mathrm{C2} & \mathrm{C4} & \mathrm{C6} & \mathrm{T8} \\ \mathrm{TP7} & \mathrm{CP5} & \mathrm{CP3} & \mathrm{CP1} & \mathrm{CPZ} & \mathrm{CP2} & \mathrm{CP4} & \mathrm{CP6} & \mathrm{TP8} \\ \mathrm{P7} & \mathrm{P5} & \mathrm{P3} & \mathrm{P1} & \mathrm{PZ} & \mathrm{P2} & \mathrm{P4} & \mathrm{P6} & \mathrm{P8} \\ \mathrm{P9} & \mathrm{PO7} & \mathrm{PO3} & - & \mathrm{POZ} & - & \mathrm{PO4} & \mathrm{PO8} & \mathrm{P10} \\ - & - & \mathrm{O1} & - & \mathrm{OZ} & - & \mathrm{O2} & - & - \end{bmatrix}$$

The resulting spatial representation serves as the input to the CNN architecture, effectively embedding anatomical structure into the learning process. An illustration of the full mapping pipeline is presented in Figure 3.8.

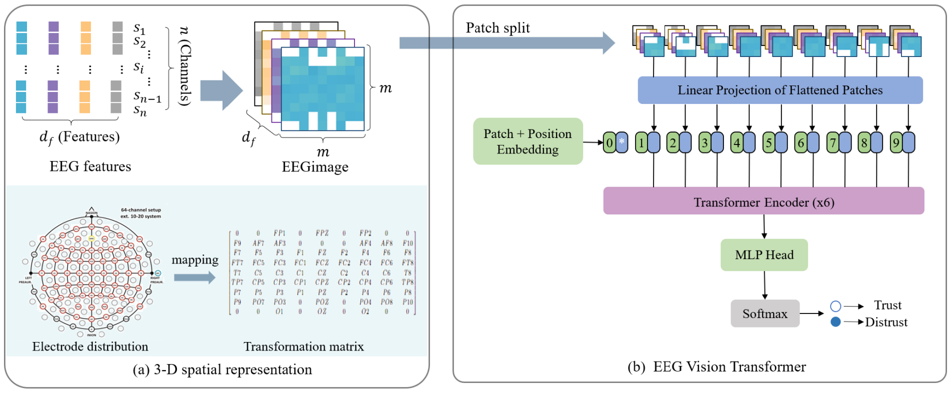

Figure 3.8: (a) Mapping of EEG channels into a 9 × 9 spatial grid based on electrode positions. (b) CNN input representation and feature flow. Extracted from [3].

3.4.6.2 Implementation Details

The CNN model used in this study was implemented using the TorchEEG library [41], which provides EEG-specific deep learning architectures and preprocessing utilities. The implementation was based on the default CNN architecture, which consists of four convolutional layers with ReLU activations and zero padding. These layers progressively increase the number of filters from 64 to 256 and then reduce it again, allowing the network to capture both low-level and high-level spatial features. Following the convolutional stages, two fully connected layers were used for classification. To enhance generalization, a Scaled Exponential Linear Unit

(SELU) activation was applied in the dense layers, and dropout with a probability of 0.5 was introduced before the output layer as a regularization strategy.

For the construction of spatial inputs, the ToGrid transformation module was applied, using a custom 9×9 layout consistent with the electrode configuration described in Section 3.4.6.1.

Finally, a fixed set of hyperparameters was used across all experiments involving the CNN model. Training was conducted using the Adam optimizer with both the learning rate and weight decay set to 10^{-4}, over a total of 500 epochs. These values were not further optimized in order to reduce computational cost, considering the high training time associated with deep neural networks. This configuration is consistent with both the original EEGTrust study, where all subjects were trained using the same hyperparameters, and the default settings provided by the TorchEEG implementation of the CNN model. Cross-entropy loss was used as the objective function.

For clarity and to provide a better understanding, it is important to note that, for each subject, there were 30 trials, each lasting 60 seconds. The EEG signals were divided into 1-second segments, resulting in a total of 1800 samples per subject. Each sample corresponds to a feature map of either $(4, 9, 9)$ or $(24, 9, 9)$, depending on whether a single feature or multiple features (6 per band) were extracted for each frequency band. These 1800 samples were used as input to the CNN, with each sample representing the EEG data for a single second of the experiment.

3.4.7 Vision Transformer

Transformers are a class of machine learning models that rely on attention mechanisms as their core computational framework. Initially introduced for natural language processing tasks [59], Transformers have demonstrated remarkable flexibility and scalability across diverse modalities. The ViT, proposed by Dosovitskiy et al. [4], extends this architecture to the domain of image classification by representing images as sequences of visual tokens, analogous to the treatment of words in text.

In contrast to conventional Transformer models, which consist of both encoder and decoder modules, ViT utilizes only the encoder component. The output of this encoder is subsequently passed through a lightweight classification head to produce the final prediction (see Figure 3.9). In the present work, EEG signals were represented as three-dimensional spatial tensors based on the topological layout of the electrodes, following the same 9×9 spatial grid structure described in the CNN architecture. This format allows the EEG data to be treated analogously to images, thereby enabling the direct application of Vision Transformer-based models.

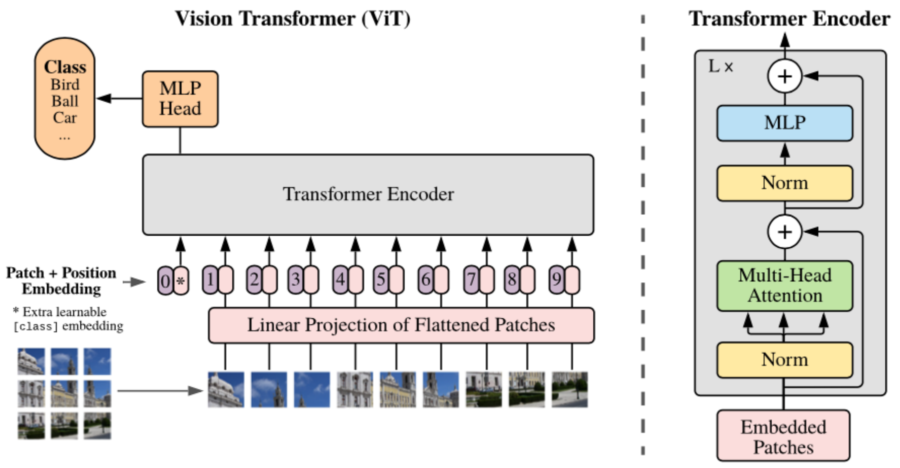

Figure 3.9: ViT architecture overview. The input image is divided into fixed-size patches, which are flattened and linearly projected. A positional encoding is added to each patch embedding, and a learnable classification token is prepended. The resulting sequence is processed by a stack of Transformer encoder layers. The output corresponding to the classification token is used for final prediction. Adapted from [4].

To process visual data, ViT first divides a two-dimensional image $x \in \mathbb{R}^{H\times W\times C}$ into non-overlapping patches of size $P \times P$. Each patch is then flattened into a one-dimensional vector of size $P^2 \cdot C$, resulting in a sequence $x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, where H, W, and C denote the height, width, and number of channels of the image, respectively; P is the patch size; and $N = \frac{HW}{P^2}$ is the total number of patches. These patch vectors are then passed through a shared linear projection that maps them into an embedding space of dimension D.

As Transformers do not encode positional information natively, positional encodings are added to the patch embeddings. These encodings, which are learnable and of the same dimensionality as the embeddings, allow the model to retain spatial information about the relative positions of patches within the original image.

A learnable classification token [CLS] is added at the sequence before it is fed into the encoder. This token does not correspond to any image patch but is intended to aggregate contextual information from the entire sequence through self-attention. After processing the sequence through multiple Transformer encoder layers, the final embedding of the [CLS] token is used for classification.

Each encoder layer consists of two primary components: a Multi-Head Self-Attention (MSA) mechanism and a feedforward multilayer perceptron (MLP). The MSA module enables each token to attend to all others in the sequence, allowing the model to learn long-range dependencies and global relationships between image regions from the very first layer. In contrast to convolutional operations, which are inherently local, self-attention facilitates global information exchange without spatial constraints. The term “multi-head” in this context refers to the use of several self-attention operations performed in parallel, each with its own set of parameters. This enables the model to simultaneously capture different types of relationships among the input tokens. An illustration of the MSA mechanism is presented in Figure 3.10.

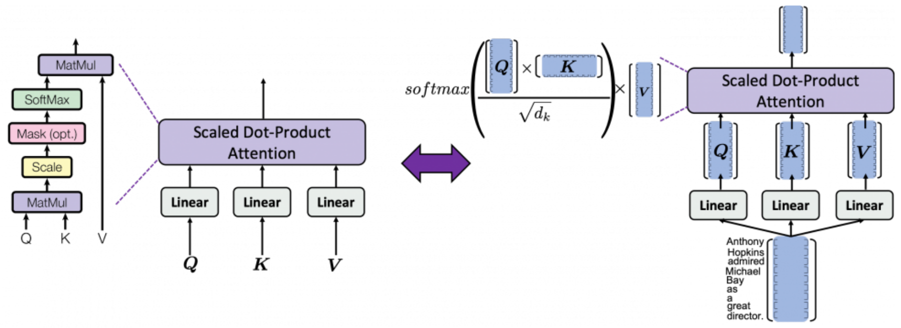
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Scaled Dot-Product Attention
Linear
Linear
Linear
Q
K
V
softmax
Q
K
V
$\sqrt{d_k}$
Scaled Dot-Product Attention
Q
K
V
Linear
Linear
Linear
Anthony Hopkins admired Michael Bay as a great director.

Figure 3.10: Diagram of the MSA mechanism. Queries (Q), keys (K), and values (V) are derived through separate linear projections. Multiple attention heads compute scaled dot-product attention in parallel. The outputs are then concatenated and passed through a final linear layer. Reproduced from [5].

Formally, given an input sequence $z \in \mathbb{R}^{N\times D}$, where N denotes the number of tokens and D the embedding dimension, the sequence is linearly projected into queries (q), keys (k), and values (v) using a shared weight matrix $U_{qkv} \in \mathbb{R}^{D\times 3D_h}$:

$$[q, k, v] = zU_{qkv}$$

The attention mechanism compares each query with all keys using dot products to compute similarity scores. These scores are then scaled by $\sqrt{D_h}$ and normalized using a softmax function to obtain attention weights. This operation, known as scaled dot-product attention, is defined as:

$$A = \text{softmax}\left(\frac{qk^\top}{\sqrt{D_h}}\right), \quad SA(z) = Av$$

where $SA(z)$ denotes the self-attention output.

The output of the MSA mechanism is obtained by applying the self-attention operation across k different heads in parallel. Each head computes an independent attention output, and the results are concatenated and linearly projected as follows:

$$MSA(z) = [SA_1(z); SA_2(z); \ldots ; SA_k(z)]U_{\text{msa}}, \quad U_{\text{msa}} \in \mathbb{R}^{kD_h \times D}$$

Following the attention block, the output is passed through an MLP consisting of two fully connected layers with a Gaussian Error Linear Unit (GELU) activation in between. Each sub-layer (MSA and MLP) is wrapped with residual connections and preceded by layer normalization, a design that facilitates gradient flow and improves training stability.

This encoder structure is repeated multiple times, commonly 12, 16, or 24 layers, depending on the model configuration. While the sequence length and embedding dimension remain fixed, the token representations become progressively more abstract and semantically rich across layers.

After the final encoder layer, the output representation is aggregated for classification. This is typically done by either using the [CLS] token embedding or applying mean pooling across all patch tokens. The resulting vector, of dimension D, is then passed through a classification head, typically a linear layer, which maps the embedding to class probabilities and produces the final output of the model.

3.4.7.1 ViT Implementation

The ViT model was also implemented using the TorchEEG library. The architecture consists of six Transformer encoder layers, each with nine attention heads and a hidden embedding dimension of 128. Dropout regularization with a probability of 0.2 was applied to mitigate overfitting. The model was trained using the Adam optimizer with a learning rate of 10^{-4} and a weight decay of 10^{-5}, over 500 epochs.

Following the same procedure as in the CNN model, the ToGrid transformation module was applied using a custom 9×9 layout consistent with the electrode configuration described in Section 3.4.6.1. Moreover, no additional hyperparameter tuning was performed in order to reduce computational cost. This configuration aligns with both the original EEGTrust study and the default settings provided by TorchEEG.

Model	**Hyperparameter Search**	**Tuned Parameters**
KNN	Yes	Number of neighbors (k), distance metric (Euclidean or Manhattan), weighting strategy (uniform or distance-based)
SVM	Yes (not for every approach)	Kernel type (linear or RBF), regularization parameter (C), kernel coefficient (γ, for RBF kernel)
NB	Yes	Variance smoothing parameter (`var_smoothing`)
RF	Yes	Number of estimators, maximum tree depth, minimum number of samples required to split an internal node
XGBoost	Yes	Learning rate, maximum tree depth, number of estimators
CNN	No	None
ViT	No	None

Table 3.2: Hyperparameter search configuration for each AI model evaluated in this study.

Results and Discussion

This section presents the model's behavior across the different experimental setups along with key contextual insights that support the interpretation of results.

Before evaluating the performance of the classification models, it is important to contextualize the trust patterns captured in the dataset. As described in Section 3.1, participants provided trust-related evaluations immediately after each trial through a structured questionnaire. These subjective ratings, stored as part of the dataset, reflect participants' perceptions of the robot's ability, collaborative behavior, and overall trustworthiness.

To better understand how these ratings vary across experimental conditions, Figure 4.1, extracted from the reference paper, presents the distribution of self-reported trust scores as a function of robot ability (HAR, MAR, LAR) and task difficulty (on a 1–5 scale). As detailed in the original study, high-ability robots (HAR) generally elicited higher trust ratings, while low-ability robots (LAR) received the lowest. Medium-ability robots (MAR) consistently occupied an intermediate position. Additionally, tasks involving simpler layouts resulted in higher and more distinguishable trust levels across robot types, whereas more complex layouts tended to reduce both the overall trust ratings and the separability between conditions.

These findings not only confirm the experimental design's effectiveness in inducing distinct trust responses, but also highlight the notable variability observed both within and across participants. This variability must be carefully considered, as it presents one of the main challenges in EEG-based trust prediction. The following analysis explores how this variability is reflected across different experimental configurations and evaluates the extent to which various modeling paradigms are capable of capturing generalizable or subject-specific trust patterns.

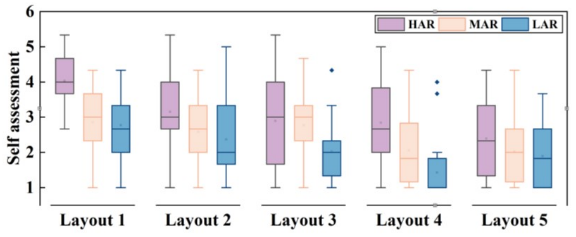

Figure 4.1: Distribution of perceived trust scores across different levels of robot performance and task complexity, extracted from [3].

Before presenting the results, it is useful to outline the expectations associated with each experimental configuration.

First, approaches based on trial-wise partitioning were expected to better reflect the model's true generalization ability, as they prevented overlap between training and test data from the same trial. In contrast, slice-wise configurations were expected to overestimate model accuracy due to temporal redundancy within trials.

Second, configurations using multiple EEG features were expected to outperform those using only Band Differential Entropy (BDE), since the additional descriptors, covering temporal, spectral, and statistical domains, could provide complementary information relevant to trust dynamics. Because BDE was part of both setups, better performance with the multi-feature configuration would have implied that the added features enhanced the model's ability to capture trust-related patterns.

Lastly, given the limited number of participants in the dataset, within-subject models were expected to perform better than between-subject models. Capturing generalized trust-related patterns across individuals was considered a significant challenge, particularly in EEG-based classification tasks, where inter-subject variability was known to be high. The between-subject paradigm, although closer to real-world cases where subject-specific data may not be available, was likely to suffer from this constraint.

Although the EEGTrust dataset originally included EEG recordings from sixteen participants, only twelve were retained for the final evaluation. Participants 1, 2, and 3 were excluded due to an insufficient number of valid trials, while Subject 6 was discarded because of inconsistencies between event markers and EEG data that prevented proper trial alignment. To ensure consistency and data integrity throughout the experimental pipeline, only participants with a complete set of usable trials and reliable event segmentation were included in the modeling phase.

4.1 Performance of Proposed Classification Models

The following sections present the classification performance achieved across all evaluated approaches. The results are summarized in Tables 4.1 to 4.7, where each table corresponds to one classification model and includes the six experimental configurations defined previously.

For each configuration, the tables report the average accuracy and F1-score, along with their corresponding standard deviations, all expressed in percentage terms. This format allows for a consistent and comparative analysis across models, feature representations, and validation strategies.

As discussed in Section 3.4, hyperparameter optimization was carried out for most traditional machine learning models. The resulting hyperparameter values for both evaluation paradigms are summarized in Annex A. In the within-subject approach, where a separate model is trained for each subject, the best hyperparameters are reported per subject. In contrast, for the between-subject approach (LOSO), the best hyperparameters selected in each fold, each corresponding to a different test subject, are presented per test subject.

For the SVM model, which is the most computationally expensive among the evaluated models, hyperparameter tuning was attempted for approaches 5 and 6. However, due to the high training time required for each fold in the LOSO configuration, a full grid search became impractical. Instead, fixed hyperparameter values were adopted for these two approaches, specifically, $gamma$ = scale, $kernel$ = rbf, and $C = 10$, as these were the most consistently selected parameters across the other within-subject approaches.

Table 4.1 reports the performance of the KNN model. The highest accuracy and F1-score were achieved in the within-subject, slice-wise configuration using a single feature per band, reaching an accuracy of 70.79%. In contrast, the lowest performance was observed in the between-subject paradigm, where results remained similarly low regardless of whether one or multiple features were used, with accuracy values remaining low and tightly clustered between 48.53% and 50.55%.

KNN Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**70.79**	**6.73**	**72.04**	**7.19**
	One characteristic	Trial-wise	61.02	6.96	61.98	8.45
	Multiple characteristics	Slice-wise	66.78	5.26	68.12	5.53
	Multiple characteristics	Trial-wise	60.39	7.07	61.48	8.12
Between-subject	One characteristic		48.53	2.94	48.56	8.88
	Multiple characteristics		50.55	2.23	50.42	8.28

Table 4.1: KNN model performance across different evaluation paradigms and feature configurations.

Table 4.2 shows the results for the SVM model. The best results were achieved under the within-subject, slice-wise validation scheme using a single feature per band, reaching an accuracy of 77.34%. Incorporating multiple features in the same setting also yielded competitive performance, with an accuracy of 72.34%. In contrast, the between-subject paradigm led to substantially lower results, with accuracy scores ranging narrowly from 48.38% to 51.53% regardless of the feature set. Notably, the F1-score dropped to 46.42% when using a single characteristic, accompanied by a high standard deviation of 19.92%, indicating considerable instability across folds.

SVM Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**77.34**	**4.59**	**77.24**	**4.59**
	One characteristic	Trial-wise	62.15	5.83	61.93	5.87
	Multiple characteristics	Slice-wise	72.34	4.49	72.75	5.12
	Multiple characteristics	Trial-wise	62.45	5.35	62.19	6.31
Between-subject	One characteristic	–	48.38	3.57	46.42	19.92
	Multiple characteristics	–	51.53	3.47	51.51	11.26

Table 4.2: SVM model performance across different evaluation paradigms and feature configurations.

The performance of the NB model is summarized in Table 4.3. As reflected in the results, the model consistently underperformed across all configurations, failing to reach 60% accuracy in any setting. Differences between within- and between-subject paradigms, as well as between feature types, were minimal, suggesting that the model was generally unable to capture discriminative patterns relevant to the classification task. Particularly poor F1-scores were observed in the between-subject setting, with values dropping as low as 14.69%.

NB Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**58.52**	**6.68**	**53.53**	**18.45**
	One characteristic	Trial-wise	56.55	10.58	49.75	21.07
	Multiple characteristics	Slice-wise	58.17	5.71	54.54	19.09
	Multiple characteristics	Trial-wise	56.13	10.31	48.43	24.63
Between-subject	One characteristic	–	48.36	3.59	14.69	14.98
	Multiple characteristics	–	50.39	4.26	17.29	18.44

Table 4.3: NB model performance across different evaluation paradigms and feature configurations.

The Random Forest model followed a similar pattern to the other traditional classifiers, achieving its highest performance under the within-subject, slice-wise configuration with single-feature input, where it reached an accuracy of 77.04%. The inclusion of multiple features in this same setting also yielded competitive results, with only a slight reduction in performance. Conversely, between-subject results, shown in Table 4.4, remained notably lower, below 50% accuracy in all cases.

Random Forest Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**77.04**	**4.99**	**77.09**	**5.88**
	One characteristic	Trial-wise	65.09	6.46	65.02	9.51
	Multiple characteristics	Slice-wise	73.98	5.61	74.04	6.47
	Multiple characteristics	Trial-wise	64.70	8.08	64.25	10.44
Between-subject	One characteristic	–	48.92	5.47	50.25	16.08
	Multiple characteristics	–	49.49	4.85	52.09	15.26

Table 4.4: Random Forest model performance across different evaluation paradigms and feature configurations.

As shown in Table 4.5, the XGBoost model achieved the highest overall performance across all classifiers evaluated in this study. The best results were obtained in the within-subject, slice-wise configuration using a single feature per band, reaching an accuracy of 79.61% and an F1-score of 79.82%. Including multiple features also produced similarly strong outcomes in this setting. On the other hand, performance dropped significantly in the between-subject paradigm, with accuracy values just below 50%, although slightly better F1-scores were observed compared to other traditional models.

XGBoost Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**79.61**	**5.97**	**79.82**	**6.37**
	One characteristic	Trial-wise	64.65	5.92	64.46	7.61
	Multiple characteristics	Slice-wise	78.01	5.72	78.27	6.11
	Multiple characteristics	Trial-wise	64.03	7.70	64.17	9.13
Between-subject	One characteristic	–	49.95	5.23	53.67	11.47
	Multiple characteristics	–	49.86	5.60	51.72	12.13

Table 4.5: XGBoost model performance across different evaluation paradigms and feature configurations.

Moving into the deep learning models, as shown in Table 4.6, CNN performed best under the within-subject, slice-wise setting using a single feature per band, achieving an accuracy of 75.46% and an F1-score of 74.53%. However, incorporating multiple features led to a noticeable drop in performance across all configurations, suggesting that the CNN architecture used may not have effectively leveraged the added feature complexity. As with the other models, generalization in the between-subject paradigm remained limited, with performance falling close to chance level, particularly when multiple features were used.

CNN Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**75.46**	**8.52**	**74.53**	**12.08**
	One characteristic	Trial-wise	63.13	7.40	61.50	9.65
	Multiple characteristics	Slice-wise	65.94	6.31	66.26	7.95
	Multiple characteristics	Trial-wise	58.41	4.89	57.66	7.25
Between-subject	One characteristic	–	52.11	4.00	57.67	15.20
	Multiple characteristics	–	49.39	2.47	48.99	18.08

Table 4.6: CNN model performance across different evaluation paradigms and feature configurations.

Finally, Table 4.7 presents the results for the ViT model. The highest performance was observed in the within-subject, slice-wise setting using a single feature per band, with an accuracy of 75.12% and an F1-score of 73.52%. However, ViT showed a marked decline in performance when multiple features were used, particularly under the trial-wise and between-subject conditions. In the between-subject setting, classification accuracy remained near chance level, and F1-scores dropped sharply, especially for the one-feature configuration, which yielded only 39.01%. These results suggest that the ViT architecture, as implemented here, struggled to generalize across subjects and may be more sensitive to feature representation than other models.

TRANSFORMER Model						
Paradigm	**Feature type**	**Validation**	**Accuracy (%)**		**F1-score (%)**	
			Mean	SD	Mean	SD
Within-subject	One characteristic	Slice-wise	**75.12**	**8.86**	**73.52**	**13.78**
	One characteristic	Trial-wise	63.91	8.29	62.22	13.28
	Multiple characteristics	Slice-wise	58.47	5.57	57.34	8.03
	Multiple characteristics	Trial-wise	53.22	3.11	49.89	8.77
Between-subject	One characteristic	–	50.55	3.76	39.01	30.14
	Multiple characteristics	–	51.10	2.56	47.97	21.83

Table 4.7: Transformer model performance across different evaluation paradigms and feature configurations.

4.2 Comparison With Reference Paper Results

Since the primary goal of this study was to replicate the methodology and findings of the reference work, it is important to compare the results obtained in this project with those reported in the original study. The paper focused exclusively on Approaches 1 and 2, which correspond to the within-subject setting using a single characteristic, evaluated under both slice-wise and trial-wise validation, respectively. Accordingly, the analysis will, for now, focus exclusively on these configurations. To facilitate this comparison, Table 4.8 provides a summary of the accuracy values obtained by each model in both this work and the original study.

Model	Validation	This Study Proposal	Reference Paper
KNN	Slice-wise	70.79%	65.35%
	Trial-wise	61.02%	57.24%
Naive Bayes	Slice-wise	58.52%	58.98%
	Trial-wise	56.55%	58.47%
SVM	Slice-wise	77.34%	70.82%
	Trial-wise	62.15%	61.90%
CNN	Slice-wise	75.46%	74.19%
	Trial-wise	63.13%	49.97%
Transformer	Slice-wise	75.12%	74.99%
	Trial-wise	63.91%	62.00%

Table 4.8: Comparison of accuracy between this work and the reference study, under slice-wise and trial-wise validation for the within-subject setting using a single characteristic.

When comparing the outcomes, the results reveal several consistent trends across models. As anticipated, both the results of this study and those reported in the original paper show that models perform considerably better under the slice-wise validation setting compared to the trial-wise setup. This can be observed across all classifiers and is consistent with the explanation given in the reference paper, which highlights the temporal continuity of EEG signals. Segments within a single trial are likely to be highly correlated over time. When these segments are distributed between training and testing sets, as occurs in slice-wise cross-validation, models may capture trial-specific patterns that recur across both sets. This overlap can result in performance scores that are somewhat inflated and do not fully reflect generalization. In contrast, trial-wise validation, where full trials are left out during training, better reflects how the models would perform in real-world use.

Going more into the specific results, the outcomes obtained with traditional machine learning models show a generally improved performance compared to those reported in the original study, particularly in the cases of KNN and SVM. For instance, under the slice-wise approach, the KNN model achieves an accuracy of 70.79%, compared to 65.35% in the reference paper. Similarly, in the trial-wise setting, the accuracy reaches 61.02% versus 57.24%. The SVM model also showed notable gains, with an accuracy of 77.34% in this work versus 70.82% in the original under slice-wise validation, and 62.15% versus 61.90% under trial-wise. In contrast, the performance of the Naive Bayes classifier remained nearly identical to that of the original study, with only marginal differences observed across both validation settings, for example, 58.52% in this work vs. 58.98% in the reference under slice-wise.

The performance improvements observed in KNN and SVM can be primarily attributed to the hyperparameter tuning carried out in this work. For instance, while the original study employed $n\_neighbors = 3$ for the KNN model, this work identified 9 and 7 as the optimal values for the slice-wise and trial-wise configurations, respectively.

A similar trend was seen in the SVM model. The original paper used $gamma =$ auto, $kernel =$ poly, and $C = 100{,}000$, whereas this work achieved better results with a more commonly used configuration: $gamma =$ scale, $kernel =$ RBF, and $C = 10$.

In the case of Naive Bayes, the original configuration set $var\_smoothing = 1\text{e}{-2}$, while a smaller value of $1\text{e}{-9}$ was adopted in this work. However, the resulting performance differences were minimal, suggesting that this hyperparameter has a more limited influence on classification outcomes compared to the more flexible tuning parameters in models like KNN or SVM.

Looking at the F1-score, the results follow a very similar trend to the accuracy results, as shown in Table 4.9. This makes sense because the dataset was balanced, so precision and recall were probably similar, which leads to F1-scores that are almost the same as the accuracy values.

Model	**Validation**	**This Study Proposal**	**Reference Paper**
KNN	Slice-wise	72.04%	63.15%
	Trial-wise	61.98%	56.98%
Naive Bayes	Slice-wise	53.53%	54.69%
	Trial-wise	49.75%	57.11%
SVM	Slice-wise	77.24%	68.12%
	Trial-wise	61.93%	58.40%
CNN	Slice-wise	74.53%	73.79%
	Trial-wise	61.50%	41.38%
Transformer	Slice-wise	73.52%	73.41%
	Trial-wise	62.22%	62.59%

Table 4.9: Comparison of F1-score between this work and the reference study, under slice-wise and trial-wise validation for the within-subject setting using a single characteristic.

Regarding the deep learning models, CNN and Transformer, certain inconsistencies were found between the parameters described in the original paper and those present in the publicly available implementation. Additionally, while code was provided for both the slice-wise and trial-wise approaches in all other models, this was not the case for CNN, where the implementation for the trial-wise configuration was not available. For these models, as already mentioned in section 3.4, no hyperparameter tuning was performed due to computational cost constraints. Instead, the configurations reported in the original paper were adopted directly. These included a batch size of 128, a learning rate of 1e−4, 500 training epochs, and the Adam optimizer, which was used to update the model weights during training. To guide the learning process, cross-entropy loss was used as the loss function.

Regarding the CNN model, the results obtained in this project for the slice-wise configuration are very close to those reported in the reference study, which is expected since the same hyperparameters were used. Specifically, this work achieved an accuracy of 75.46%, compared to 74.19% in the original. However, under the trial-wise setting, the difference between the two implementations became more noticeable. Both showed lower performance in this setting, which is reasonable given that trial-wise validation is generally more demanding. In our case, the accuracy dropped to 63.13%, while in the reference study it fell to 49.97%. This suggests that the CNN model used in this work handled trial-wise validation slightly more robustly, although the overall performance remains suboptimal. Since the original paper did not provide the code for the trial-wise setup, it was not possible to directly compare how the data were split or how training was handled. These differences in implementation may explain the performance gap between the two versions.

In the case of the Transformer model, using the same parameters as those reported in the paper resulted in slightly lower performance under the slice-wise setting, with an accuracy of 71.44% compared to 74.99% in the reference study. To address this, a manual adjustment was made by increasing the batch size from 128 to 256, based on the idea that a larger batch size could produce more stable gradient estimates and improve generalization during training. This change led to an improved accuracy of 75.12%.

Under the trial-wise validation setting, the model achieved an accuracy of 63.91% with the original batch size of 128, which is very close to the 62.00% reported in the original paper.

To conclude this comparison with the reference study, it can be observed that all models, except Naive Bayes, achieved solid performance in the task of classifying trust, particularly when using the slice-wise validation approach. This supports the idea, also mentioned in the original study, that EEG signals change gradually over time, and models can take advantage of that to improve their performance.

As for the deep learning models, both CNN and Transformer achieved results comparable to those reported in the original paper. This supports the idea that spatial encoding of EEG data enables neural networks to extract meaningful patterns related to trust. However, in this work, some traditional machine learning models such as KNN and SVM ended up outperforming them. This doesn't necessarily indicate a limitation of deep learning itself. It is important to consider that these models typically require larger amounts of data to perform optimally, and the dataset used, 12 participants with 30 trials each, may have been too limited. Moreover, unlike the traditional classifiers, the CNN and Transformer models were not subjected to any hyperparameter tuning in this study. Therefore, while deep learning remains a promising approach for trust recognition, the current results highlight the importance of both dataset size and model optimization when comparing across methodologies.

4.3 Discussion and Interpretation of Results

Now that the results have been compared with those reported in the original study, the focus shifts to evaluating the performance achieved across the different experimental configurations explored in this work. This analysis aims to identify which approaches yielded the most consistent and effective results, and to better understand how different design choices, such as feature selection and modeling paradigm (within-subject vs. between-subject), affect model performance.

One important and somewhat unexpected observation is that adding more features led to a decline in classification accuracy for most models. For instance, in the KNN model under the slice-wise setting, accuracy dropped from 70.79% when using a single characteristic to 66.78% with multiple features. The same trend appeared in the SVM model, where accuracy fell from 77.34% to 72.34% under the same conditions. These results suggest that BDE alone captures the most relevant information for trust prediction in this context. The inclusion of additional descriptors may introduce redundancy or noise, increasing complexity without improving and sometimes even degrading performance.

However, this performance drop was not consistent across all models. In particular, tree-based models such as Random Forest and XGBoost were far less affected by the inclusion of extra features. For example, in the slice-wise configuration, XGBoost's accuracy decreased only slightly from 79.61% to 78.01%, while Random Forest saw a drop from 77.04% to 73.98%. This robustness is likely due to the inherent feature selection mechanisms these models apply during training, allowing them to ignore irrelevant or redundant inputs. In addition, XGBoost includes regularization techniques that help prevent overfitting, which may explain its stable performance even with more complex input spaces.

Building on this robustness to multiple features, feature importance scores were computed for both models under the multiple-characteristics setting. Figures 4.2 and 4.3 show the 20 most relevant features per subject for XGBoost and Random Forest, respectively, in the within-subject, slice-wise configuration. These visualizations clearly show that BDE consistently appears among the top features across all subjects, confirming its principal role in classifying trust.

Although both models highlight the importance of BDE, Random Forest relies on it even more heavily than XGBoost. This is likely due to differences in how feature relevance is computed. Random Forest evaluates importance based on how much each feature improves decision splits across many trees, which tends to emphasize consistently strong predictors like BDE. In contrast, XGBoost builds trees sequentially and applies regularization, encouraging the model to consider a broader range of features. As a result, it spreads importance more evenly across additional descriptors such as PSD, Hjorth parameters, Kurtosis, and Skewness. This reflects XGBoost's strength in capturing more complex interactions while remaining stable.

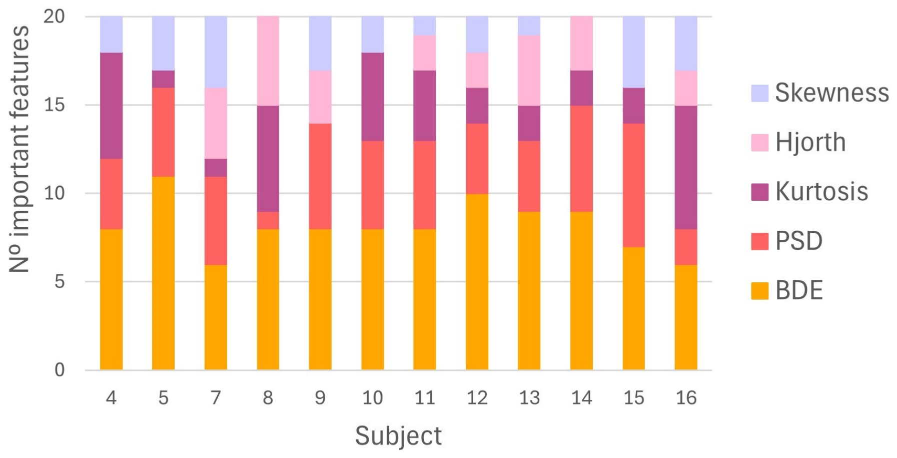

Figure 4.2: Distribution of the top 20 most important features per subject using the XG-Boost model. Results correspond to the within-subject, multiple characteristics, slice-wise approach and are categorized by feature type.

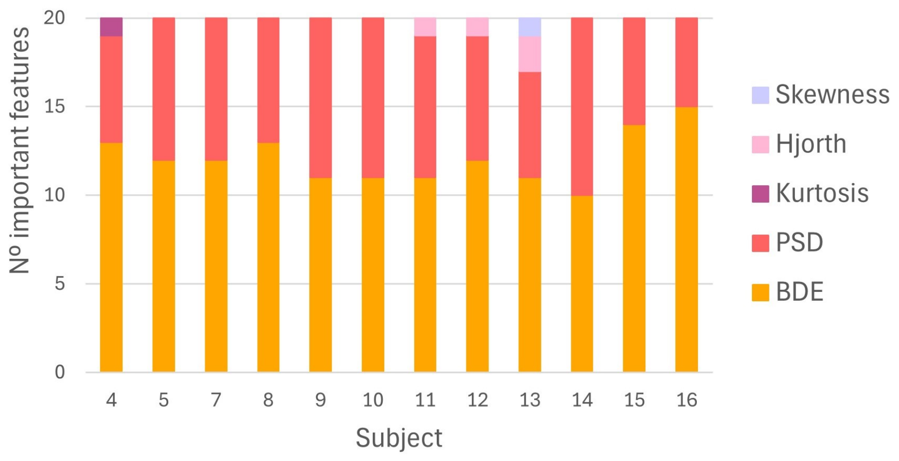

Figure 4.3: Distribution of the top 20 most important features per subject using the Random Forest model. Results correspond to the within-subject, multiple characteristics, slice-wise approach and are categorized by feature type.

Taking advantage of the feature importance values obtained from the Random Forest and XGBoost models, an exploratory topographic visualization was conducted to examine which EEG channels contributed the most to trust classification. Specifically, a topoplot was generated based on the XGBoost model trained under the within-subject, slice-wise, multiple-characteristics configuration, but using only the BDE features.

To construct this map, the feature importance values from the XGBoost model were first grouped by EEG channel for each subject, considering only the BDE features. Then, the importance scores were averaged across subjects to estimate how relevant each channel was, on average, for trust classification. While this procedure does not represent a rigorous neurophysiological activation map, it provides a rough approximation of which regions may carry the most informative signals for trust prediction, as identified by the model.

The resulting visualization is shown in Figure 4.4. The topographic feature-importance map based on BDE indicates that frontal and parietal regions contribute most prominently to trust prediction, with additional importance observed in occipital areas. Increased relevance in frontal channels is consistent with the role of the prefrontal cortex in trust-related decision-making and the evaluation of uncertainty or risk [60]. The parietal region, particularly the posterior parietal cortex, has been associated with attentional control and integration of sensory information, both essential in tasks requiring continuous interaction [61]. Occipital contributions may reflect the strong visual demands of the video-game-based experimental task [62].

While this topographic representation should be interpreted with caution and doesn't reflect a precise neural activation map, it offers a valuable starting point. It shows how model-driven interpretability tools can help us gain spatial insights and could guide future efforts to connect machine learning outputs with underlying neurophysiological processes.

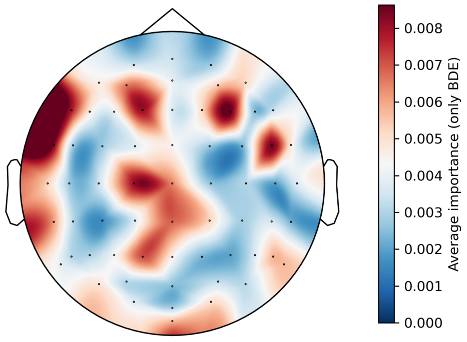

Figure 4.4: Topographic distribution of average feature importance across EEG channels using only Band Differential Entropy (BDE) features extracted from the XGBoost model. Results correspond to the within-subject, slice-wise, multiple-characteristics configuration.

Having examined the impact of using multiple features versus a single characteristic, and after previously discussing the differences between slice-wise and trial-wise validation, the final comparison focuses on the within-subject versus between-subject configurations. As expected, within-subject approaches consistently outperform between-subject models across all classifiers and feature setups.

For instance, in the KNN model using slice-wise validation and a single characteristic, accuracy drops from 70.79% in the within-subject setting to just 48.53% in the between-subject scenario. A similar trend is observed for XGBoost, where performance under the same conditions drops from 79.61% to 49.95%. This performance gap remains evident even when using multiple features. For example, with XGBoost under trial-wise validation, accuracy decreases from 64.03% (within-subject) to 49.86% (between-subject).

These results confirm that the models struggle to generalize trust-related patterns across different individuals, regardless of the number of features used. This highlights the persistent challenge of inter-subject variability in EEG-based trust classification. It is also important to note that these conclusions are based on data from only 12 participants. Given the considerable variability observed between individuals, overcoming this limitation would likely require a much larger and more diverse dataset.

To wrap up this comparative analysis, the results presented so far suggest that the most effective configuration for classifying trust is the within-subject setting using a single characteristic and slice-wise validation (Approach 1). This setup consistently delivered the highest performance across models.

The observed performance differences also highlight a key challenge in EEG-based modeling of trust: the significant variability between subjects. Despite the use of multiple features or advanced classifiers, between-subject models struggled to generalize. This suggests that achieving reliable generalization across individuals likely requires a much larger and more diverse dataset than the one used in this study, which included only 12 participants.

Figure 4.5 summarizes the performance of all models under the best-performing configuration, within-subject, slice-wise validation using a single feature. Overall, most models performed well in this setting, with the exception of Naive Bayes, which clearly underperformed compared to the rest. This may be due to its strong assumptions, such as feature independence, which are unlikely to hold in a complex signal like EEG.

Among the remaining models, XGBoost achieved the highest accuracy on average, making it the top performer in this study. However, Random Forest and SVM also delivered strong results, with slightly more consistent performance across participants. This kind of robustness is particularly important in trust modeling, where individual differences can greatly influence outcomes.

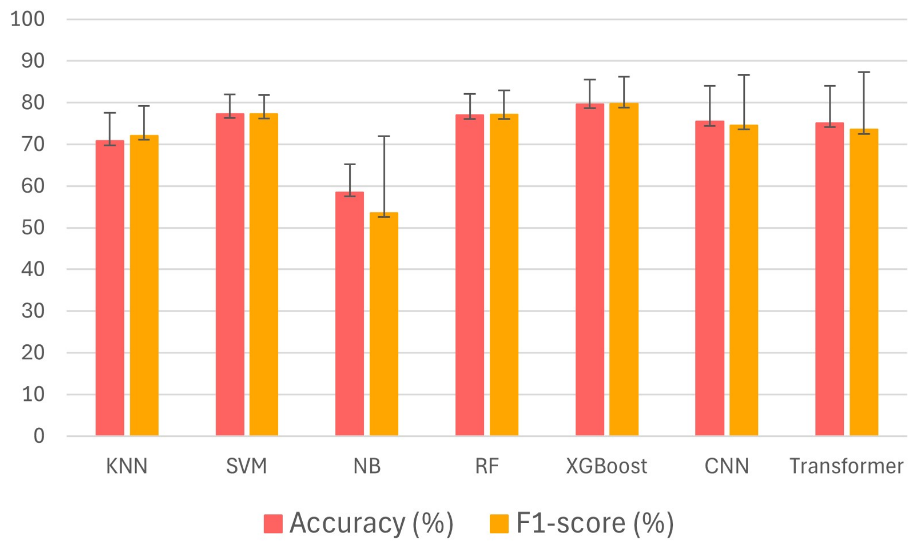

Figure 4.5: Comparison of model performance under the within-subject, slice-wise, single-feature configuration.

Conclusions and Future Works

This thesis aimed to explore whether human trust in robots can be inferred from brain activity, using EEG signals and AI models. The motivation lies in the growing importance of trust in human–robot collaboration, where the success of an interaction often depends on how users perceive and respond to the robot's behavior. Traditional methods for measuring trust are typically based on questionnaires and fail to capture rapid shifts that may occur during real-time interactions. EEG, by contrast, offers a continuous and objective window into the user's internal state, making it a promising tool for decoding trust-related signals in a non-invasive manner.

This work builds on the methodology proposed in a previous study [3], which evaluated trust recognition using several machine learning and deep learning models, such as KNN, SVM, Naive Bayes, CNN, and Transformer, applied to EEG data under two experimental settings: slice-wise and trial-wise. In this thesis, the same experimental design was reproduced and extended by including additional models such as Random Forest and XGBoost, and by exploring a broader range of model configurations. These included comparisons between single-feature and multi-feature inputs, as well as analyses under both within-subject and between-subject settings. This expanded approach allowed for a more complete understanding of how different combinations of features and models behave under various experimental conditions.

The results confirmed that trust can be classified from EEG signals with reasonably high accuracy, particularly when models are trained and tested on the same individual. In this within-subject setting, classical models achieved strong performance: KNN reached an average accuracy of 75.28%, SVM 72.92%, Random Forest 74.31%, and XGBoost 72.64%. Among the input features, BDE consistently proved to be the most informative across models. In fact, adding more features beyond BDE often resulted in lower accuracy, suggesting that some models struggled to handle the additional noise or redundancy. Ensemble methods like Random Forest and XGBoost, however, were more robust to multi-feature inputs, likely due to their ability to internally prioritize relevant information during training.

In addition to classical models, deep learning architectures were evaluated using the same experimental settings. In the within-subject, slice-wise configuration, the CNN achieved an average accuracy of 75.46%, while the Vision Transformer reached 73.82%. These results highlight the potential of deep models to capture meaningful spatial patterns in EEG data, especially when the input is structured to reflect the layout of the electrodes. Despite their competitive performance, these models did not outperform the best classical classifiers, likely due to limitations such as the absence of fine-tuned hyperparameters and the relatively low number of training subjects. Overall, these findings suggest that deep learning methods could offer greater benefits in future work, provided that larger and more diverse datasets are available.

Despite these promising results, several limitations must be acknowledged. Most notably, all models showed a clear performance drop under trial-wise and between-subject evaluations. When entire trials were held out from training, or when the model was tested on previously unseen individuals, accuracy declined significantly, often falling below 60%. This highlights the challenge of generalization in EEG-based trust modeling, where high inter-subject variability and task-specific patterns can limit a model's ability to perform in real-world applications. It also points to the need for more robust evaluation protocols, since slice-wise cross-validation may overestimate model performance by introducing data leakage across segments from the same trial that is not immediately obvious.

Beyond performance metrics, the thesis also explored the neurophysiological relevance of the features selected by the models. Feature importance analysis indicated that frontal and parietal regions, especially those associated with decision-making, attention, and social cognition, were consistently among the most informative areas. While this topographic insight should be interpreted cautiously, it offers a valuable starting point for connecting machine learning outcomes with known cognitive functions involved in trust formation.

Looking ahead, future research should focus on addressing the generalizability problem. A good next step would be to collect larger and more diverse datasets that better capture the variability across individuals, contexts, and tasks. This would enable models to learn more robust representations of trust and facilitate the use of deep learning techniques that rely on big data. Transfer learning strategies could also help personalize models to new users with less need for individual retraining. At the same time, experiments should be expanded to include different types of interactions and robot behaviors, to test whether EEG-based trust detection remains reliable across broader application domains.

Finally, improving the interpretability of the models and linking their outputs more clearly to brain activity remains an important goal. Future work could include source localization techniques or more focused analyses to check whether the brain areas highlighted by the models are truly related to trust. This would help make the findings more scientifically grounded, and also improve the transparency and reliability of AI systems that aim to understand human mental states.

In conclusion, this thesis demonstrates the feasibility of using EEG and AI models to classify trust in human–robot collaboration settings. While high accuracy was achieved under controlled conditions, several challenges remain in terms of generalization, interpretability, and practical application. Even so, the results offer a strong starting point for future research on trust recognition and contribute to the growing effort to better understand how mental states can be inferred from neural data.

Ethical, Economic, Societal and Environmental Aspects

6.1 Introduction

This project investigated how human trust could be detected using EEG signals and AI models, in the context of a cooperative task between a human and a robot. Although the main objective was technical, the project also raises important questions beyond engineering. These include ethical concerns about data privacy, potential societal impact, economic relevance, and environmental considerations.

As trust becomes increasingly important in human-AI collaboration, especially in demanding environments that involve high cognitive load, complex decision-making, or safety-critical operations, it is essential to understand not only how to model it, but also the broader implications of doing so. This report outlines the main impacts related to the project and presents a detailed ethical analysis based on internationally recognized frameworks, such as the United Nations Sustainable Development Goals (SDGs) [63] and current discussions in neuroethics.

6.2 Description of Relevant Impacts Related to the Project

The following sections look at the project's wider impact outside its main focus. This helps frame its relevance in real-world situations and serves as an introduction to the ethical analysis that comes next.

6.2.1 Ethical Considerations

From an ethical perspective, this project used EEG signals collected from real human participants, provided in an anonymised and publicly available dataset. Although no new data were gathered, this type of neurophysiological data is inherently sensitive, as it can reflect aspects of a person's cognitive or emotional state. Working with such information, even in secondary analyses, requires particular attention to privacy, informed consent, and responsible usage.

While the study did not involve the creation of an operational system, it investigated whether AI models could be used to distinguish different levels of trust based on brain activity recordings. In doing so, it raises initial ethical questions about how choices in data selection, model training, and generalisation might affect fairness and reliability in future applications. Transparency and inclusiveness at the research stage could help guide the development of more responsible and socially aware systems in the long term.

6.2.2 Economic Implications

Consistent with its non-commercial nature, the project was conducted using public datasets, academic computing resources, and open-source software, which kept its direct costs to a minimum.

However, it forms part of a broader research line focused on trust recognition in human-machine interaction, a topic that is gaining relevance in demanding settings such as aviation, where understanding a pilot's mental state could help improve decision-making, safety, or training protocols. In this context, the capacity to assess trust through EEG data could offer added value in professional applications, supporting the development of more adaptive systems that respond to cognitive states in real time.

As research in this area continues to grow, these tools could eventually help optimize resources, reduce the need for constant human supervision, and improve overall system efficiency, factors that may have a direct economic impact in sectors where trust and performance are closely linked.

6.2.3 Social Impact

Though not yet ready for practical deployment, this research took a step toward systems that understand and adapt to human psychological states. Exploring whether neural data could reflect trust levels may eventually support the design of more intuitive and responsive technologies, potentially improving collaboration between humans and machines in a variety of contexts.

These possibilities are especially relevant in areas like assistive robotics, elderly care, or therapeutic environments, where users may rely on technology not only for practical help, but also for a sense of safety and emotional support. In the future, these insights could also help inform the design of systems for more demanding situations, where trust plays a key role in safety and decision-making. While these applications were not the focus of the current study, they offer a valuable direction for future research.

6.2.4 Environmental Impact

Finally, from an environmental point of view, this project has a very limited footprint. It made use of existing datasets and university computing resources, without requiring any new data collection or physical materials.

The main environmental consideration relates to the computational cost of training certain artificial intelligence models. In particular, some of the most resource-intensive experiments involved SVMs and deep learning architectures such as CNNs and ViTs. While the energy consumption involved in academic research is relatively low, it is still important to be aware of its cumulative impact, which becomes more relevant when these methods are used beyond the research context.

6.2.5 Relevant Stakeholders

The points discussed above help identify the people and organizations who could be impacted by future developments in trust-aware technology, as well as the key aspects they may need to consider. These stakeholders include academic and research institutions, developers of intelligent and interactive systems, professionals working in domains such as healthcare, aviation, or assistive robotics, and the end-users who may ultimately interact with these technologies.

For research institutions and developers, ethical considerations around data privacy, fairness, and transparency are particularly relevant, as they guide the responsible design and use of neurophysiological data. Economic and environmental aspects, such as the computational cost and scalability of machine learning models, also influence the feasibility of future applications.

Professionals in critical fields like healthcare or aviation may depend on trust-aware systems to improve decision-making, training, or user support. For them, the accuracy and reliability of such technologies are crucial.

Meanwhile, end-users, especially those in vulnerable or assistive contexts, may be directly affected by how trustworthy and understandable these systems feel in practice. Their acceptance often depends not only on performance, but also on perceived fairness, privacy, and emotional safety.

Addressing these diverse concerns from the early stages of development can help ensure that future trust-aware systems are not only technically robust, but also ethically responsible, socially acceptable, economically viable, and sustainable.

6.3 Detailed Analysis of One of the Principal Impacts: Ethical Considerations

Among the potential impacts explored in this project, the ethical dimension is considered the most directly relevant at this point in development. While the social implications of trust-aware systems are expected to grow in the future, especially as such technologies become more embedded in everyday life, their societal impact remains limited for now. By contrast, the ethical considerations surrounding the use of neural data are already critical from the outset. This is mainly due to the nature of the data used: EEG signals recorded from real participants. Although the project does not involve new data collection or direct interaction with users, the analysis of brain-related information carries significant ethical implications that cannot be overlooked.

Such signals can reflect delicate aspects of a person's mental state, including attention or emotional patterns. Even when anonymised and made publicly available, this kind of data must be handled with care. The project follows a responsible approach by acknowledging these concerns and seeks to remain consistent with principles such as transparency, privacy, and fairness. To better understand the ethical relevance of this work, the following analysis considers how these concerns relate to internationally recognized frameworks, including the SDGs [63] and recent neuroethics initiatives.

One of the most directly related frameworks is SDG 16, which encourages the development of institutions that uphold justice, accountability, and the protection of human rights. In the context of this project, this includes respecting mental privacy and the ethical use of neurophysiological data. Even in academic settings, neural signals, though anonymised, can carry information about an individual's inner states. For this reason, documenting data usage clearly and ensuring ethical awareness during model development contribute to the broader goals of responsible and transparent research.

SDG 10, which focuses on reducing inequalities, is also relevant. When working with datasets that are small or not fully representative, there is a risk that resulting systems may not generalize well to all populations. While this project does not control the composition of the original data, it does attempt to address this issue by evaluating models both within and across subjects. This helps highlight the importance of inclusiveness in future data collection efforts.

Ethical considerations become even more important when imagining potential future applications in areas like aviation, healthcare, or assistive robotics, where trust and decision-making are closely linked. In these sensitive contexts, even small classification errors could lead to a loss of user confidence or safety risks. This perspective aligns with SDG 3, which advocates for well-being and safety in human-centered technologies.

These questions are also being explored in current discussions around neuroethics, where several organizations and initiatives have proposed frameworks to guide the responsible development of technologies that interact with the brain. While the project does not propose specific ethical safeguards, it contributes to this broader debate by reflecting values such as transparency, fairness, and psychological integrity. This includes alignment with ideas like neurorights [64], which are principles aimed at protecting mental privacy and cognitive freedom, as well as technical standards such as the IEEE 7000 series [65], which promote ethical system design from the outset.

6.4 Conclusions

While the project's focus is mostly technical, it also raises important questions about how such technologies might affect people and society. These include ethical responsibilities when working with sensitive neurophysiological data, the potential social and economic impact of trust-aware systems, and the environmental footprint of AI-based solutions.

Of the four dimensions considered, the ethical aspect stands out as the most immediate and significant. This is due to the nature of the data involved and the importance of respecting mental privacy, fairness, and inclusiveness from the very beginning. The project aligns with key principles found in the SDGs, particularly those promoting well-being (SDG 3), equality (SDG 10), and human rights (SDG 16), as well as with emerging neuroethical frameworks such as the concept of neurorights.

Looking ahead, trust-aware technologies are likely to become more common in areas where humans and machines must work together closely. Anticipating their wider implications now can help shape systems that not only perform well, but are also aligned with societal values. By addressing these issues from the start, this project contributes to a more thoughtful and responsible development of AI systems that engage with the human mind.

Financial Budget

Although this Master's Thesis did not involve direct economic expenses, a theoretical budget has been estimated based on the resources used throughout the project.

The main cost corresponds to the student's personal time, calculated according to the number of hours typically assigned to an 18 ECTS credit project.

In addition, material resources such as the student's personal computer have been considered, along with the use of institutional computing infrastructure, specifically the university's high-performance computing (HPC) cluster.

The following table 7.1 presents a detailed breakdown of the estimated project budget.

PERSONNEL COSTS (direct costs)		Hours	Hourly cost	Total
		450	15 €	**6.750,00 €**
MATERIAL RESOURCES COSTS (direct costs, DC)	**Purchase price**	**Months of use**	**Depreciation (in years)**	**Total**
Personal computer, including software	1.500,00 €	6	5	**150,00 €**
Use of university HPC cluster	0,00 €	6	-	**0,00 €**
TOTAL COSTS OF MATERIAL RESOURCES				**150,00 €**
GENERAL COSTS (indirect costs) 15%	on DC			**1.035,00 €**
INDUSTRIAL BENEFIT 6%	on DC+IC			**414,00 €**
SUBTOTAL BUDGET				**8.349,00 €**
VAT			21%	**1.753,29 €**
TOTAL BUDGET				**10.102,29 €**

Table 7.1: Estimated budget for the project

Bibliography

[1] IndoML, "Student notes: Convolutional neural networks (cnn) introduction." `https : / / indoml . com / 2018 / 03 / 07 / student-notes-convolutional-neural-networks-cnn-introduction/`, 2018. Accessed: 2025-06-12.

[2] MRI Questions, "Convolutional neural network (cnn)." `https://mriquestions.com/convolutional-network.html`, n.d. Accessed: 2025-06-25.

[3] C. Xu, C. Zhang, Y. Zhou, Z. Wang, P. Lu, and B. He, "Trust recognition in human-robot cooperation using eeg." `https://ieeexplore.ieee.org/document/10610156`, 2024. Accessed: 2025-06-06.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representations (ICLR)*, 2021.

[5] Y. Tamura, "Multi-head attention mechanism: q̈ueries,̈ k̈eys,̈ and v̈alues.̈" `https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/`, 2021. Accessed: 2025-06-02.

[6] International Federation of Robotics, "Collaborative robots - how robots work alongside humans," 2024. Accessed: 2025-06-28.

[7] G. Campagna and M. Rehm, "A systematic review of trust assessments in human–robot interaction," *ACM Transactions on Human–Robot Interaction*, vol. 14, no. 2, pp. 1–25, 2025.

[8] S. Makeig, A. Delorme, T.-P. Jung, and T. Sejnowski, "Dynamic brain sources of visual evoked responses," *Science*, vol. 295, no. 5556, pp. 690–694, 2004.

[9] C. Xu, C. Zhang, Y. Zhou, Z. Wang, P. Lu, and B. He, "Eeg-vision dataset and code repository." `https://github.com/cyxu9/EEGTrust_ViT`, 2024. Accessed: June 2025.

[10] E. Niedermeyer and F. L. da Silva, *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams & Wilkins, 5th ed., 2004.

[11] J. Yedukondalu, K. Sunkara, V. Radhika, *et al.*, "Cognitive load detection through eeg lead wise feature optimization and ensemble classification," *Scientific Reports*, vol. 15, p. 842, 2025.

[12] A. Natalizio, S. Sieghartsleitner, L. Schreiner, M. Walchshofer, A. Esposito, J. Scharinger, H. Pretl, P. Arpaia, M. Parvis, and J. Solé-Casals, "Real-time estimation of eeg-based engagement in different tasks," *Journal of Neural Engineering*, vol. 21, no. 1, p. 016014, 2024.

[13] G. Campagna and M. Rehm, "Trust assessment with eeg signals in social human-robot interaction," in *Social Robotics. ICSR 2023. Lecture Notes in Computer Science* (A. A. Ali *et al.*, eds.), vol. 14453, pp. 33–42, Springer, Singapore, 2024.

[14] E. J. de Visser, P. J. Beatty, J. R. Estepp, S. Kohn, A. Abubshait, J. R. Fedota, and C. G. McDonald, "Learning from the slips of others: Neural correlates of trust in automated agents," *Frontiers in Human Neuroscience*, vol. 12, p. 309, 2018.

[15] T. O. Zander and C. Kothe, "Towards passive brain–computer interfaces: applying brain–computer interface technology to human–machine systems in general," *Journal of Neural Engineering*, vol. 8, no. 2, p. 025005, 2011.

[16] I. B. Ajenaghughrure, S. C. Sousa, I. J. Kosunen, and D. Lamas, "Predictive model to assess user trust: a psycho-physiological approach," in *Proceedings of the 10th Indian Conference on Human-Computer Interaction*, IndiaHCI '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[17] E. Loizaga, L. Bastida, S. Sillaurren, A. Moya, and N. Toledo, "Modelling and measuring trust in human–robot collaboration," *Applied Sciences*, vol. 14, no. 5, 2024.

[18] H. Li, M. Liang, K. Niu, and Y. Zhang, "A human-machine trust evaluation method for high-speed train drivers based on multi-modal physiological information," *International Journal of Human–Computer Interaction*, vol. 41, no. 4, pp. 2659–2676, 2025.

[19] H. Chauhan, Y. Jang, and I. Jeong, "Predicting human trust in human-robot collaborations using machine learning and psychophysiological responses," *Advanced Engineering Informatics*, vol. 62, p. 102720, 2024.

[20] M. Diab and Y. Demiris, "A framework for trust-related knowledge transfer in human–robot interaction," *Autonomous Agents and Multi-Agent Systems*, vol. 38, no. 24, 2024.

[21] G. Campagna and M. Rehm, "Trust assessment with eeg signals in social human-robot interaction," *Lecture Notes in Computer Science*, vol. 1, pp. 33–42, 2023.

[22] G. Campagna and M. Rehm, "Eeg trust dataset for social human-robot interaction." `https://zenodo.org/record/12754928`, 2024. Accessed: 2025-06-29.

[23] A. Xu and G. Dudek, "Optimo: Online probabilistic trust inference model for asymmetric human-robot collaborations," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 221–228, 2015.

[24] M. X. Cohen, *Analyzing Neural Time Series Data.* Cambridge, Massachusetts: MIT Press, 2014.

[25] A. Delorme and S. Makeig, "Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis," *Journal of neuroscience methods*, vol. 134, no. 1, pp. 9–21, 2004.

[26] D. L. Schomer and F. H. Lopes da Silva, *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields.* Oxford University Press, 11 2017.

[27] Sapien Labs, "Common average vs. infinity reference in eeg." `https://sapienlabs.org/common-average-vs-infinity-reference-in-eeg/`, 2020. Accessed: 2025-05-20.

[28] S. Tremblay, K. M. Sharika, and M. L. Platt, "Social decision-making and the brain: A comparative perspective," *Trends in Cognitive Sciences*, vol. 21, no. 4, pp. 265–276, 2017.

[29] E. Başar, C. Başar-Eroğlu, S. Karakaş, and M. Schürmann, "Brain oscillations in perception and memory," *International Journal of Psychophysiology*, vol. 35, no. 2-3, pp. 95–124, 2000.

[30] M. X. Cohen, *Analyzing Neural Time Series Data: Theory and Practice.* Cambridge, Massachusetts: MIT Press, 2014.

[31] A. K. Engel and P. Fries, "Beta-band oscillations—signalling the status quo?," *Current Opinion in Neurobiology*, vol. 20, no. 2, pp. 156–165, 2010.

[32] A. Widmann, E. Schröger, and B. Maess, "Digital filter design for electrophysiological data – a practical approach," *Journal of Neuroscience Methods*, vol. 250, pp. 34–46, 2015. Cutting-edge EEG Methods.

[33] E. Wiki, "Fir filtering - faq." `https://eeglab.org/others/Firfilt_FAQ.html`, 2023. Accessed: 2025-05-20.

[34] S. J. Luck, *An Introduction to the Event-Related Potential Technique.* Cambridge, MA: MIT Press, 2 ed., 2014.

[35] S. Leske and S. S. Dalal, "Reducing power line noise in eeg and meg data via spectrum interpolation," *NeuroImage*, vol. 189, pp. 763–776, 2019.

[36] S. Makeig, A. J. Bell, T. P. Jung, T.-P. Jung, and T. J. Sejnowski, "Independent component analysis of electroencephalographic data," *Advances in Neural Information Processing Systems*, vol. 8, pp. 145–151, 1996.

[37] L. Pion-Tonachini, K. Kreutz-Delgado, and S. Makeig, "Iclabel: An automated electroencephalographic independent component classifier, dataset, and website," *NeuroImage*, vol. 198, pp. 181–197, 2019.

[38] T. Wang, X. Huang, Z. Xiao, and Y. Tai, "Eeg emotion recognition based on differential entropy feature matrix through 2d-cnn-lstm network," *EURASIP Journal on Advances in Signal Processing*, vol. 2024, no. 49, 2024. Received: 29 September 2023, Accepted: 02 April 2024, Published: 08 April 2024.

[39] Y. Lu, "Exploring differential entropy and multifractal cumulants for eeg-based mental workload recognition," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 5, pp. 1–6, 2024. Accessed: 2025-06-30.

[40] K. Akash, W.-L. Hu, N. Jain, and T. Reid, "A classification model for sensing human trust in machines using eeg and gsr," *ACM Trans. Interact. Intell. Syst.*, vol. 8, Nov. 2018.

[41] T. Contributors, "Torcheeg documentation: Transforms module." `https://torcheeg.readthedocs.io/en/latest/torcheeg.transforms.html`, 2024. Accessed: 2025-06-06.

[42] Y. Shahsavar and A. Choudhury, "User trust in a depression screening app when outcomes are labelled as either ai-generated or doctor-generated: a pilot study," in *Human Factors in Design, Engineering, and Computing* (T. Ahram and W. Karwowski, eds.), vol. 159 of *AHFE Open Access*, USA: AHFE International, 2024.

[43] J. R. Tatz, A. Mather, and J. R. Wessel, "-bursts over frontal cortex track the surprise of unexpected events in auditory, visual, and tactile modalities," *Journal of Cognitive Neuroscience*, vol. 35, no. 3, pp. 485–508, 2023.

[44] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 278, no. 6, pp. H2039–H2049, 2000.

[45] B. Hjorth, "Eeg analysis based on time domain properties," *Electroencephalography and Clinical Neurophysiology*, vol. 29, no. 3, pp. 306–310, 1970.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[47] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat, "Enhancing k-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications." `https://doi.org/10.1186/s40537-024-00973-y`, 2024. Accessed: 2025-06-06.

[48] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[49] T. Mitchell, “Machine learning.” https://www.cs.cmu.edu/~tom/mlbook.html, 1997. Accessed: 2025-06-06.

[50] M.-P. Hosseini, A. Hosseini, and K. Ahi, “A review on machine learning for eeg signal processing in bioengineering.” https://doi.org/10.1109/RBME.2020.2969915, 2021. Accessed: 2025-06-06.

[51] L. Breiman, “Random forests.” https://doi.org/10.1023/A:1010933404324, 2001. Accessed: 2025-06-06.

[52] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system.” https://doi.org/10.1145/2939672.2939785, 2016. Accessed: 2025-06-06.

[53] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[55] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” *arXiv preprint arXiv:1603.07285*, 2016.

[56] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” pp. 807–814, 2010.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[58] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[60] X. Li, Y. Pan, Y. Zhang, F. Wang, and W. Xu, “Eeg-based trust prediction in human–robot interaction using convolutional neural networks,” *Cognitive Neurodynamics*, vol. 15, no. 2, pp. 359–372, 2021.

[61] J. K. Rilling and A. G. Sanfey, “The neural correlates of theory of mind within interpersonal interactions,” *NeuroImage*, vol. 40, no. 2, pp. 377–388, 2008.

[62] S. Ba and R. Niyazov, “Eeg brain dynamics following visual perception and video game playing: A review,” *Computational Intelligence and Neuroscience*, 2009.

[63] Naciones Unidas, “Objetivos de desarrollo sostenible.” https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/, 2015. Accessed June 2025.

[64] OECD, “Recommendation on responsible innovation in neurotechnology,” 2023. Accessed June 2025.

[65] IEEE Standards Association, “Ieee 7000-2021 - ieee standard model process for addressing ethical concerns during system design.” `https://standards.ieee.org/ieee/7000/10201/`, 2021. Accessed June 2025.

Annexes

A Hyperparameter Search

Subject	**KNN Metric**	**N Neighbors**	**Weights**
sub04	euclidean	9	uniform
sub05	manhattan	9	uniform
sub07	manhattan	9	uniform
sub08	euclidean	9	uniform
sub09	manhattan	7	uniform
sub10	manhattan	3	uniform
sub11	manhattan	9	uniform
sub12	manhattan	7	uniform
sub13	euclidean	7	uniform
sub14	euclidean	7	distance
sub15	manhattan	5	uniform
sub16	manhattan	5	uniform
Most Common	**manhattan**	**9**	**uniform**

Table 7.2: Best hyperparameters per subject using KNN on within-subject, one characteristic, slice-wise approach.

Subject	KNN Metric	N Neighbors	Weights
sub04	manhattan	5	uniform
sub05	manhattan	7	uniform
sub07	manhattan	9	uniform
sub08	euclidean	7	uniform
sub09	manhattan	5	uniform
sub10	manhattan	7	uniform
sub11	manhattan	9	uniform
sub12	manhattan	7	uniform
sub13	manhattan	9	distance
sub14	manhattan	5	uniform
sub15	manhattan	9	uniform
sub16	manhattan	7	uniform
Most Common	**manhattan**	**7**	**uniform**

Table 7.3: Best hyperparameters per subject using KNN on within-subject, one characteristic, trial-wise approach.

Subject	KNN Metric	N Neighbors	Weights
sub04	manhattan	7	uniform
sub05	manhattan	9	uniform
sub07	manhattan	7	uniform
sub08	manhattan	9	uniform
sub09	manhattan	9	uniform
sub10	manhattan	5	uniform
sub11	manhattan	9	uniform
sub12	manhattan	9	uniform
sub13	manhattan	9	distance
sub14	manhattan	9	uniform
sub15	manhattan	9	uniform
sub16	manhattan	7	uniform
Most Common	**manhattan**	**9**	**uniform**

Table 7.4: Best hyperparameters per subject using KNN on within-subject, multiple characteristics, slice-wise approach.

Subject	KNN Metric	N Neighbors	Weights
sub04	manhattan	5	uniform
sub05	manhattan	9	uniform
sub07	manhattan	7	uniform
sub08	manhattan	9	uniform
sub09	manhattan	9	uniform
sub10	manhattan	9	uniform
sub11	manhattan	9	uniform
sub12	manhattan	9	uniform
sub13	manhattan	7	uniform
sub14	manhattan	9	uniform
sub15	manhattan	9	uniform
sub16	manhattan	9	uniform
Most Common	**manhattan**	**9**	**uniform**

Table 7.5: Best hyperparameters per subject using KNN on within-subject, multiple characteristics, trial-wise approach.

Test Subject	KNN Metric	N Neighbors	Weights
sub04	euclidean	3	uniform
sub05	manhattan	3	uniform
sub07	manhattan	5	uniform
sub08	euclidean	3	uniform
sub09	euclidean	7	uniform
sub10	manhattan	5	uniform
sub11	euclidean	3	uniform
sub12	manhattan	5	uniform
sub13	euclidean	3	uniform
sub14	manhattan	3	uniform
sub15	euclidean	3	uniform
sub16	euclidean	3	uniform
Most Common	**euclidean**	**3**	**uniform**

Table 7.6: Best hyperparameters per test subject using KNN on between-subject, one characteristic.

Test Subject	KNN Metric	N Neighbors	Weights
sub04	euclidean	5	uniform
sub05	euclidean	3	uniform
sub07	euclidean	3	uniform
sub08	euclidean	7	uniform
sub09	euclidean	7	uniform
sub10	euclidean	5	distance
sub11	euclidean	7	uniform
sub12	euclidean	5	uniform
sub13	euclidean	5	uniform
sub14	euclidean	7	uniform
sub15	euclidean	7	uniform
sub16	euclidean	5	uniform
Most Common	**euclidean**	**5/7 (tie)**	**uniform**

Table 7.7: Best hyperparameters per test subject using KNN on between-subject, multiple characteristics.

Subject	C	Kernel	Gamma
sub04	10	rbf	scale
sub05	10	rbf	scale
sub07	10	rbf	scale
sub08	0.1	linear	scale
sub09	10	rbf	scale
sub10	10	rbf	scale
sub11	10	rbf	scale
sub12	100	rbf	scale
sub13	10	rbf	scale
sub14	1	rbf	scale
sub15	100	rbf	scale
sub16	100	rbf	scale
Most Common	**10**	**rbf**	**scale**

Table 7.8: Best hyperparameters per subject using SVM on within-subject, one characteristic, slice-wise approach.

Subject	**C**	**Kernel**	**Gamma**
sub04	10	rbf	scale
sub05	10	rbf	scale
sub07	10	rbf	scale
sub08	1	rbf	scale
sub09	10	rbf	scale
sub10	10	rbf	scale
sub11	10	rbf	scale
sub12	10	rbf	scale
sub13	10	rbf	scale
sub14	10	rbf	scale
sub15	10	rbf	scale
sub16	100	rbf	scale
Most Common	**10**	**rbf**	**scale**

Table 7.9: Best hyperparameters per subject using SVM on within-subject, one characteristic, trial-wise approach.

Subject	**C**	**Kernel**	**Gamma**
sub04	10	rbf	auto
sub05	10	rbf	scale
sub07	10	rbf	auto
sub08	10	rbf	auto
sub09	10	rbf	auto
sub10	10	rbf	scale
sub11	10	rbf	scale
sub12	0.1	linear	scale
sub13	10	rbf	scale
sub14	1	rbf	scale
sub15	10	rbf	auto
sub16	10	rbf	scale
Most Common	**10**	**rbf**	**scale**

Table 7.10: Best hyperparameters per subject using SVM on within-subject, multiple characteristics, slice-wise approach.

Subject	C	Kernel	Gamma
sub04	10	rbf	auto
sub05	10	rbf	auto
sub07	10	rbf	auto
sub08	10	rbf	auto
sub09	10	rbf	auto
sub10	10	rbf	auto
sub11	10	rbf	scale
sub12	10	rbf	scale
sub13	10	rbf	auto
sub14	10	rbf	auto
sub15	10	rbf	auto
sub16	10	rbf	scale
Most Common	**10**	**rbf**	**auto**

Table 7.11: Best hyperparameters per subject using SVM on within-subject, multiple characteristics, trial-wise approach.

Subject	Var Smoothing
sub04	1e-09
sub05	1e-09
sub07	1e-09
sub08	1e-09
sub09	1e-09
sub10	1e-09
sub11	1e-09
sub12	1e-09
sub13	1e-09
sub14	1e-09
sub15	1e-09
sub16	1e-09
Most Common	**1e-09**

Table 7.12: Best hyperparameters per subject using NB on within-subject, one characteristic, slice-wise approach.

Subject	Var Smoothing
sub04	1e-09
sub05	1e-09
sub07	1e-09
sub08	1e-09
sub09	1e-09
sub10	1e-09
sub11	1e-09
sub12	1e-09
sub13	1e-09
sub14	1e-09
sub15	1e-09
sub16	1e-09
Most Common	**1e-09**

Table 7.13: Best hyperparameters per subject using NB on within-subject, one characteristic, trial-wise approach.

Subject	Var Smoothing
sub04	1e-09
sub05	1e-05
sub07	1e-09
sub08	1e-09
sub09	1e-06
sub10	1e-09
sub11	1e-09
sub12	1e-09
sub13	1e-09
sub14	1e-09
sub15	1e-09
sub16	1e-09
Most Common	**1e-09**

Table 7.14: Best hyperparameters per subject using NB on within-subject, multiple characteristics, slice-wise approach.

Subject	Var Smoothing
sub04	1e-09
sub05	1e-05
sub07	1e-09
sub08	1e-09
sub09	1e-05
sub10	1e-09
sub11	1e-09
sub12	1e-09
sub13	1e-09
sub14	1e-09
sub15	1e-09
sub16	1e-09
Most Common	**1e-09**

Table 7.15: Best hyperparameters per subject using NB on within-subject, multiple characteristics, trial-wise approach.

Subject	Var Smoothing
sub04	1e-09
sub05	1e-05
sub07	1e-09
sub08	1e-09
sub09	1e-05
sub10	1e-09
sub11	1e-09
sub12	1e-09
sub13	1e-09
sub14	1e-09
sub15	1e-09
sub16	1e-09
Most Common	**1e-09**

Table 7.16: Best hyperparameters per test subject using NB on between-subject, one characteristics.

Subject	Var Smoothing
sub04	1e-05
sub05	1e-09
sub07	1e-06
sub08	1e-09
sub09	1e-09
sub10	1e-05
sub11	1e-05
sub12	1e-09
sub13	1e-09
sub14	1e-09
sub15	1e-09
sub16	1e-09
Most Common	**1e-09**

Table 7.17: Best hyperparameters per test subject using NB on between-subject, multiple characteristics.

Subject	Max Depth	N Estimators
sub04	20	200
sub05	10	100
sub07	20	200
sub08	10	200
sub09	–	200
sub10	20	200
sub11	20	200
sub12	–	200
sub13	10	200
sub14	10	200
sub15	10	100
sub16	10	200
Most Common	**10**	**200**

Table 7.18: Best hyperparameters per subject using RF on within-subject, one characteristic, slice-wise approach.

Subject	**Max Depth**	**N Estimators**
sub04	20	200
sub05	20	200
sub07	20	200
sub08	10	200
sub09	20	200
sub10	–	200
sub11	10	100
sub12	–	200
sub13	20	200
sub14	10	100
sub15	–	200
sub16	20	200
Most Common	**20**	**200**

Table 7.19: Best hyperparameters per subject using RF on within-subject, one characteristic, trial-wise approach.

Subject	**Max Depth**	**N Estimators**
sub04	–	200
sub05	10	200
sub07	10	200
sub08	–	200
sub09	–	100
sub10	20	200
sub11	–	200
sub12	–	200
sub13	10	200
sub14	–	200
sub15	–	200
sub16	10	200
Most Common	**10**	**200**

Table 7.20: Best hyperparameters per subject using RF on within-subject, multiple characteristics, slice-wise approach.

Subject	Max Depth	N Estimators
sub04	20	100
sub05	20	100
sub07	20	200
sub08	20	200
sub09	10	200
sub10	10	100
sub11	10	200
sub12	–	100
sub13	–	200
sub14	10	100
sub15	20	200
sub16	–	200
Most Common	**20**	**200**

Table 7.21: Best hyperparameters per subject using RF on within-subject, multiple characteristics, trial-wise approach.

Test Subject	Max Depth	N Estimators	Min Samples Split
sub04	10	100	5
sub05	–	200	2
sub07	–	200	2
sub08	20	100	5
sub09	–	200	2
sub10	–	100	5
sub11	–	100	2
sub12	20	200	2
sub13	10	200	2
sub14	10	200	5
sub15	20	100	2
sub16	10	200	2
Most Common	**10**	**200**	**2**

Table 7.22: Best hyperparameters per test subject using RF on between-subject, one characteristic.

Test Subject	Max Depth	N Estimators	Min Samples Split
sub04	20	100	5
sub05	20	200	2
sub07	20	100	2
sub08	20	100	5
sub09	10	100	5
sub10	–	200	2
sub11	–	100	5
sub12	–	200	5
sub13	10	200	2
sub14	20	200	2
sub15	20	100	2
sub16	10	200	2
Most Common	**20**	**200**	**2**

Table 7.23: Best hyperparameters per test subject using RF on between-subject, multiple characteristics.

Subject	Learning Rate	Max Depth	N Estimators	Subsample
sub04	0.1	5	200	0.8
sub05	0.1	3	200	1.0
sub07	0.1	5	200	0.8
sub08	0.01	5	200	1.0
sub09	0.1	5	200	0.8
sub10	0.1	3	200	1.0
sub11	0.1	5	200	0.8
sub12	0.1	5	100	1.0
sub13	0.1	5	200	0.8
sub14	0.1	5	200	1.0
sub15	0.1	3	200	1.0
sub16	0.1	5	100	1.0
Most Common	**0.1**	**5**	**200**	**1.0**

Table 7.24: Best hyperparameters per subject using XGBoost on within-subject, one characteristic, slice-wise approach.

Subject	**Learning Rate**	**Max Depth**	**N Estimators**	**Subsample**
sub04	0.1	5	200	1.0
sub05	0.1	5	200	0.8
sub07	0.1	5	200	1.0
sub08	0.1	5	200	1.0
sub09	0.1	5	200	1.0
sub10	0.1	5	100	1.0
sub11	0.1	5	200	1.0
sub12	0.1	5	200	1.0
sub13	0.1	5	100	1.0
sub14	0.1	5	200	1.0
sub15	0.1	5	200	1.0
sub16	0.1	5	200	1.0
Most Common	**0.1**	**5**	**200**	**1.0**

Table 7.25: Best hyperparameters per subject using XGBoost on within-subject, one characteristic, trial-wise approach.

Subject	**Learning Rate**	**Max Depth**	**N Estimators**	**Subsample**
sub04	0.1	3	200	1.0
sub05	0.1	5	200	1.0
sub07	0.1	5	100	0.8
sub08	0.01	5	200	0.8
sub09	0.1	5	200	1.0
sub10	0.1	5	100	1.0
sub11	0.1	5	200	0.8
sub12	0.1	5	200	1.0
sub13	0.01	5	200	0.8
sub14	0.1	5	100	1.0
sub15	0.1	5	100	1.0
sub16	0.1	5	200	1.0
Most Common	**0.1**	**5**	**200**	**1.0**

Table 7.26: Best hyperparameters per subject using XGBoost on within-subject, multiple characteristics, slice-wise approach.

Subject	**Learning Rate**	**Max Depth**	**N Estimators**	**Subsample**
sub04	0.1	5	100	0.8
sub05	0.1	3	200	1.0
sub07	0.1	3	200	1.0
sub08	0.1	3	200	1.0
sub09	0.1	5	200	1.0
sub10	0.1	5	200	1.0
sub11	0.1	3	200	1.0
sub12	0.1	5	200	1.0
sub13	0.01	5	200	0.8
sub14	0.1	5	100	1.0
sub15	0.1	3	200	1.0
sub16	0.1	5	100	1.0
Most Common	**0.1**	**5**	**200**	**1.0**

Table 7.27: Best hyperparameters per subject using XGBoost on within-subject, multiple characteristics, trial-wise approach.

Test Subject	**Learning Rate**	**Max Depth**	**N Estimators**	**Subsample**
sub04	0.1	5	200	0.8
sub05	0.01	5	100	1.0
sub07	0.1	5	100	0.8
sub08	0.01	3	100	1.0
sub09	0.1	5	100	1.0
sub10	0.1	5	200	0.8
sub11	0.1	5	200	1.0
sub12	0.1	5	100	0.8
sub13	0.1	3	100	0.8
sub14	0.1	3	100	1.0
sub15	0.1	5	200	1.0
sub16	0.1	3	200	1.0
Most Common	**0.1**	**5**	**200**	**1.0**

Table 7.28: Best hyperparameters per test subject using XGBoost on between-subject, one characteristic.

Test Subject	**Learning Rate**	**Max Depth**	**N Estimators**	**Subsample**
sub04	0.1	5	100	0.8
sub05	0.01	5	100	1.0
sub07	0.01	3	100	1.0
sub08	0.1	3	200	1.0
sub09	0.01	3	100	1.0
sub10	0.1	5	200	1.0
sub11	0.01	5	100	0.8
sub12	0.1	3	100	1.0
sub13	0.1	3	200	1.0
sub14	0.1	3	100	1.0
sub15	0.1	5	100	0.8
sub16	0.1	3	200	0.8
Most Common	**0.1**	**3**	**100**	**1.0**

Table 7.29: Best hyperparameters per test subject using XGBoost on between-subject, multiple characteristics.