SAPIENZA
UNIVERSITÀ DI ROMA

TA-EPE: A Transformer-based Approach to EEG Personality Estimation

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Computer Science

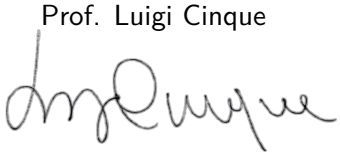
Candidate

Alessio Lucciola

ID number 1823638

Thesis Advisor

Prof. Luigi Cinque



External Advisor

Dr. Romeo Lanzino

Academic Year 2023/2024

TA-EPE: A Transformer-based Approach to EEG Personality Estimation
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: lucciola.1823638@studenti.uniroma1.it

Abstract

Personality estimation involves predicting individual personality traits using various data sources, such as physiological signals, behavioral patterns, or brain activity. This task is intriguing because it can provide deeper insights into human behavior, enhance personalized user experiences, and improve mental health interventions.

This thesis proposes an innovative approach to personality estimation from EEG data using Transformer architectures. The main contribution is demonstrating the effectiveness of deep learning, particularly Transformers, in this complex task and exploring various validation schemes to ensure robust performance comparison. While EEG-based personality prediction is challenging due to the intricacies of brain signals, this work highlights the potential of advanced neural networks to improve accuracy.

The results demonstrate that our proposed approach significantly outperforms existing methods. To ensure a fair comparison with the related works, two validation schemes have been employed: Leave One Out (LOOCV) and K-FOLD cross-validation. With the LOO validation scheme on the AMIGOS dataset, our EEGTransformer achieved an impressive F1 score of 81.72%, a substantial improvement over the 43.40% F1 score of the current state-of-the-art, marking a difference of 38.32 percentage points. Under the KFOLDCV scheme, our model attained an accuracy of 99.30%, surpassing the best-reported accuracy of 90.58% by 8.72 percentage points. For the ASCERTAIN dataset, our EEGTransformer achieved an F1 score of 89.39% with LOO validation, significantly exceeding the 43.40% F1 score of the previous leading method, representing a remarkable improvement of 45.99 percentage points. This research not only highlights the effectiveness of our approach but also paves the way for future advancements in EEG-based personality prediction.

Contents

Chapter 1

Introduction

Artificial Intelligence (AI) has become an essential component of modern life, transforming various sectors by automating tasks, enhancing decision-making, and enabling new forms of interaction. Recent advancements in AI have led to the proliferation of AI-driven applications such as chatbots, autonomous vehicles, and Internet of Things (IoT) devices, demonstrating AI's expanding influence in everyday activities.

AI is currently utilized across many critical sectors including, but not limited to:

- **Healthcare**: AI algorithms enhance diagnostics, patient management, and treatment planning, increasing the precision and efficiency of medical care;
- **Finance**: Financial institutions deploy AI for fraud detection, risk assessment, and personalized banking services, improving security and customer satisfaction;
- **Logistics**: AI optimizes supply chain management and delivery routing, leading to cost savings and increased efficiency;
- **Tourism**: AI provides personalized travel recommendations, automates customer service, and optimizes booking processes;

A burgeoning area of AI research is the analysis of neural patterns, with applications ranging from brain-computer interfaces to mental health diagnostics. Among these applications is **personality estimation** [37], a process that predicts an individual's personality traits from observed data, including physiological signals and behavioral patterns. This task has significant implications for enhancing personalized user experiences, improving mental health assessments, and advancing human-computer interaction.

Traditionally, personality traits have been assessed through psychometric tools and self-report questionnaires. However, these methods can be subjective and time-consuming. Recent advances propose using **electroencephalogram (EEG)** signals for personality estimation [41]. EEG provides a non-invasive way to measure brain activity in real time, capturing cognitive and emotional processes that correlate with personality traits.

This research, titled "TA-EPE: A Transformer-based Approach to EEG Personality Estimation", aims to develop a deep learning model, specifically a **Transformer-based architecture** [7], to estimate personality traits from EEG data. Transformers, initially developed for natural language processing, have demonstrated exceptional capabilities in handling sequential data through self-attention mechanisms. By

adapting the Transformer model to process EEG signals represented as mel spectrograms, this work seeks to combine the advantages of time-frequency analysis with the powerful feature extraction capabilities of Transformers.

More in detail, the model's approach involves converting EEG signals into mel spectrograms, capturing the frequency and temporal characteristics of the signals. This representation is then processed by the Transformer model to extract relevant features for personality trait estimation. The anticipated benefits of this approach include improved accuracy, and generalization in personality estimation compared to traditional methods and simpler machine learning models.

The ultimate goal is to enhance the reliability and applicability of EEG-based personality estimation, contributing to the broader field of AI-driven analysis of human behavior and expanding the potential applications of AI in personalized technologies.

The thesis is structured as follows:

- Chapter 2 provides an overview of deep learning concepts relevant to the study;
- Chapter 3 summarizes existing research on personality estimation using EEG;
- Chapter 4 describes the methodology, including the Transformer model architecture and preprocessing of EEG data into mel spectrograms;
- Chapter 5 presents the experimental setup, results, and a comparison with state-of-the-art techniques;
- Chapter 6 concludes the findings, discusses the implications and suggests directions for future research.

Chapter 2

Background

Many of the approaches that will be explained in Chapter 3 make use of **Artificial Intelligence** to capture patterns in brain signals. Artificial intelligence is the broader field of computer science that deals with the creation of intelligent machines. Although this field is often generalized, the main models that are exploited within the applications that are used by users every day make use of machine learning and deep learning which are specific techniques used to obtain artificial intelligence. **Machine learning** refers to the ability of computers to learn without being explicitly programmed and involves algorithms that analyze data, identify patterns, and make predictions based on those patterns. **Deep learning** is a subset of machine learning that uses artificial neural networks, which are complex algorithms inspired by the structure and function of the human brain. Deep learning excels at handling large amounts of unstructured data, like images, text, speech, and even brain signals [21].

This section is organized as follows:

- Section 2.1 introduces the various learning paradigms;
- Section 2.2 explains the classification tasks addressed often used in deep learning tasks;
- Section 2.3 discusses the learning process and the optimizers used during training;
- Section 2.4 provides an overview of the models considered in this work and compares them with related approaches.

2.1 Learning paradigms

Learning refers to the process by which a neural network adjusts its internal parameters (weights and biases) to minimize the difference between its predicted output and the actual output, given a set of input data. There are two main learning paradigms:

- **Supervised learning**: In supervised learning, the algorithm learns from a labeled dataset, where each example consists of input data paired with corresponding output labels. The goal of supervised learning is to learn a mapping from input variables to output variables based on the labeled data. Let's assume that we have a dataset consisting of m labeled examples:

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})$$

where x^i represents the input features of the i-th example, and y^i represents the corresponding output label. The goal is to learn a function $h : X \to Y$, where X is the input space and Y is the output space, such that $h(x)$ approximates the true output label y as accurately as possible for any input x. The learned function $h(x)$ is typically represented by a model parametrized by weights (w) and biases (b).

- **Unsupervised learning**: In unsupervised learning, the algorithm learns from an unlabeled dataset, where no explicit output labels are provided. The goal of unsupervised learning is to find patterns, structures, or relationships within the data without explicit guidance. Examples of unsupervised learning algorithms include k-means clustering, hierarchical clustering, principal component analysis (PCA), and autoencoders.

2.2 Classification tasks

Supervised learning has two fundamental types of learning tasks which are regression and classification. The goal of **regression** is to predict continuous numerical values rather than discrete categories. A typical example is predicting house prices based on features such as location, size, and number of bedrooms, predicting stock prices, and estimating sales revenue. Another important task is **classification** where the goal is to predict the categorical class labels of new instances based on past observations. It is necessary to focus on this task since personality estimation requires assigning a class (i.e. a personality) to a specific EEG signal, hence it is an example of a classification task. Other examples include spam email detection, sentiment analysis, image recognition, and medical diagnosis. There are different types of classification tasks:

- **Binary Classification**: It involves classifying instances into one of two classes. For example, predicting whether an email is spam or not spam;
- **Multiclass Classification**: In this case, there are more than two classes, and the task involves classifying instances into one of multiple classes. For example, classifying images of animals into categories like cats, dogs, and birds;
- **Multi-label Classification**: In multi-label classification, each instance can belong to multiple classes simultaneously. For example, classifying news articles into categories such as politics, sports, and entertainment where an article can belong to more than one category.

Imbalanced classification is when classes are not represented equally in the dataset. One class may have significantly more instances than the others. This could lead to biases during the learning process which could result in the model focusing more on learning patterns from the majority class and potentially ignoring patterns from the minority classes. There are various techniques to solve this issue such as *resampling methods* (e.g., oversampling the minority class, undersampling the majority class) often used with *data augmentation* which consists in artificially increasing the size of a dataset by applying various transformations to existing data samples.

2.3 Learning process and optimizers

The learning process of a deep learning algorithm involves several key steps that can vary depending on the problem domain. However, this general summary aims to provide an overview of the structure that the candidate intends to follow for the construction of the personality estimation model:

1. **Data collection and preparation**: The first step in the learning process is to collect and prepare the data. This involves gathering relevant data samples that represent the problem domain and preparing them for input into the learning algorithm. This step often includes feature extraction (i.e. extracting vectors from brain signals) and feature selection (i.e. understanding which features help most in pattern detection);

2. **Model selection**: It is necessary to select the appropriate model architecture or algorithm based on the nature of the problem, the type of data available, and the desired output. In the next sections, several methods are going to be exposed;

3. **Weight initialization**: After selecting the model, parameters must be initialized. This step is crucial, as the initial values of the model parameters can influence the learning process and the final performance of the model;

4. **Forward propagation**: Once the model is initialized, the training data is fed forward through the model. This process is called forward propagation, where the input data is passed through the layers of the model, and computations are performed to generate predictions or outputs;

5. **Loss Calculation**: After obtaining predictions from the model, the next step is to calculate a loss or cost function that quantifies the difference between the model's predictions and the actual target values in the training data. For binary and multi-label classification tasks, a commonly used loss function is the **Binary Cross-Entropy (BCE) Loss**. Assume that the model predicts a probability p for a single class (with a value between 0 and 1). The true class label is represented by y, where $y = 1$ for the positive class and $y = 0$ for the negative class. The formula to calculate the binary cross-entropy loss for a single sample is:

$$L(y, p) = -\left[y \log(p) + (1 - y) \log(1 - p)\right]$$

   This loss function calculates the negative logarithm of the predicted probability for the true class label. For each sample, if the true label y is 1, the loss increases as the predicted probability p deviates from 1. Conversely, if y is 0, the loss increases as p deviates from 0. This helps to penalize incorrect predictions more heavily and drives the model towards better accuracy;

6. **Backward propagation**: With the loss calculated, the learning algorithm uses an optimization technique such as **gradient descent** to adjust the model parameters in a direction that minimizes the loss. The goal is to minimize an objective (i.e. loss) function $L(\theta_t)$ parametrized by the model's parameter $\theta_t \in \mathbb{R}^d$ in the iteration t by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta L(\theta_t)$ w.r.t. the parameters. After computing the gradients, the model parameters are updated using the chosen optimization algorithm (e.g., stochastic gradient descent, Adam optimizer).

The learning rate η, which determines the size of the parameter updates, is an important hyperparameter that affects the convergence and stability of the learning process. The gradient descent formula is:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t)$$

Stocastic gradient descent is a variant of gradient descent where the model parameters are updated after computing the gradient of the loss function with respect to a randomly selected subset of the training data (mini-batch). This introduces stochasticity into the optimization process, which can help avoid getting stuck in local minima and speed up convergence. One popular optimization method is **Adaptive Moment Estimation (ADAM)** which is a method that computes adaptive learning rates for each parameter. The update rule for Adam is given by:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla_\theta L(\theta_t)$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2)(\nabla_\theta L(\theta_t))^2$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}$$

$$\hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}} + \epsilon} \hat{m}_{t+1}$$

where m_t and v_t are respectively estimates of the first moment (mean) and second moment (variance) of the gradients, β_1 and β_2 are decay rates for the moment estimates, typically close to 1 (e.g. 0.9 and 0.999, respectively) and ϵ is a small constant added to the denominator for numerical stability;

7. **Training iteration**: Steps 4-6 are repeated iteratively until a convergence criterion is met (e.g. a certain target epoch is reached);

8. **Validation and Testing**: Throughout the learning process, the model's performance is evaluated using validation data to monitor its progress and prevent overfitting. Once training is complete, the final model is evaluated on unseen test data to assess its generalization performance.

2.4 Models

Classical Machine Learning

Before explaining how deep learning works, it is necessary to briefly introduce some classical machine learning algorithms that were used by some papers exposed in the next sections:

- **K-Nearest Neighbors (KNN)** [22]: KNN relies on the assumption that similar data points tend to belong to the same class or have similar output values. For classification, the algorithm assigns the majority class among the k nearest neighbors of a given data point;

- **Support Vector Machine (SVM)** [38]: SVM aims to maximize the margin between the support vectors (data points closest to the decision boundary) and the decision boundary, which leads to better generalization. In classification, SVM finds the optimal hyperplane that separates data points belonging to different classes with the maximum margin and tries to make a prediction by understanding in which part of the hyperplane it lies;
- **Naive Bayes** [31]: It is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of feature independence. It calculates the probability of each class given the input features and selects the class with the highest probability (maximum a posteriori).

These kinds of algorithms typically rely on handcrafted features, have simpler model architectures, offer interpretability, and are computationally less demanding compared to deep learning models. These algorithms generalize well to diverse datasets but may struggle with high-dimensional data and complex patterns [29].

Perceptron

Moving towards deep learning, one of the simplest Artificial Neural Network (ANN) architectures, consisting of a single layer of neurons with direct connections to the input layer is the **perceptron**. A perceptron consists of:

- Input layer: This layer receives the input features;
- Neurons: Each neuron computes a weighted sum of its inputs and applies an activation function to produce the output;
- Output: It is typically binary (0 or 1) or a real-valued number depending on the activation function used.

Formally, let's consider a perceptron with n input features. The output y of the perceptron is computed as follows:

$$y = f(\sum_{i=1}^{n} w_i x_i + b)$$

where x_i is the i-th input feature, w_i is the weight associated with the i-th input feature, b is the bias and f is an activation function. **Activation functions** introduce non-linear transformations to the output of each neuron in a neural network layer and allow neural networks to capture more complex patterns in the data. Without activation functions, the composition of multiple layers in a neural network would collapse into a single linear transformation, rendering the network incapable of learning non-linear relationships in the data. There are several types of activation functions (e.g. ReLU, Tanh, Sigmoid) used in deep learning, each with its own advantages and disadvantages that can be chosen depending on the specific domain.

Multilayer Perceptron

A perceptron can be considered as the building block of more complex neural network architectures. A **Multilayer Perceptron (MLP)** consists of multiple layers of neurons, each layer fully connected to the next layer. It is an extension of a perceptron in fact the general structure is the same:

- Input layer: It consists of neurons that represent the input features of the data where each neuron corresponds to one feature, and the number of neurons in the input layer is equal to the dimensionality of the input data;
- Hidden layers: They are intermediate layers between the input and output layers. Each hidden layer consists of multiple neurons, and the number of hidden layers and the number of neurons in each hidden layer are configurable parameters of the model. Again, neurons in the hidden layers apply an activation function to the weighted sum of inputs from the previous layer to guarantee non-linearity;
- Output layer: It consists of neurons that represent the outputs or predictions of the model. For multiclass classification tasks, there is one neuron for each class, with each neuron outputting the probability of belonging to that class.

When talking about **Neural Networks (NN)** and Multilayer Perceptrons (MLP) we often refer to the same kind of architecture.

Convolutional Neural Network

Convolutional Neural Networks (CNN) is a powerful type of artificial neural network that excels at capturing patterns in data with a grid-like structure like images and videos, even though they are commonly used to solve other tasks such as Natural Language Processing (NLP), recommendation systems, time series forecasting, and many others. Unlike traditional neural networks that treat each data point individually, CNNs leverage the inherent structure of the data. The main differences with Neural Networks (NN) are:

- **Local Connectivity**: Unlike fully connected layers in NNs where each neuron connects to all neurons in the previous layer, CNNs have neurons in a convolutional layer that connect only to a small region of the previous layer;
- **Parameter Sharing**: Filters (kernels) in CNNs are applied across the entire image. These filters are shared across different neurons within a feature map, and this helps to drastically reduce the number of parameters compared to fully connected NNs.

The **convolutional layer** is the core building block of CNNs and makes use of the convolution operations to extract features. Convolution involves sliding a filter (a.k.a. kernel) over the input data and computing dot products to produce feature maps. Assuming to have an image I and a kernel F at a spatial location (m, n) of the image, we can represent the 2D convolution operation as follows:

$$G[m, n] = I * F = \sum_{k,l} I(m - k, n - l)F[k - l]$$

The output of the convolution operation is a feature map that represents the presence of specific patterns or features in the input data (as already explained in Neural Networks). **Pooling layers** are used to downsample the feature maps, reducing their spatial dimensions while retaining important information. Stride refers to the step size with which the pooling window moves across the input feature map. A stride of 1 means the window moves one pixel at a time, resulting in overlapping regions between adjacent pooling windows while a larger stride leads to a greater reduction of the spatial dimensions. Max pooling is a common pooling operation,

where the maximum value within each region of the feature map is retained, even though there are other pooling operations such as average pooling. Mathematically, max pooling with a pooling size of $2x2$ and a stride of 2 can be represented as:

$$MaxPooling(m,n) = \max(I(2m,2n), I(2m,2n+1), I(2m+1,2n), I(2m+1,2n+1))$$

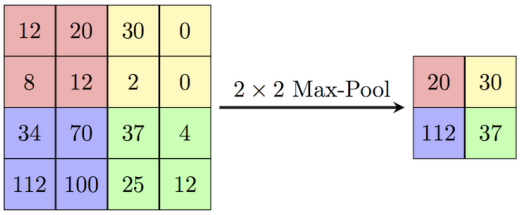


Figure 2.1. Example of a pooling operation with a pooling size of 2x2 and a stride of 2. Image taken from [26].

After several convolutional and pooling layers, the feature maps are flattened and passed through one or more fully connected layers to make the final classification.

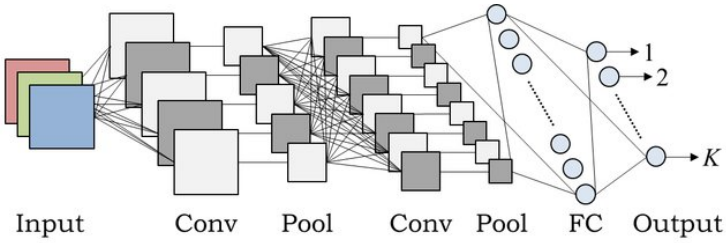


Figure 2.2. Example of a CNN architecture. Image taken from [2].

Several works presented so far, instead of training a model from scratch, make use of **transfer learning**. It is a technique where a model trained on one task is reused or adapted as the starting point for a model on a second related task. It allows users to leverage the knowledge gained from the source task to improve learning and performance on the target task, especially when the target task has limited data available for training. It usually consists of two steps:

- Feature Extraction: The pre-trained model is used as a fixed feature extractor. The weights of the pre-trained model are frozen, and only the final layers (i.e. fully connected layers) are trained on the target task data;
- Fine-tuning: The pre-trained model is further trained (fine-tuned) on the target task data.

Several pre-trained CNN-based models were used for personality estimation such as VGG [43]. Another popular pre-trained model is ResNet [20].

LSTM

Another kind of architecture exploited for solving personality estimation using brain signals is the **Long Short-Term Memory (LSTM)** network which is a specific type of **Recurrent Neural Network (RNN)**. RNNs are a class of neural

networks designed for sequential data processing, where each input instance is part of a sequence, and the order of inputs matters. Unlike neural networks, which process each input independently, RNNs maintain an internal state (hidden state) that captures information about previous inputs seen in the sequence, and this ability makes them suitable for many tasks such as time series prediction. In personality estimations, they were used for capturing temporal dependencies in brain signals. One major limitation of traditional RNNs is the vanishing gradient problem, where gradients tend to become extremely small as they propagate back through time during training. For this reason, LSTMs were introduced to try to decrease the likelihood of its occurrence. The core component of an LSTM unit is the memory cell, which consists of a cell state (c_t) that retains long-term information over time, and various gates that control the flow of information:

- Forget gate (f_t): It determines how much of the previous cell state to retain or forget;
- Input gate (i_t): It determines how much new information to add to the cell state;
- Output gate (o_t): It determines how much of the cell state to expose as the output.

The information flow in an LSTM is governed by the gates that control how information is added to, retained in, and extracted from the memory cells over time. This architecture allows LSTMs to effectively capture long-term dependencies in sequential data and make informed decisions based on the input sequence.

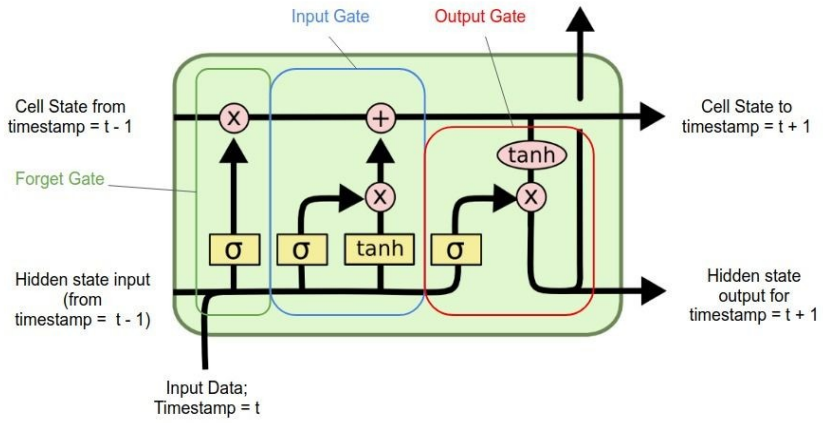


Figure 2.3. Information flow inside an LSTM layer. Image taken from [24].

Transformers

Transformers [7] gained significant popularity, especially in Natural Language Processing (NLP) tasks, due to their ability to capture long-range dependencies and handle sequential data efficiently. They rely heavily on the self-attention mechanism, which allows them to weigh the importance of different input tokens when processing a sequence. At the core of transformers is the self-attention mechanism, which computes attention scores between all pairs of positions in the input sequence to

determine the importance of each token compared to the others. There are different ways to compute attention but the authors of the original article proposed to use the scaled dot-product operation. Given a query matrix Q, a key matrix K and a value matrix V, the attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimensionality of the queries and keys. Usually, **multi-head attention** is used where the model attends different parts of the input sequence simultaneously, each focusing on different aspects or patterns. The key idea is to perform multiple independent attention operations in parallel, each with its own set of learned parameters (queries, keys, and values). Given an input sequence X of length N and dimension d_{model}, the multi-head attention mechanism with h heads can be formulated as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, head_2, .., head_h)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ represents the i-th attention head and W_i^Q, W_i^K, W_i^V are the learnable parameter matrices for each head. In order to perform the concatenation of the outputs of each head, a learnable parameter matrix W^O is used for computing the linear transformation.

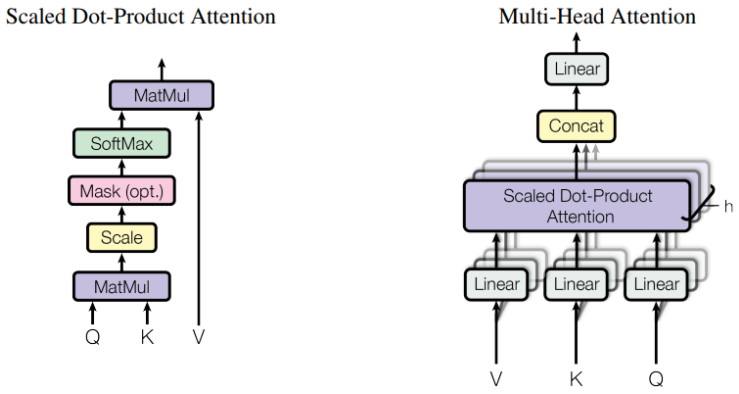


Figure 2.4. Scaled Dot-Product Attention and Multi-Head Attention consist of several attention layers running in parallel. Image taken from [7].

Another key component of transformers is the **positional encoding** that can be used to inject information about the position of tokens in the input sequence into the Transformer model. Since Transformers do not inherently understand the order of the input tokens like recurrent or convolutional models, positional encoding is essential for capturing sequential relationships. A commonly used method for positional encoding involves adding fixed sinusoidal functions of different frequencies to the embedding vectors of the input tokens. The positional encoding for a token at position pos and dimension i can be computed as:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

The architecture of transformers is composed of an **encoder** and a **decoder**. The encoder processes the input sequence and generates a sequence of hidden representations that capture the input's information. The decoder generates the output sequence based on the encoder's representations and previously generated tokens. While the Transformer architecture is commonly used as an end-to-end model for sequence-to-sequence tasks, it is possible to use the Transformer encoder alone to extract representations of input sequences, which can then be fed into downstream models for tasks such as text classification, sentiment analysis, encoding of brain signals and so forth.

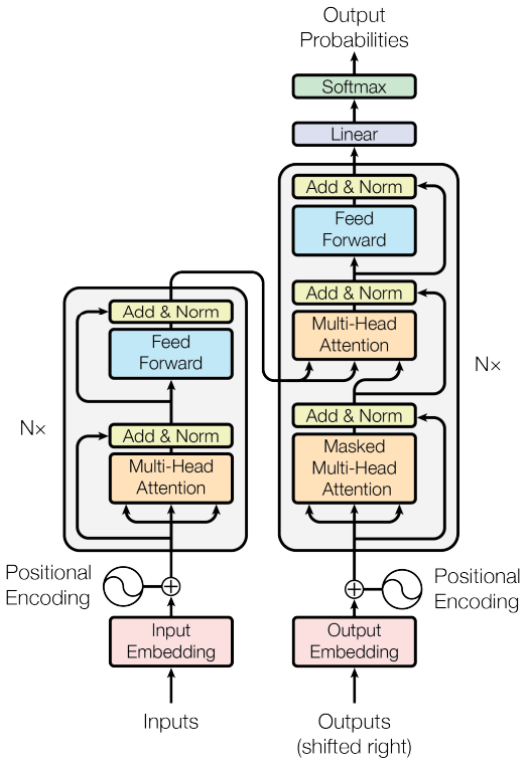


Figure 2.5. Architecture of a transformer (encoder on the left, decoder on the right). Image taken from [7].

Chapter 3

Related work

This section will cover notable related works:

- Section 3.1 presents some general EEG tasks already explored in research;
- Section 3.1 focuses on the specific task of personality estimation, which is also the objective of this thesis.

3.1 Related EEG tasks

In recent years, the study of signals coming from the brain is gaining growing interest in research, also thanks to the advent of artificial intelligence which has greatly facilitated the study of neural patterns. The integration of AI with neurotechnologies [41], including machine learning, deep learning, and data analytics, has revolutionized neural signal processing by enhancing the accuracy, speed, and efficiency of data analysis. Although numerous steps forward have been made, EEG signals are complex and tend to vary significantly across individuals, making it difficult to extract meaningful patterns. They still provide invaluable information and also have the advantage of capturing complex neural patterns at a high-speed rate, paving the way for the development of many potential applications in various domains. Some fields have already been explored:

- **Emotion recognition**: Emotion recognition involves identifying and classifying emotional states based on EEG signals. EEG data contain brain activity patterns associated with emotions such as happiness, sadness, anger, and fear, which can be analyzed using computational techniques. Emotion recognition based on EEG signals [9] has broad application prospects in entertainment and games, education, criminal investigation, assisted driving, health care, and other fields. The BiLSTM method [4] explored the possibility of performing emotion recognition using a specific deep learning model called a Bi-directional Long Short-Term Memory (Bi-LSTM) network that exploits the LSTM architecture in a bidirectional way. A standard LSTM processes information in one direction (chronologically). A Bi-LSTM tackles the sequence from both directions (forward and backward), allowing it to capture more complex relationships within the EEG data. The DCoT method [17] moves beyond traditional approaches like Bi-LSTMs and explores using transformers for EEG emotion recognition. This method proposes a novel architecture composed of a Depthwise Convolution useful for extracting local features from the EEG data and a transformer encoder that captures the long-range dependencies between

different segments of the EEG signal, potentially leading to a more accurate understanding of the underlying brain activity;

- **Motor imagery**: Motor imagery involves mentally simulating the execution of motor actions without physically performing them. EEG signals exhibit distinct patterns during motor imagery tasks, which can be utilized for brain-computer interface (BCI) applications and rehabilitation purposes. Computational neuroscience tasks for motor imagery [5] typically involve decoding EEG signals to discriminate between different imagined motor actions, such as left-hand vs. right-hand movements or specific gestures. Traditional methods like Transformed Common Spatial Pattern [6] relied on feature engineering (specifically on spatial filtering of EEG signals), where researchers manually identified specific patterns in EEG data associated with different motor imagery tasks. The EEG Classification of Motor Imagery method [28] proposed a new approach for classifying motor imagery based on Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTM) that combined together can capture both the spatial and temporal information of the EEG signals;

- **EEG-to-Text**: EEG-to-Text [35] is a developing technology that aims to translate brain activity directly into text. It uses electroencephalography (EEG), which measures electrical signals in the brain, to decode a person's thoughts into written words. This technology is particularly promising for people with speech limitations due to conditions like ALS or stroke. The Encoder-Decoder framework [18] drew inspiration from the encoder-decoder models in machine translation tasks and adapted this framework to translate brain activity (cortical activity) into written text. The encoder analyzes a sequence of neural activity (recorded through EEG) and compresses it into a more condensed representation, capturing the essence of the thought. The decoder takes the encoded representation and generates text word by word, translating the brain's message into an understandable sentence. The Open Vocabulary approach [45] extends this work by proposing a novel approach using pre-trained language models (like BART) to achieve open vocabulary EEG-to-text decoding, allowing for a much wider range of words and phrases;

- **Neuromarketing**: Neuromarketing [15] [44] is an emerging field that aims to analyze consumers' behaviors from a psychological viewpoint. The main objective of neuromarketing is to transfer commercial messages to consumers and increase the likelihood of purchasing a specific item. This field gives the possibility to extract hidden, reliable information from customers' brains based on their emotions and cognitions without making any inquiries. The KNN-based predictive model [36] proposes a method for analyzing EEG signals to capture customer preferences regarding the purchase of an electric car. This involves recording brain signals while submitting a questionnaire, and this information is then used to build a KNN-based predictive model. The Preference Classification method [3] investigated the possibility of detecting two preference states, namely pleasant and unpleasant, by using EEG and a self-defined deep neural network model trained on the DEAP dataset [11];

- **Social interaction**: Social interaction [12] explores how brain activity patterns change during social interactions compared to individual tasks. EEG can be exploited in this field to see if brain activity patterns differ when people are engaged in social interaction (conversation, cooperation) compared to being

alone or performing non-social tasks. It can also be used to investigate brain activity related to empathy, the ability to understand another person's emotions. The Social Emotion Perception method [23] explores how EEG can be used to understand how people perceive emotions expressed by others. A simple CNN-based classifier is used to scan social emotions and derive predictions. The Deep Learning applied to EEG method [10] aims to understand which areas of the brain are activated when people perceive social stimuli (both for face and body) by looking at EEG data. A model is then trained using deep learning architectures composed of CNNs.

3.2 Personality estimation

Another emerging task is personality estimation which involves studying EEG signals with the goal of inferring personality traits due to the potential information it can provide about individual differences in cognition, behavior, and mental health. Most of the works currently published [13] [19] [32] follow a standard solution approach which involves recording EEG signals to understand the stimuli and reactions to certain events. Typically a varied group of users is asked to watch images or videos (for example pieces of movies) with different scenarios, and an attempt is made to capture information on the psychological spectrum through the use of EEG. Once the brain signals have been captured, a questionnaire is given to each user to try to understand their personality and assign a label. There are some frameworks that have scientific value that are used for understanding personality:

- **The Big Five (OCEAN) Model** [33] [39]: It was developed in the 1980s through lexical studies analyzing personality-descriptive words in various languages. The five core dimensions taken into account are:
  - Openness to Experience (O): This describes a person's intellectual curiosity, creativity, and willingness to try new things;
  - Conscientiousness (C): This reflects a person's level of organization, self-discipline, and goal-oriented behavior;
  - Extraversion (E): This captures a person's sociability, energy level, and need for external stimulation;
  - Agreeableness (A): This describes a person's cooperativeness, empathy, and altruism;
  - Neuroticism (N): This reflects a person's tendency to experience negative emotions like anxiety, fear, and moodiness.

  This approach provides a way for understanding personality which is backed by extensive research, and it is widely recognised across different cultures and languages. Contrarily it may not capture the full complexity of personality and traits are not entirely independent, and some overlap can occur.

- **The Myers-Briggs Type Indicator (MBTI)** [8] [30]: It was developed by Isabel Briggs Myers and Katharine Briggs in the mid-20th century, based on the theories of Carl Jung. It is composed of four dichotomies:
  - Extraversion (E) vs. Introversion (I): How a person gains and expends energy (extroverts from interaction, introverts from reflection);

  - Sensing (S) vs. Intuition (N): How a person perceives and gathers information (sensors through facts and details, intuitive through hunches and possibilities);
  - Thinking (T) vs. Feeling (F): How a person makes decisions (thinkers logically, feelers emotionally);
  - Judging (J) vs. Perceiving (P): How a person prefers to structure their life (judging prefers closure and planning, perceiving prefers flexibility and adaptation).

  This approach is widely used in personal and professional development and offers a detailed description of personality types. The problems regarding the MBTI are that it lacks strong scientific backing and can be criticized for being overly deterministic and results can be subjective and depend on self-reporting.

Given the possibility of obtaining information on personality via EEG, the way has been opened to models capable of predicting it:

- **EEG-based personality prediction using genetic programming**: The genetic programming approach [13] aimed to infer an individual's personality based on the brain signals generated through emotions or feelings while watching the video clips. To achieve this goal, they created a dataset from brain signals collected from a pool of 65 people of various genders and ages. Users were asked to watch short videos during which EEG signals were collected with a portable device called "NeuroSky MindWave Mobile 2". Four sets of film clips were shown, each representing one of the four MBTI trait groups (e.g., Introversion vs. Extraversion). Each clip aimed to elicit the corresponding personality trait. One-minute buffer and a neutral clip were included between each set to minimize carry-over effects. After each set of clips, participants completed a self-evaluation questionnaire using a Likert scale (agree, neutral, disagree). The questionnaire had seven questions and assessed the specific personality traits targeted by the clips. Subsequently, the labels linked to the personality of the individuals were extracted using the MBTI personality test. In addition to using the dataset they created, they also used two other public datasets called "ASCERTAIN" and "AMIGOS". This research proposes a new model called BSHGP (Best first search, Standard crossover, Hill climbing Genetic Programming) for predicting MBTI personality traits using EEG data. The model uses four separate GP trees, one for each of the four MBTI trait groups (e.g., Introversion vs. Extraversion). Each tree represents a classifier for a two-class problem (belonging to one of the two opposing traits). An initial population of 100 individuals is generated, where each individual consists of these four GP trees. The model employs a genetic programming (GP) algorithm to evolve the population of GP trees over multiple generations. A fitness measure is used to regulate the generation. The cycle involves three main operators:

  - Reproduction: The top 10% of performing individuals (trees) are directly copied to the next generation;
  - BSH Crossover (proposed): This novel crossover operator combines Best First Search (BFS) which is applied to trees with depth up to 5 and explores all possible offspring, eventually selecting the two best based on fitness, Standard Crossover (SC) which is applied to a portion of the population and promotes diversity by swapping subtrees between two

parent trees and Hill Climbing (HC) that is applied to the remaining population and generates new offspring and replaces parents if their fitness improves. The fitness measure is computed as:

$$\text{Fitness} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = true positive, TN = true negative, FP = false positive, and FN = false negative;

– Mutation: A random modification is introduced to some individuals with a low probability of maintaining diversity.

Concerning the classification, a new individual (set of four trees) is evaluated based on its accuracy in predicting personality traits from unseen EEG data. The GP process stops when the maximum number of fitness measures are done, the total number of fitness measures in this analysis is used instead of entire generations as a stop criterion or if a classifier training accuracy reaches 100%.

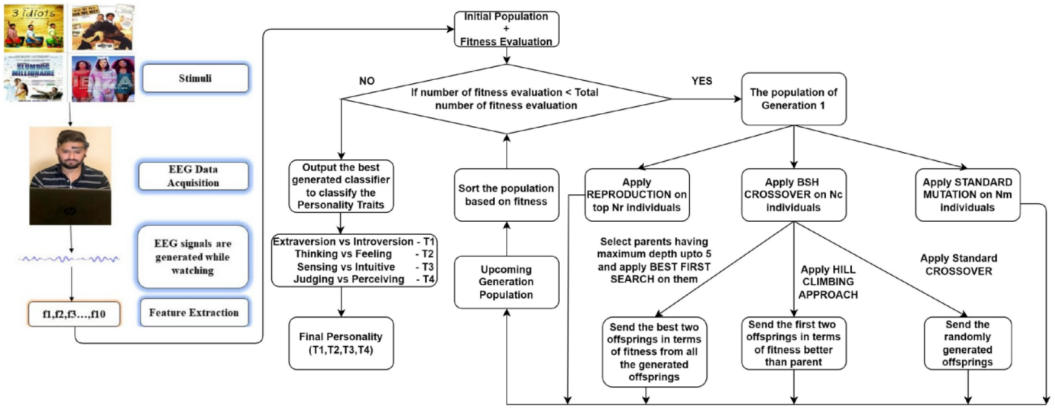


Figure 3.1. Summary of the GP process using BSH crossover for personality prediction proposed by the EEG-based personality prediction using genetic programming approach. Image taken from [13].

- **Personality-Based Emotion Recognition Using EEG Signals with a CNN-LSTM Network**: The CNN-LSTM method [16] aimed to improve the accuracy of the emotion recognition process by capturing both spatial and temporal dependencies in the EEG signals and considering individual differences in personality traits. This work is presented as it includes the use of a sub-architecture to study personality traits, but the main objective of the paper was to use this data to perform emotion recognition. The study recruited 270 volunteers and divided them into three groups based on personality traits measured by a standardized test. The chosen groups were the unstable extrovert group, the unstable introvert group, and the normal group. Researchers used pictures and videos designed to evoke specific emotions (happiness, sadness, neutral) while recording EEG data. They ensured the chosen stimuli effectively induced emotions by having a separate group of volunteers rate them on a standardized scale called PANAS [40]. In order to reach the designated objective, researchers decided to use a CNN-LSTM network to classify emotional states. After the EEG feature extraction, the pre-processed EEG signals were fed into a CNN network, which utilized spatial

features for emotion classification. They decided to use transfer learning by exploiting the VGG-16 pre-trained CNN. Furthermore, the spatial features obtained by the CNN network were used as inputs for the LSTM network, which focuses on capturing the temporal dynamics between these features for personality train classification. The network is composed of three LSTM layers, each followed by a fully connected layer used to map the output of the LSTM layers to the Big Five personality trains. The input of the LSTM is the flattened features extracted from the pre-processed EEG data by the CNN previously exposed. The EEG data were divided into windows of fixed duration, and each window was labeled with the participant's personality traits. The output of the fully connected layers, to predict the participant's Big Five personality traits (extroversion, agreeableness, conscientiousness, neuroticism, and openness) and the dense layer, are used for classification. To evaluate the performance of the LSTM network, they used a leave-one-subject-out cross-validation approach, where we trained the LSTM network on all but one participant's data and tested its performance on the left-out participant's data. Finally, they take the features extracted by the CNN and LSTM and use them for the final emotion classification.

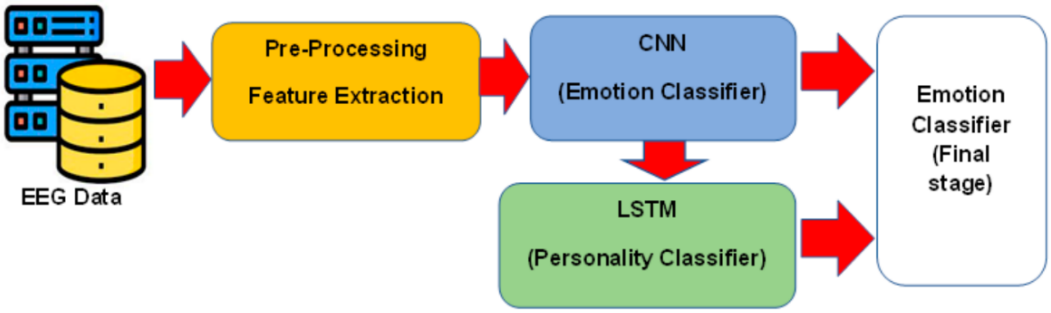


Figure 3.2. Summary of the emotion recognition system implemented in the CNN-LSTM Network approach. Image taken from [16].

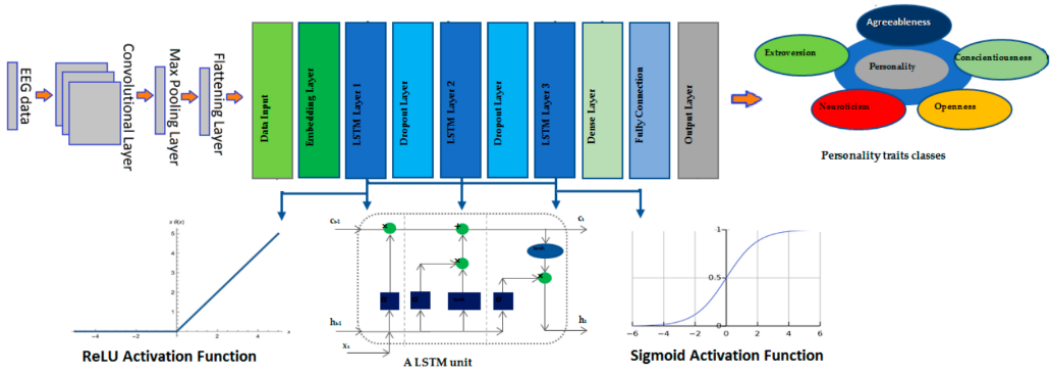


Figure 3.3. LTSM architecture implemented in the CNN-LSTM Network approach to perform personality estimation using EEG signals preprocessed by a pre-trained CNN. Image taken from [16].

The LSTM model achieved an average accuracy of 90.72% in predicting personality traits.

- **EEG-Based Personality Prediction Using Fast Fourier Transform and DeepLSTM Model**: DeepLSTM [14] proposed a way to perform personality prediction by using an LSTM network. This work precedes that of the same authors which was described previously so the data collection and preprocessing parts are exactly the same. On this occasion, they proposed to use a simple DeepLSTM architecture composed of 3 layers with a decreasing number of memory units (512, 256, 128) as it goes deeper. Dropout layers (with a probability of 0.2) are inserted between existing layers to prevent overfitting and improve training speed. The "tanh" activation function is used in most layers, while "softmax" is used in the final layer to generate probability scores for five personality classes. For training and testing the model, four cross-validation schemes are used (50-50, 60-40, 70-30, and 10-fold). The DeepLSTM model achieves higher maximum classification accuracy on both datasets with 10-fold cross-validation (95.32% for ASCERTAIN, 96.94% for the proposed dataset).

- **Personality traits classification from EEG signals using EEGNet**: EEGNet [42] proposed a way to predict personality traits using a deep neural network architecture. The work was accomplished using the AMIGOS dataset. EEG data from 40 subjects was used after excluding 2 due to missing personality information. Three datasets (D1, D2, D3) were created with varying preprocessing levels:

  - D1 (most preprocessed): Bandpass filtering, bad channel removal, ICA for artifact removal, channel interpolation, common average reference.
  - D2 (moderately preprocessed): Bandpass filtering only.
  - D3 (least preprocessed): Bandpass filtering to remove delta band noise, standardization.

  Each dataset was split into training (70%), validation (15%), and test (15%) sets using stratified splitting to ensure a balanced representation of subjects across sets. EEG signals were segmented into non-overlapping 3-second windows for each video presentation. Moreover, personality trait scores were converted into two classes: below and above the mean score. Agreeableness was the only perfectly balanced trait, while others had slight imbalances. The architecture used to create the personality classification model is the EEGNet, which is a simple CNN composed of three main blocks. The first block applies temporal filters to capture features over time in the EEG signal and uses depthwise spatial filters to extract features from each channel. The second block performs depthwise convolution to capture features across time steps and uses pointwise convolutions to combine features across channels. The third block is the final classification layer.

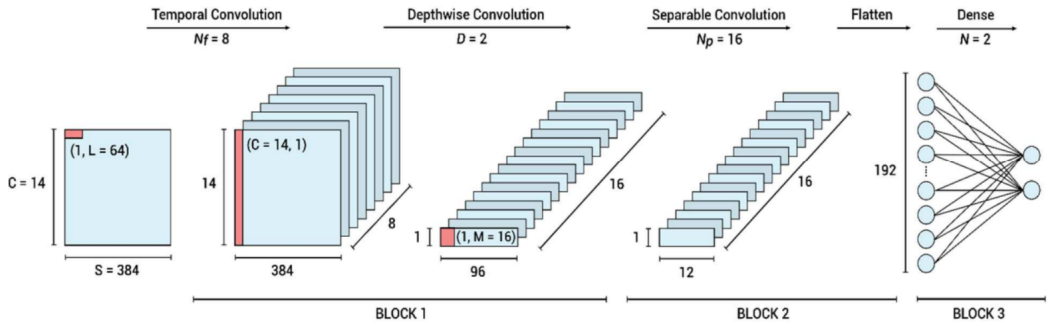


Figure 3.4. Architecture of the EEGNet model. Image taken from [42].

They managed to reach an average accuracy of 90.5% with a peak in Agreeableness and Extraversion (i.e., 93% and 92.1% respectively) using the D3 dataset.

- **Automatic Recognition of Personality Profiles Using EEG Functional Connectivity during Emotional Processing**: The Personality Profiles Recognition method [25] explores the possibility of automatically recognizing personality traits using EEG data recorded during emotional processing tasks regarding watching videos. Again, each video aimed to elicit specific emotions (happiness, sadness, fear, disgust, anger, or neutral) during which EEG signals were captured. They made use of the "AMIGOS" dataset where participants self-reported their emotions after each video using six basic emotion categories. Additionally, they rated each video on valence (positive vs. negative) and arousal (calm vs. excited). Based on these ratings, the videos were categorized into four groups: High Valence High Arousal (HVHA), High Valence Low Arousal (HVLA), Low Valence High Arousal (LVHA) and Low Valence Low Arousal (LVLA). The authors proposed to categorize each dimension into "low" or "high" traits (e.g. low and high extroversion). Using a simple mean value led to unbalanced classes. For this reason, they applied k-means clustering separately to each of the five personality dimensions. Once the EEG signals were obtained, the ReliefF algorithm [34] was used to reduce the number of features. After this step, classification was performed (in this case binary classification since they divided the personality traits into high/low cases). They measured the performance on the HVHA, LVHA, and the fusion (both) scenarios obtaining an average accuracy of 74.08%, 65.44%, and 82.18% respectively.

- **AMIGOS: A Dataset for Affect, Personality, and Mood Research on Individuals and Groups**: The AMIGOS dataset [19] is designed to study the interplay between emotions, personality traits, mood, and social context. Detailed in section Section 5.1, it plays a critical role in this work. AMIGOS profiles participants' personalities using the Big Five traits: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. It integrates multiple modalities such as EEG signals for comprehensive personality estimation. The dataset also explores personality prediction using Support Vector Machines (SVM) and evaluates different experimental conditions based on the emotional response elicited by videos.

- **ASCERTAIN: Emotion and Personality Recognition Using Commercial Sensors**: This paper introduces the ASCERTAIN dataset [32], designed

for the assessment of emotion and personality using commercial off-the-shelf sensors. Section 5.1 elaborates on its use in this work. ASCERTAIN incorporates the Big Five personality traits as in AMIGOS [19] and utilizes modalities such as EEG, ECG (Electrocardiogram), and GSR (Galvanic Skin Response) to capture physiological responses. The dataset facilitates research on emotion recognition and personality estimation, providing insights into how commercial sensors can be used to predict individual differences in personality traits and emotional states. The study also evaluates various machine learning approaches, including SVMs, for predicting personality traits based on the physiological data collected.

Chapter 4

Proposed approach

This section will explain the proposed approach and its key concepts:

- Section 4.1 describes the preprocessing of the EEG signals, the windowing process, and the usage of mel spectrograms for solving the task;
- Section 4.2 exposes the proposed architecture used for training a model for personality estimation.

4.1 Handling and preprocessing of EEG signals

The initial task involved **uploading EEG signals**, which were then preprocessed for training a deep learning model. Given the diversity of datasets, a generalized dataset class named *EEG_classification_dataset* was developed. This class aimed to standardize the handling of EEG signals across different datasets to ensure consistent results. For each dataset, the brain's electrical activity for each subject during each trial was extracted, including the EEG signals, the corresponding subject, and their personality labels.

4.1.1 Labels discretization

In both datasets tested in this work, **personality labels** are uniformly associated with all EEG signals for each subject, meaning that all trials of a subject share the same personality labels. Each dataset includes five personality traits: extroversion, agreeableness, conscientiousness, emotional stability, and openness to experience (creativity). These traits are quantified with labels from 0 to 4, and each label is scored on a scale from 1 (low) to 7 (high), indicating the extent to which a subject exhibits a specific personality trait. A discretization process was applied to the personality labels to convert the task into a **multilabel classification** problem. Two discretization approaches were tested:

- **Personality mean**: Following the methods used in EEGNet [42] and Automatic Recognition of Personality Profiles method [25], this approach involves calculating the mean value for each personality trait and binarizing the labels based on this threshold. For example, if the mean value for extroversion is 4.5, values below this threshold are labeled as 0, and values above are labeled as 1. This method helps ensure that each class has a balanced number of instances, as the distribution of personality traits is not typically normal. Hence, this approach balances class distribution by adjusting to the mean, reducing the

impact of outliers. At the same time, the mean may not fully represent trait variability, leading to an inconsistent discretization of the classes.

- **Fixed mean**: This approach uses a fixed threshold of 4 for discretizing all personality traits, given that trait values range from 1 to 7. This fixed median value allows exploration of the effects of using a consistent threshold across all traits. Consistent threshold simplifies the discretization process and allows for direct comparisons. The problem with this approach is that it may not align with actual data distribution, potentially causing imbalanced class distribution.

The following figures illustrate the distribution of agreeableness and creativity traits in the AMIGOS dataset. For better visualization, labels were organized into bins. Each bin includes values in the range [x, x+1). For instance, a creativity score of 4.3 is placed in the bin labeled 4.

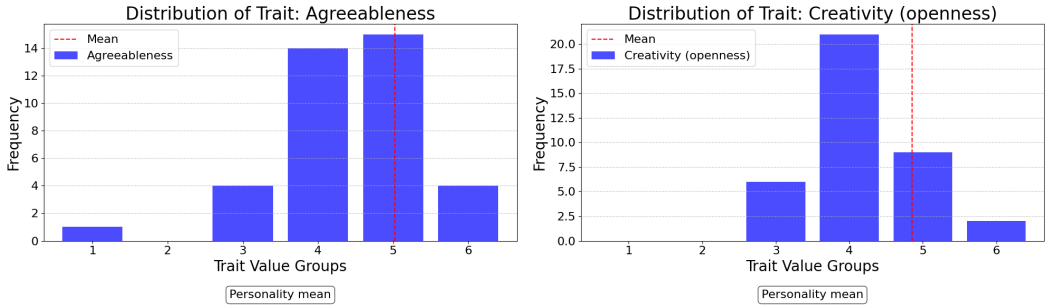


Figure 4.1. Distribution of labels for the agreeableness and creativity traits. In the personality mean approach, the mean value of each trait is computed. The values above the mean value are categorized as 1, while values equal to or below the mean value are categorized as 0.

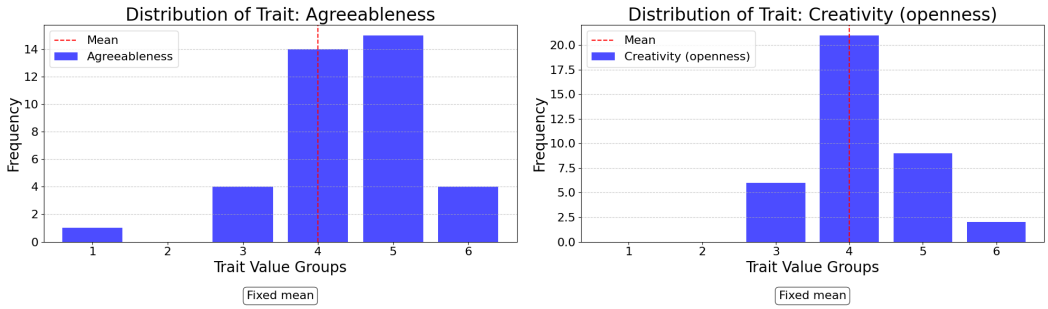


Figure 4.2. Distribution of labels for the agreeableness and creativity traits. In the fixed mean approach, values exceeding 4 are categorized as 1, while values 4 or below are categorized as 0.

4.1.2 Data cleaning and normalization

After uploading the data, it was preprocessed to prepare it for the training process. First of all, **some trials had missing or corrupted data** and this was a problem because it could interfere with the training process. To address this, a

function was implemented to discard such corrupted experiments. The function first examines each trial within a subject's dataset to identify the extent of corruption. **A trial is considered corrupted if the proportion of NaN values exceeds 90% of the total data points** within that trial, like in the Automated Removal of EKG Artifact From EEG approach [27] even though that approach used a lower threshold. For each subject, the function filters out the corrupted trials and retains only those with acceptable levels of NaNs. This is done by counting the number of NaNs in each trial and comparing it against a threshold (90% of the trial's total size). If a trial has fewer NaNs than the threshold, it is retained. Otherwise, **it is discarded**.

After cleaning the dataset from corrupted experiments, it was **scaled** and **normalized to a specific interval**. By standardizing the dataset, it was possible to improve the consistency and comparability of EEG signals across different studies and subjects. The normalization process is as follows:

- For each subject's EEG data, a function *normalize_data* initializes a scaler using MNE's Scaler with the configuration of the EEG channels (electrodes) and the sampling rate. This scaler is responsible for scaling the EEG signals. The function loops through each trial for the subject. Each trial's data is converted to a floating-point format and any NaN values are replaced with zeros using *np.nan_to_num*. This is to avoid errors with the scaler since it can't deal with this kind of value;

- The data is rearranged to have channels first *(c s)* instead of samples first *(s c)* to align with the expected format for MNE's functions. A bandpass filter is applied to the data to retain only the frequencies between 1 Hz and 50 Hz. This step helps remove unwanted noise and artifacts from the EEG data;

- After that, the filtered data is rearranged again to include a batch dimension *(() c s)* and then scaled using the initialized scaler. The scaler standardizes the data to have a mean of zero and a unit variance;

- The data is further normalized to the range of -1 to 1. This involves calculating the maximum and minimum values of the scaled data and adjusting it to fit within the desired range. A small constant (epsilon) equal to 1e-9 is added to the divisor to prevent division by zero;

- Finally, the normalized trial data replaces the original trial data in the subject's dataset.

This process has shown several benefits, such as the fact that the function improves the overall quality of EEG data, making it more suitable for analysis. Furthermore, data normalization ensures that different trials and subjects have comparable EEG signals, which is essential for model training and evaluation.

An example signal from the T8 and P7 channels of subject 1 in the AMIGOS dataset is provided, showcasing the raw EEG data before and after normalization.

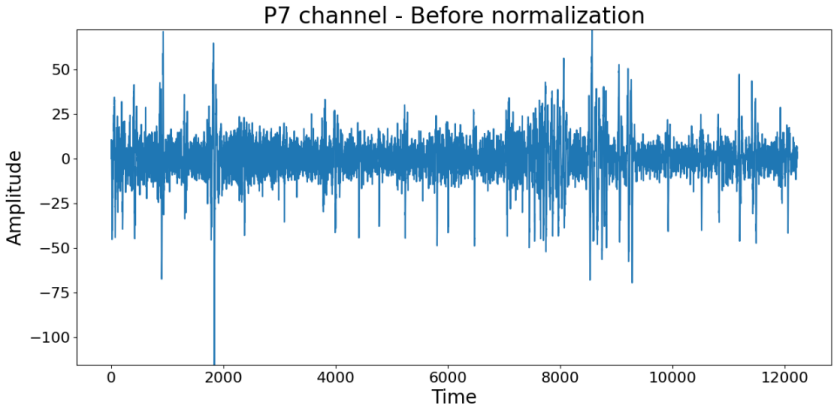


Figure 4.3. P7 channel signal of subject 1 in the AMIGOS dataset before normalization.

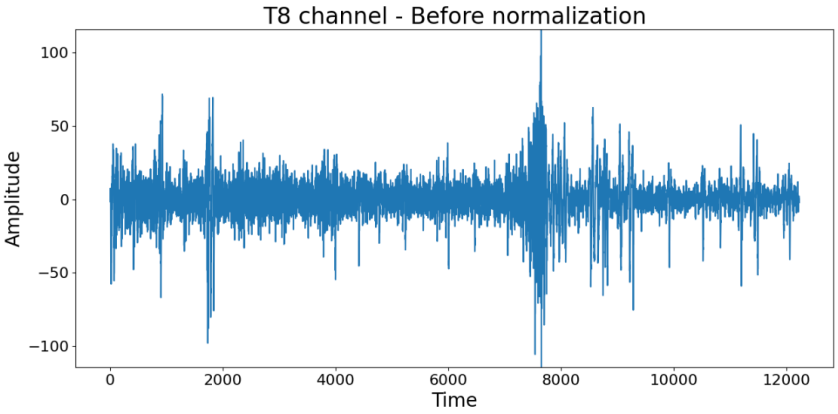


Figure 4.4. T8 channel signal of subject 1 in the AMIGOS dataset before normalization.

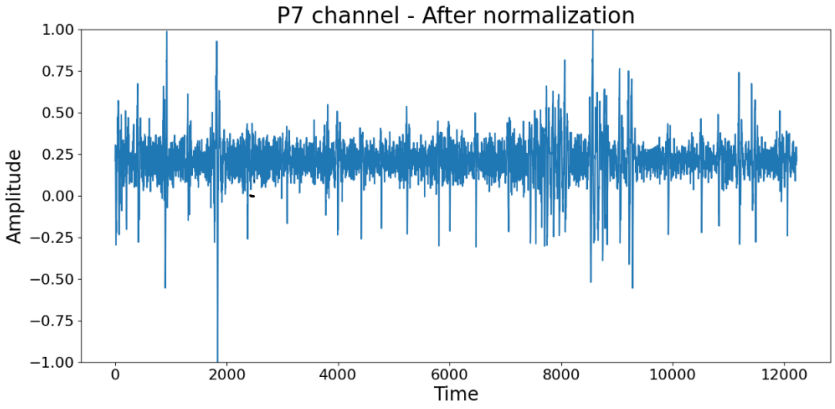


Figure 4.5. P7 channel signal of subject 1 in the AMIGOS dataset after normalization.

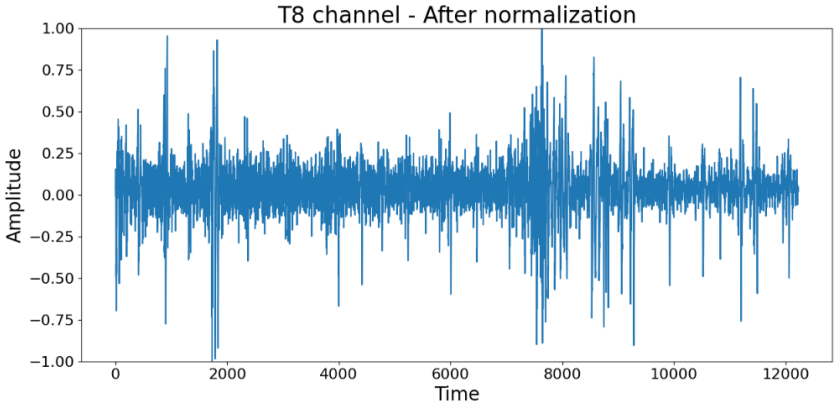


Figure 4.6. T8 channel signal of subject 1 in the AMIGOS dataset after normalization.

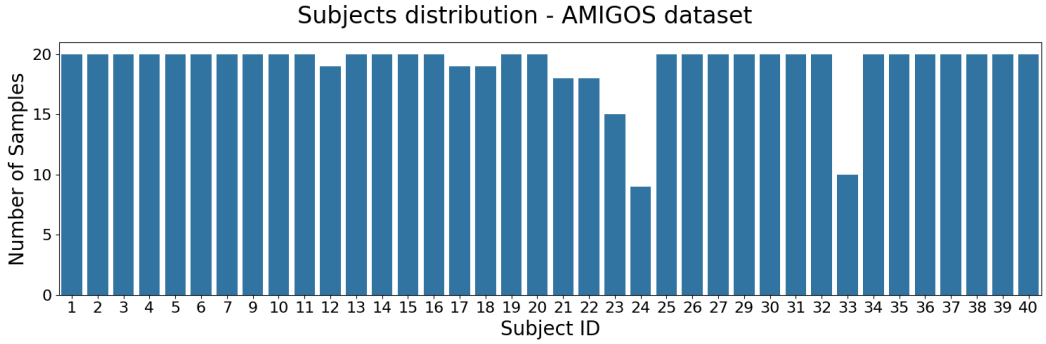


Figure 4.7. Distribution of samples across subjects in the AMIGOS dataset after data cleaning. Note that some experiments were discarded due to data corruption.

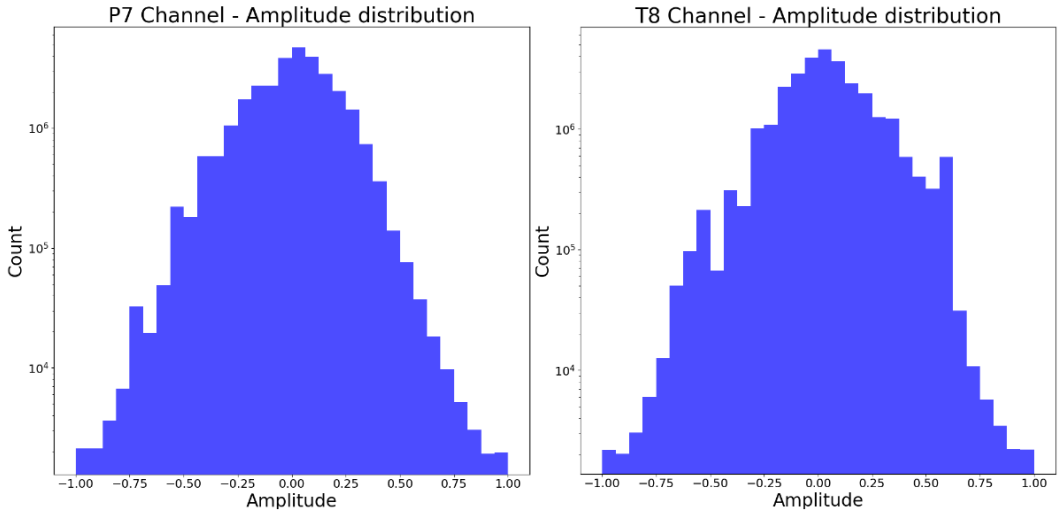


Figure 4.8. Example of the distribution of amplitude values recorded from the P7 and T8 channels of an EEG signal in the AMIGOS dataset. The x-axis represents the amplitude, and the y-axis shows the count of occurrences for each amplitude range.

Further preprocessing steps were applied to remove unreliable trials from the ASCERTAIN dataset. The authors of the dataset provided a document rating the reliability of each trial on a scale from 1 (perfect data) to 6 (missing data). To ensure the quality of the data used in our study, it was decided to remove all trials with a reliability level of 4 or above.

4.1.3 EEG windowing process

The next step was to **divide the EEG signals** into smaller, more manageable segments called **windows**. Dividing EEG data into smaller windows enables more granular analysis and processing, which is essential for time-series data. In particular, windowing facilitates the analysis of temporal dynamics in EEG signals by breaking continuous data into manageable chunks. The windowed data is then used as input to deep learning models, allowing them to learn patterns within smaller, more coherent segments of EEG data. The candidate opted to apply the same process used in the EEGNet method [42] that trained a model using EEG windows instead of feeding the entire signal. The windowing process works as follows:

- An empty *windows* list is initialized to store the resulting windows of EEG data. The function iterates through the EEG data which contains multiple trials for each subject;
- For each trial, the function divides the data into overlapping or non-overlapping windows based on *samples_per_stride* (the step size for creating the windows) and *samples_per_window* (the window size). Both parameters are represented in seconds and are calculated by multiplying the chosen window size and the window step by the sampling rate. A value of 3 was decided for both parameters, which is the value used in EEGNet [42]. This means that each window lasts 3 seconds and that they are not overlapping. This allows the candidate to compare the results of this work with theirs;
- The start *(window_start)* and end *(window_end)* of each window are determined from time to time using the window size and stride and iterating over the EEG signals. If *window_end* exceeds the length of the trial, it is adjusted to match the trial's length to avoid out-of-bounds errors;
- The segment of EEG data corresponding to the window *(window_eeg)* is extracted from the trial. If the extracted window's length is less than *samples_per_window*, the function handles it in two ways based on a hyperparameter *drop_last*:
  - **Drop the window**: If *drop_last* is True, the window is discarded;
  - **Zero-Pad the window**: If *drop_last* is False, the window is padded with zeros to reach the desired length.

  After extensive testing, it was opted for the second approach, consistent with similar works in the field. This decision was made because the impact on performance is minimal, as this adjustment is only applied to the last window of an EEG signal, and such incomplete windows are relatively few compared to the total number of windows. Zero-padding ensures that all windows maintain uniform size, which is essential for consistent processing and accurate modeling in subsequent analysis.

- Each window is stored in a dictionary containing the windowed segment of EEG data, the ID of the subject from whom the data was recorded, and the labels associated with the subject.

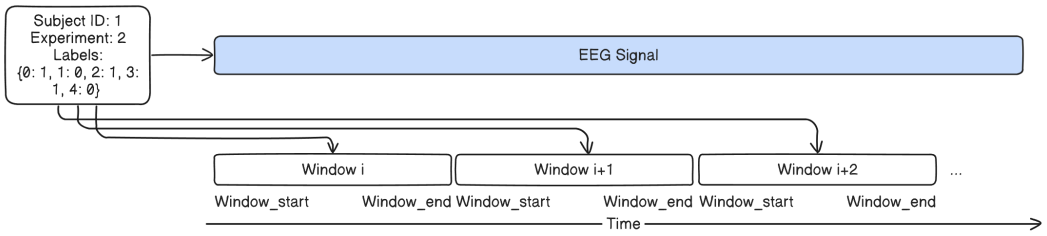


Figure 4.9. Each EEG signal is associated with a subject, their corresponding personality label, and the specific experiment conducted. During the windowing process, the EEG signal is divided into non-overlapping 3-second windows. Each resulting window inherits the metadata of the original EEG signal, including the subject, personality label, and experiment details. Additionally, each window is annotated with *window_start* and *window_end* markers, indicating the start and end points of the window in terms of sample numbers.

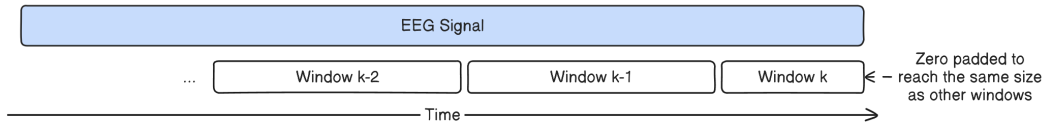


Figure 4.10. Since EEG signals can vary in length, the last window may sometimes be shorter than the others. In such cases, it is zero-padded to match the length of the other windows, ensuring consistency.

4.1.4 Mel spectrograms

Once the EEG signals were divided into windows, **Mel spectrograms** were calculated. The mel spectrogram is a type of time-frequency representation commonly used in signal processing. **Contrary to other works previously analyzed** that use EEG signals as input of the models, **this work tries to extract further features by applying the mel spectrogram to the EEG windows previously computed**. Mel spectrograms provide a time-frequency representation of EEG signals, capturing how the power of different frequency components evolves over time. This is crucial for understanding the dynamic nature of EEG data, where brain activities can vary significantly over short time periods. The Mel scale is designed to approximate human auditory perception, emphasizing frequencies to which humans are more sensitive. This can help in extracting features that are more relevant to the perceptual and cognitive processes reflected in EEG signals, potentially leading to more effective feature representations for tasks like personality estimation. It is also necessary to consider that the 2D representation of mel spectrograms aligns well with visual pattern recognition techniques and machine learning algorithms, especially convolutional neural networks (CNNs), which make it easier to work with common deep learning architectures such as Transformers. The computation of the mel spectrograms works as follows:

- A function iterates over each window, computing its mel spectrogram. For

each window, it calls a spectrogram module to compute the Mel spectrogram of the EEG data;

- After processing all windows, the function returns the updated list of windows, each containing its mel spectrogram.

The **spectrogram module** is responsible for converting EEG data into mel spectrograms using PyTorch and the torchaudio library. This class encapsulates the configuration and computation logic required for this transformation and it works as follows:

- First of all, it sets the sampling rate, minimum, and maximum frequencies for the mel spectrogram, the number of Mel bands, and other parameters like window size and stride, which define the resolution of the spectrogram;
- It configures the mel spectrogram transformation using the torchaudio library. The parameters used for configuration are the ones previously defined;
- It computes the mel spectrogram. Several rearrangements to the dimensions are applied from time to time to match the expected formats.

The resulting data is finally ready for the training process.

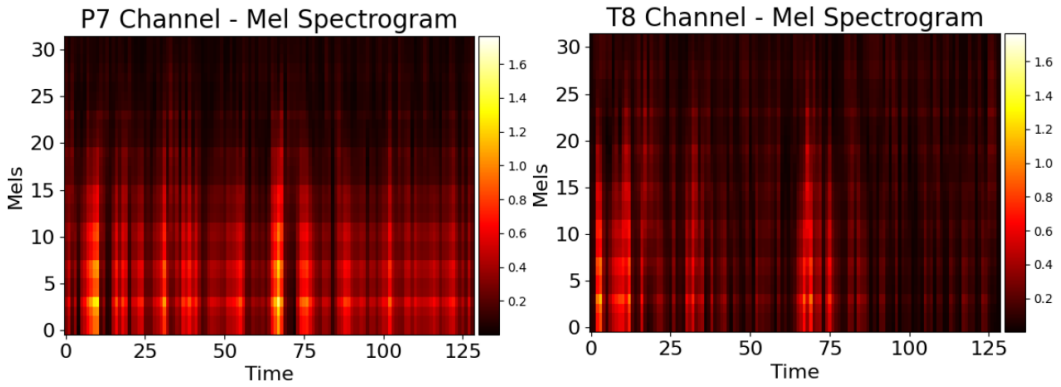


Figure 4.11. Example of Mel spectrograms for the P7 and T8 channels from a window of a subject in the AMIGOS dataset. In this example, time represents the number of samples.

4.1.5 Data augmentation

To enhance model robustness, data augmentation is applied to the training set. Augmentation techniques introduce variability into the training data, helping models learn more generalized features and reducing overfitting. Data augmentation is applied optionally based on the *apply_augmentation* hyperparameter. If set to True, it is also possible to control what augmentation techniques are used. A function iterates through each spectrogram and applies the specified augmentation methods. The methods that can be applied are:

- **Spec augment**: It applies time and frequency masking to the mel spectrogram, which is a common data augmentation technique in audio processing. This method helps the model become invariant to slight distortions in time and frequency. To control how the mask is applied, two hyperparameters are

used: *time_mask_param* regulates the maximum width for time masking, and *freq_mask_param* controls the maximum width for frequency masking.

- **Add noise**: It introduces random noise to the mel spectrogram, which helps in making the model robust to minor variations and noise present in the real data.
- **Flip**: Horizontally flips the mel spectrogram along the time axis, which simulates time reversal.

4.2 EEGTransformer

To address the problem, the candidate opted for an architecture not previously employed in related works about EEG personality estimation: the **Transformer**. This type of architecture, previously explained in Section 2.4, was adapted to process EEG signals represented as Mel spectrograms.

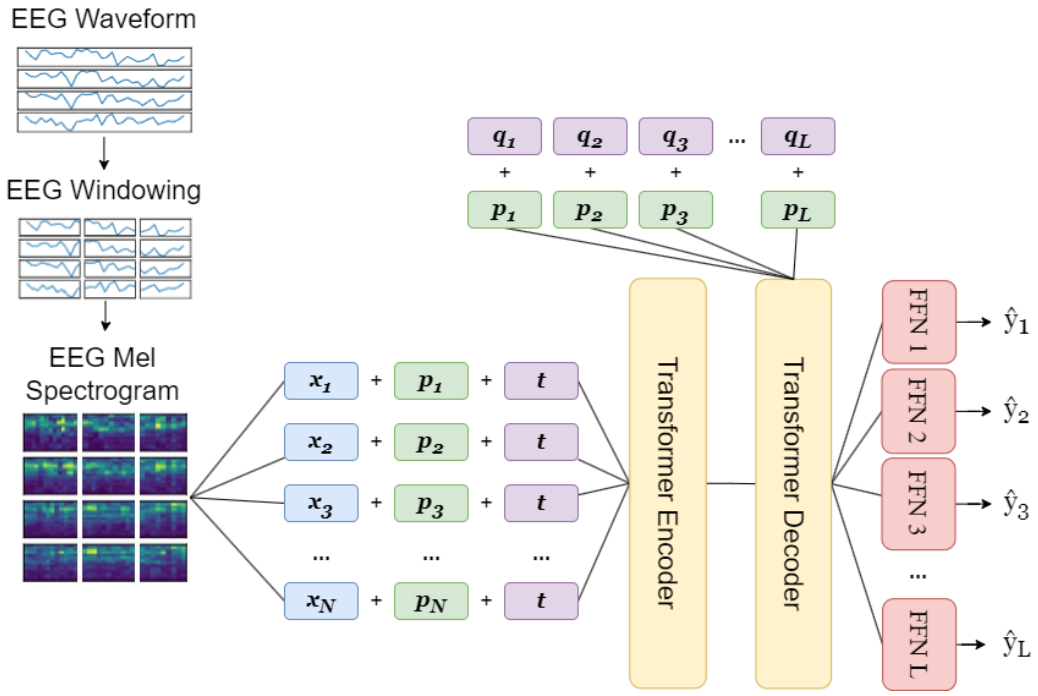


Figure 4.12. The overall architecture of the proposed model where x is the input, p is the positional encoding, t is the learnable token, q is the query, and ŷ is the output. The different components are described in Section 4.2.1.

4.2.1 Model general structure

The encoder and decoder modules were initialized as follows:

- The encoder is configured with specified layers (*num_encoders*), attention heads (*n_heads*), and dimensions (*d_model*) defined as hyperparameters. Here, *d_model* represents the dimension of the input and output vectors of the transformer model, also referred to as the hidden size. This dimension determines the size of the query, key, and value vectors in the self-attention

mechanism. The dimension of the feed-forward network (*dim_feedforward*) within each Transformer layer is set to four times *d_model* to facilitate richer intermediate representations. This multiplier is a common practice to balance model complexity and computational efficiency.

- A dropout layer is included within the transformer to enhance the model's generalization and mitigate overfitting.
- Regarding the activation function, ReLU was chosen for its high efficiency. Furthermore, EEG signals can be noisy, and ReLU's ability to zero out negative values can help in reducing certain types of noise. This enables the network to concentrate on significant positive signals while disregarding minor negative fluctuations.

The decoder is instantiated only if the *use_encoder_only* flag is set to **false**. Its configuration closely mirrors that of the encoder. During the forward pass, the decoder receives the encoder's output and a **label embedding** which transforms categorical labels into dense vectors of *hidden_size* dimensions. This process converts labels into dense representations that can be processed by the decoder, allowing it to incorporate label information effectively into the decoding process. By embedding labels, the decoder can understand and integrate label information directly, enabling the generation of outputs that are conditioned on these labels.

4.2.2 Mels merging process

Another crucial step involves preparing the data for the transformer network. To this end, the candidate developed a **MergeMels** module specifically designed to preprocess mel spectrogram data for use in a transformer model. This module employs a convolutional layer to consolidate the frequency bands of mel spectrograms into a more compact representation, suitable for input to a transformer encoder or decoder. This preprocessing reduces the dimensionality of the input while capturing essential features from the mel spectrograms, enhancing their suitability for subsequent processing by a transformer model. The candidate explored different preprocessing methods:

- **Channels**: This method merges the mels and samples while keeping the number of channels unchanged;
- **Samples**: This method merges the mels and channels while keeping the number of samples unchanged.

The data structures for these merging methods are *(b c (m s))* and *(b s (m c))* respectively, where b is the batch size, s is the number of samples, m is the number of mels in a spectrogram, and c is the number of channels or electrodes. Given that the number of samples is significantly higher than the number of channels, the channels method can accelerate the training process but may reduce the complexity of the features captured.

4.2.3 Implementation of a learnable token

The candidate opted to utilize a **learnable token** in the training process. A learnable token is an additional token appended to the input sequence, designed to aggregate information from the entire sequence, aiding in the final classification. In transformer architectures, the learnable token operates through self-attention

mechanisms, enabling it to attend to every other token in the input. This interaction allows the learnable token to capture a summary of the input sequence. By appending this token to the sequence, the transformer can learn to focus attention on it and utilize it for classification decisions. The output representation of the learnable token, after passing through the transformer layers, is then used for the final classification task. The learnable token in this work is used as follows:

- **Initialization**: The learnable token is initialized as an embedding with a size matching the model's hidden size. This embedding starts with random values and it is learned during training;
- **Forward Pass**: During the forward pass, the learnable token is repeated to align with the batch size and it is prepended to the input sequence;
- **Processing**: The concatenated sequence, now including the learnable token, is passed through the transformer encoder (and optionally the decoder). The transformer processes this entire sequence, allowing the learnable token to attend to all other tokens via the self-attention mechanism, thereby summarizing the information from the sequence;
- **Classification**: After processing through the transformer, the output associated with the learnable token is extracted and used for the final classification task.

4.2.4 Positional encoding

For the positional encoding, two methods are used:

- **Sinusoidal positional encoding**: It uses trigonometric functions to generate fixed positional embeddings. A function computes positions and division terms for sine and cosine functions. It then populates *pe* with sine and cosine values based on position and division terms, where *pe* is a vector initialized as zeros with dimensions *(sequence_length, embeddings_dim)*. Later, it repeats *pe* across the batch dimension. The result is the tensor of learnable positional embeddings.
- **Learnable positional encoding**: It uses a learnable embedding layer to generate position-specific embeddings. It takes *hidden_size* and *max_position_embeddings* as parameters that are used to initialize an embedding. It then uses the embedder to fetch embeddings for positions up to the second dimension of the input x which can be the number of channels or samples depending on the merge mels method used. Later, it repeats embeddings across the batch dimension. The result is the tensor of learnable positional embeddings.

4.2.5 Proposed architecture

To sum up, the proposed architecture processes input data through several steps:

1. **Label embedding preparation** (conditional): If *use_encoder_only* is False, the model prepares dense embeddings *label_tokens* for categorical labels. These embeddings are expanded to match the batch size and are used to incorporate label information during decoding.

2. **Mel spectrogram processing**: The input mel spectrograms x undergo preprocessing via *merge_mels*, which consolidates frequency bands according to a specified typology.

3. **Positional encoding addition**: If *positional_encoding* is provided, it adds positional information to both the input x and the label embeddings *label_tokens*, enhancing the model's ability to understand sequence order.

4. **Learnable token inclusion**: If *use_learnable_token* is True, a learnable token *cls_token* is added to the input sequence. This token, expanded to match the batch size, allows the transformer to aggregate information across the entire sequence during processing.

5. **Transformer encoding**: The processed input (x with added tokens) is passed through the transformer encoder. This component applies self-attention mechanisms to capture dependencies across the input sequence.

6. **Decoder usage** (conditional): If *use_encoder_only* is False, the model also utilizes the transformer decoder. This component generates output conditioned on both the encoded input x and the label embeddings *label_tokens*.

7. **Final classification**: The final classification layer in this neural network architecture is a straightforward fully connected layer. It accepts inputs of size *hidden_size* and produces outputs equal to the number of labels in the classification task. The output from either the encoder or the decoder is fed into a fully connected layer based on the *use_encoder_only* flag. Before this layer, a dropout layer is applied to improve generalization by randomly ignoring some neurons during training, thus preventing the model from relying too heavily on specific features and reducing overfitting. This setup ensures that the network can effectively map the extracted features to the correct output labels, making it suitable for multi-label classification tasks.

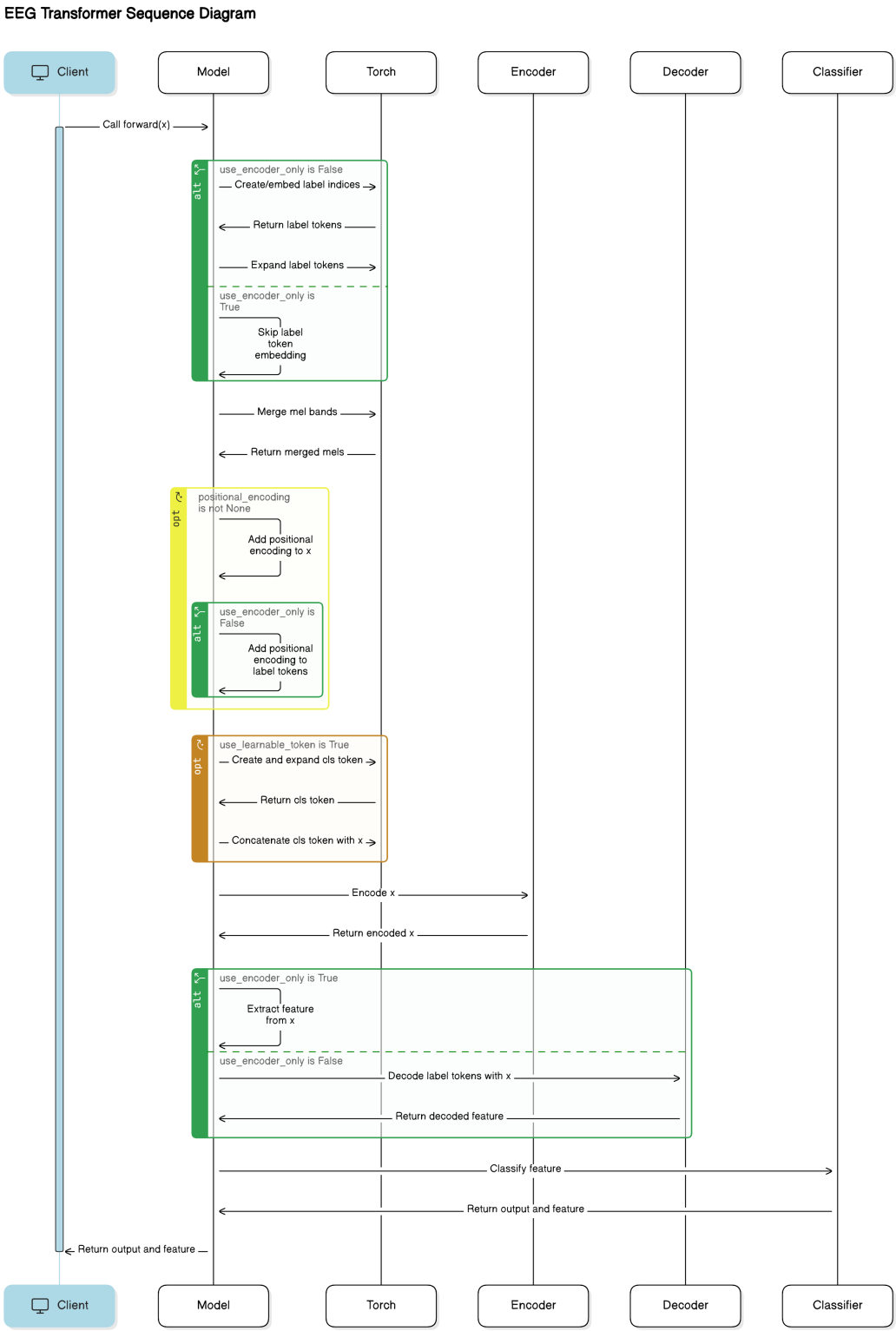


Figure 4.13. Sequence diagram that explains the flow of operations performed by the proposed EEG Transformer model.

Chapter 5

Experiments

This section is organized as follows:

- Section 5.1 provides a summary of the commonly used datasets from the literature that will be tested in this study;
- Section 5.2 describes how this work was implemented;
- Section 5.3 details the Python version, key libraries, training parameters, and the system configuration used for the experiments;
- Section 5.4 presents the results of the proposed method and compares them with the state-of-the-art, along with insights into the findings;
- Section 5.5 discusses the ablation study conducted to identify the optimal hyperparameters for the final evaluations.

5.1 Datasets

In this work, the AMIGOS and ASCERTAIN datasets were chosen for their comprehensive scope and relevance. Both datasets are widely recognized in affective computing research, providing large sample sizes, EEG recordings, and detailed personality labels. Their usage in related works presented in Section 3.1 facilitates the comparison of the results.

5.1.1 AMIGOS

The **AMIGOS dataset** (A Dataset for Affect, Personality, and Mood Research on Individuals and Groups) [19] is a dataset used in affective computing and EEG-based emotion recognition research. It focuses on capturing the emotional responses of individuals and groups while watching video stimuli. EEG data from 40 participants (20 males and 20 females) aged between 21 and 40 years was captured using a device called EPOC+. Each piece of data consisted of 14 EEG channels (AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4). The dataset also contains peripheral physiological signals such as ECG (Electrocardiography), GSR (Galvanic Skin Response), and respiration rate as well as videos capturing facial expressions during the sessions. Participants were shown a series of video clips and their physiological and EEG responses were recorded. The dataset includes individual sessions and group sessions. Each video clip's emotional impact was rated by participants for valence (positive/negative) and arousal (calm/excited). Based

on that, participants made self-reports on emotional states and personality traits, including Big Five personality scores and mood questionnaires from which labels for training deep learning models were extracted. Regarding the AMIGOS data structure, each file provided belongs to a single subject and contains raw EEG signals for each trial.

5.1.2 ASCERTAIN

The **ASCERTAIN dataset** (Affect and Personality Study with EEG Recordings and Video Stimuli) [32] is another significant dataset used in the study of affective computing, emotion recognition, and personality analysis through EEG and other physiological signals. ASCERTAIN is designed to investigate the relationship between affective states, personality traits, and physiological responses. It's widely used in studies focusing on emotion recognition and the interaction between personality and affect. EEG data from 58 participants (32 males, 26 females), aged 18-35 years was captured using the BioSemi ActiveTwo system. In this case, the raw data consisted of 8 EEG channels. The dataset also contains peripheral physiological signals such as ECG (Electrocardiography), GSR (Galvanic Skin Response), and PPG (Photoplethysmography) as well as facial video recordings capturing participants' expressions during the experiment. Participants viewed 36 video clips while their physiological and EEG responses were recorded. Each video clip was rated for its emotional impact on valence and arousal scales. The dataset includes metadata for each trial, including emotional ratings and associated personality traits of the subjects. The ASCERTAIN data structure consists of 36 files for each subject that contain the EEG signal of the associated trial.

5.2 Implementation

The code is hosted in a repo that can be accessed at `https://github.com/AlessioLucciola/eeg-personality-estimation`. It is written in Python using the PyTorch framework, enabling efficient parallelization and GPU acceleration. All dependencies are listed in the *requirements.txt* file. The repository is organized as follows:

- **dataloaders**: This directory contains custom PyTorch data loaders designed for splitting the data based on the chosen validation scheme and applying data augmentation to the training set;
- **datasets**: Here, custom PyTorch datasets, *AMIGOS_dataset* and *ASCERTAIN_dataset*, are implemented to load and standardize data across their respective datasets. These inherit from the *EEG_classification_dataset* module, which is responsible for data preprocessing;
- **models** and **train_modules**: The **models** directory defines the structure of various models, while **train_modules** includes the components necessary to initiate the training process. To train a model, execute the relevant script in **train_modules** (e.g., *python -m train_modules.eeg_transformer*);
- **utils** and **shared**: These directories contain utility functions and constants used throughout the application;
- **data**: This directory stores the datasets along with their associated metadata, including labels;

- **plots**: Contains functions for generating plots;
- **results**: This folder is designated for storing both checkpoints and metrics generated by each model. When a model training session starts, a new subfolder is created under **results** corresponding to the test. Several *.json* files are generated within this subfolder: *configuration.json* (storing the training hyperparameters), *train_val_results.json* (containing metric values for each epoch), and *fold_results.json* (providing metrics for each fold in k-fold and LOO validation schemes). Model checkpoints for each epoch are saved under *results/test_name/models*;
- **config.py**: This file allows users to define all training hyperparameters, enabling comprehensive control over experiments without modifying the core code. This setup facilitates rapid testing of various configurations.

Each *train_module* utilizes either *train_loop_split* or *train_loop_kfold_loo* based on the validation scheme selected in the *config.py* file. There are several possible validation schemes as explained in Section 5.4.1. These files contain the training loops necessary for developing the personality estimation model. Both include loops that update the model's weights during each epoch and for each batch in the training phase. In contrast, during the validation phase, performance metrics are evaluated without updating the weights. Metrics such as loss, accuracy, recall, precision, and F1 score (using the *micro* averaging method) are assessed in each epoch and stored in the *results* directory.

The primary distinction between the two files lies in their handling of folds. The *train_loop_kfold_loo* file replicates the training loop for each fold (in the case of k-fold cross-validation) or for each subject (in leave-one-out cross-validation). The result of a fold is determined by the epoch that achieves the highest accuracy within that fold. The final results are computed by averaging the best epoch results from each fold.

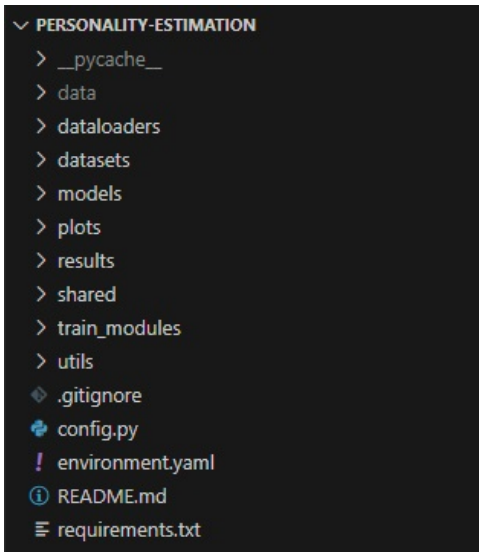


Figure 5.1. Project structure

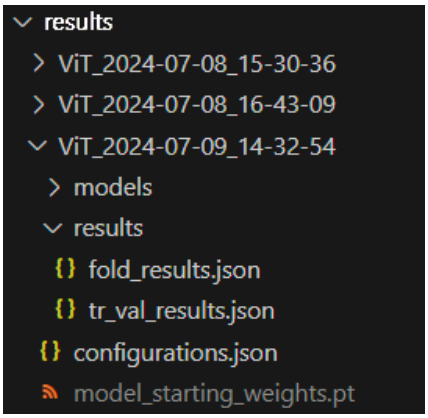


Figure 5.2. Results folder structure

5.3 Experimental setup

The code was developed using Python 3.10.6 and PyTorch 2.3.1. As detailed in Section 5.2, various libraries were utilized to support the implementation. The primary libraries include TorchVision 0.15.1, TorchAudio 2.3.1, MNE 1.6.1, NumPy 1.26.4, Einops 1.7.0, Pandas 2.2.1, and TorchMetrics 1.3.2. Additional dependencies and their respective versions can be found in the *requirements.txt* file.

In the final tests, the configuration employed was:

- **AdamW** optimizer [1] with a learning rate of 5e-5 for the AMIGOS dataset and 1e-4 for the ASCERTAIN dataset, and a weight decay of 0.1. AMIGOS, being a larger dataset than ASCERTAIN, requires a smaller learning rate to allow for more precise updates and smoother convergence. This prevents the model from overshooting the optimal parameters, as it encounters a broader variety of examples. In contrast, ASCERTAIN's smaller dataset uses a larger learning rate (1e-4) to make significant progress and avoid underfitting. AdamW is a variant of the Adam optimizer that decouples the weight decay from the gradient update, addressing some of the limitations of Adam. This approach enhances regularization, reduces overfitting, and often accelerates convergence, making it particularly effective when used with Transformers;
- **BCEWithLogitsLoss** as the loss function. This criterion merges Binary Cross-Entropy (BCE) loss with a sigmoid activation, ensuring numerical stability and efficiency for binary classification tasks. It was ideal for this multi-label classification scenario, where each label is an independent binary decision. The PyTorch implementation was slightly customized to incorporate an optional label smoothing factor, which is not natively supported (unlike CrossEntropyLoss which has this feature). Various smoothing factors were experimented with to mitigate overfitting, but ultimately, this feature was excluded from the final tests as discussed in Section 5.5;
- **CosineAnnealingLR** as the learning rate scheduler.

The parameters, detailed in Section 5.5, were refined through an ablation study and applied consistently across both datasets after determining the optimal configuration.

The final tests were performed on a system with this configuration:

- CPU: Intel(R) Core(TM) i5-10600 CPU (12 CPUs), 3.3GHz
- RAM: 32GB
- GPU: NVIDIA GeForce RTX 4070, 12GB

5.4 Results

5.4.1 Validation schemes

Since the related works use different approaches for testing the performance of their models, it is necessary to use several methods to compare their results with the ones obtained by this work. Therefore, this work uses three different **validation schemes**:

- **Split method**: This is a simple Train-Test Split. The dataset is split into training and validation sets. 80% of the dataset is put in the training set and the remaining is used for validation and testing purposes. Training and testing data loaders are created and returned as a tuple;

- **K-Fold method**: K-Fold Cross Validation splits the dataset into k subsets (folds). Each fold is used once as a validation set while the remaining folds form the training set. This method provides a more generalizable performance metric by averaging the results across folds. Training and validation dataloaders for each fold are created and stored in a dictionary.
- **Leave-one-out (LOO) method**: LOOCV is a rigorous validation method where each subject's data is used as a test set while the remaining subjects' data form the training set. Subject IDs are used to iterate through the dataset. For each subject, their data is isolated as the validation set, and the rest is used for training. For each subject, a pair of training and validation dataloaders is created and stored in a dictionary.

In Section 5.4.2, the results achieved by this study are compared with those obtained by related works, as presented in Section 3.1.

5.4.2 Comparison with the state of the art

This section presents a comparison with state-of-the-art works addressing personality estimation. Table 5.1 compares the performance of the proposed method with related works on the AMIGOS dataset, while Table 5.2 provides a similar comparison for the ASCERTAIN dataset.

Method	Validation scheme	Discretiz. method	Windows size (s)	Windows stride (s)	Precision (%) ↑	Recall (%) ↑	Accuracy (%) ↑	F_1 (%) ↑
EEGNet[1] [42]	5-fold CV	Pers. mean	3	3	-	-	<u>90.58</u>	<u>90.74</u>
SVM[2] [25]			-	-	<u>83.80</u>	<u>66.80</u>	82.18	-
EEGTr. (Ours)			3	3	**98.93** ± 0.28	**98.26** ± 0.30	**98.58** ± 0.11	**98.60** ± 0.11
EEGTr. (Ours)		Fixed Mean	3	3	99.45 ± 0.09	99.56 ± 0.12	99.29 ± 0.07	99.51 ± 0.05
BSHGP[3] [13]	10-fold CV	-	-	-	-	-	80.42	-
EEGTr. (Ours)		Pers. mean	3	3	99.06 ± 0.16	98.62 ± 0.29	98.83 ± 0.11	98.84 ± 0.11
EEGTr. (Ours)		Fixed mean	3	3	99.47 ± 0.10	99.56 ± 0.07	99.30 ± 0.08	99.52 ± 0.06
SVM[4] [19]	LOOCV	Pers. mean	-	-	-	-	-	<u>38.79</u>
EEGTr. (Ours)			3	3	58.65 ± 30.62	50.80 ± 22.26	61.82 ± 9.98	**52.02** ± 23.13
EEGTr. (Ours)		Fixed mean	3	3	82.23 ± 24.47	85.89 ± 16.80	77.04 ± 15.16	81.72 ± 18.69

Table 5.1. Results of the performances of the proposed method on the AMIGOS dataset. For EEGTransformer both the mean and the standard deviation are reported.

Regarding the AMIGOS dataset, previous studies have employed various validation schemes, including 5-fold cross-validation (CV), 10-fold CV, and leave-one-out (LOO) validation. To ensure a fair comparison of results, the same validation schemes were adopted in this work. Despite the abundance of similar studies, comparing their results is challenging due to the differing methodologies and often vague descriptions of EEG data preprocessing.

The only study that employs an approach similar to ours is EEGNet [42], which uses the EEG windowing process detailed in Section 4.1.3. To facilitate a direct comparison with EEGNet, the same *window_size* and *window_stride* parameters were utilized. Additionally, this study aims to benchmark the proposed method against all other relevant works to evaluate its effectiveness comprehensively.

[1]The average of the metrics obtained for each label is considered under the D3 preprocessing setting, which achieved the best results.

[2]The average of the metrics obtained for each label in the fusion scheme is considered

[3]The average values of each fold are considered

[4]The average of the metrics obtained for each label in every video is considered

As evident from Table 5.1, our approach surpasses the performance achieved by other studies using K-Fold cross-validation. Despite the widespread use of this validation scheme for evaluating model performance in related works, it may not be the most appropriate method. The division of EEG signals into windows can lead to similar windows from the same subject and experiment appearing in both the training and validation sets. This can artificially inflate validation performance as the model may learn to recognize specific windows rather than underlying patterns in the data, thereby reducing generalization.

Even if the EEG windowing process is avoided by providing the model with entire EEG signals, the problem persists since signals from different trials might still be present in both the training and validation sets. Nonetheless, our approach demonstrates superior performance compared to other works utilizing the same validation method.

A possible solution to obtain more reliable results is to use Leave-One-Out Cross-Validation, where training is conducted using all subjects except one, which is reserved for validation/testing. This ensures that windows from the same subject are exclusively in either the training or validation set, allowing for a more accurate assessment of the model's ability to generalize to new subjects. This method is only employed by the authors of the AMIGOS dataset, who used an SVM model for personality estimation. Our approach, however, demonstrates significantly better performance, surpassing their results by approximately 29%, thereby highlighting the effectiveness of our proposed method.

Since most related works use the personality mean approach to discretize the labels, the results were initially compared using this method. Subsequently, the candidate explored the impact of using a fixed mean for all labels, as described in Section 4.1.1, instead of computing the mean for each label individually. This approach significantly improved the model's performance. The improvement is particularly evident when using Leave-One-Out Cross-Validation, where the F1 score increased from 52.02% to 81.72%, enhancing the performance of the SVM model [19] by approximately 110%.

Even though using a personality mean can have several benefits as explained in Section 4.1.1, the increased performance with a fixed mean might be due to reduced variability and noise in the labels, making it easier for the model to learn distinct patterns. Additionally, a uniform threshold could enhance the model's ability to generalize across different subjects, as it avoids overfitting specific label distributions in the training data.

Method	Validation scheme	Discretiz. method	Windows size (s)	Windows stride (s)	Precision (%) ↑	Recall (%) ↑	Accuracy (%) ↑	F_1 (%) ↑
DeepLSTM [14]	10-fold CV	-	-	-	93.48	94.72	94.16	93.68
BSHGP[5] [13]					-	-	76.48	-
EEGTr. (Ours)		Pers. mean	3	3	69.87 ± 1.98	42.53 ± 3.36	60.14 ± 1.49	52.78 ± 2.60
EEGTr. (Ours)		Fixed mean	3	3	89.98 ± 0.54	90.14 ± 0.84	83.56 ± 0.51	89.98 ± 8.84
SVM[6] [32]	LOOCV	Pers. mean	-	-	-	-	-	43.40
EEGTr. (Ours)			3	3	63.97 ± 28.46	36.30 ± 17.83	55.99 ± 18.84	**44.00** ± 18.86
EEGTr. (Ours)		Fixed mean	3	3	88.92 ± 16.24	92.46 ± 7.78	84.67 ± 11.90	89.39 ± 10.23

Table 5.2. Results of the performances of the proposed method on the ASCERTAIN dataset. For EEGTransformer both the mean and the standard deviation are reported.

[5]The average values of each fold are considered

[6]The average of the metrics obtained with linear SVM for each label in every video is considered

The ASCERTAIN dataset was evaluated using various validation schemes, including 10-fold cross-validation (CV) and Leave-One-Out Cross-Validation (LOOCV), to ensure a fair comparison with previous studies. Notably, similar to the approach taken with the AMIGOS dataset, our method was benchmarked against other relevant works to comprehensively evaluate its effectiveness. As shown in Table 5.2, the personality mean and fixed mean approaches had a significant impact on the performance metrics, including precision, recall, accuracy, and F1 score.

When employing the 10-fold CV scheme, the EEGTransformer using the personality mean approach achieved moderate performance, with an F1 score of 52.78%. This performance, while respectable, is significantly lower compared to the 93.68% F1 score reported by the DeepLSTM model [14]. This discrepancy can be attributed to differences in methodologies used by the DeepLSTM model, including unspecified discretization methods and preprocessing steps for the EEG data. The lack of detailed methodological information and publicly available code for DeepLSTM makes it challenging to fully understand and replicate their results, thereby questioning the reliability and comparability of their reported performance. However, the adoption of a fixed mean for label discretization in our method resulted in a substantial improvement, achieving an F1 score of 89.98%. This remarkable enhancement indicates that a fixed mean approach might reduce variability and noise in the labels, making it easier for the model to learn distinct patterns. Moreover, a uniform threshold could aid the model in generalizing across different subjects, thus avoiding overfitting specific label distributions in the training data.

LOOCV provides a more rigorous test of the model's generalization ability, as it ensures that windows from the same subject are not split between the training and validation sets. Using the personality mean approach under LOOCV, our model achieved an F1 score of 44.00%, which is slightly higher than the 43.40% F1 score obtained by the SVM model [32] employed by the dataset's authors. This shows that our method performs comparably to traditional machine learning approaches even in stringent validation settings.

The most significant improvement was observed when using the fixed mean approach under LOOCV, where the F1 score soared to 89.39%. This improvement underscores the effectiveness of the fixed mean method in enhancing the model's ability to generalize to new subjects. The fixed mean approach likely provides a more stable and consistent labeling method, thereby enabling the model to focus on learning underlying patterns in the EEG data rather than being confounded by label variability.

5.5 Ablation study

To identify the optimal hyperparameters for training our models, we conducted an **ablation study**. This method allows us to systematically assess the contribution of different components and configurations within our model. By analyzing these variations, we can determine which elements are crucial and which may be redundant or less effective. Starting from a baseline model that employs the lowest parameter values, we iteratively adjusted each hyperparameter to observe changes in performance. The study evaluates these modifications based on accuracy and considers both the number of parameters and the computational operations required by each model to gauge complexity. This evaluation was performed using the AMIGOS dataset and the LOO validation scheme. The tested hyperparameters are:

- Sampling rate: The frequency at which EEG data is sampled. Higher rates capture more detail, potentially improving model performance by providing a

richer temporal resolution;

- Mels: The number of mel bands used in the mel spectrogram. Increasing this number captures more frequency details, which can enhance feature extraction from the EEG data;
- Mels window size: The length of each window (in seconds) used to calculate the mel spectrogram. Smaller windows provide finer temporal resolution but may increase computational complexity.
- Mels window stride: The step size between successive windows. A stride equal to half the window size provides overlap, improving the continuity of captured features;
- Learning rate: It controls the speed at which the model learns. Lower values slow down learning, potentially leading to more stable convergence but requiring more epochs;
- Regularization: It adds a penalty to the model to prevent overfitting. Higher regularization values increase this penalty, which can improve generalization but might underfit if too high;
- Dropout: It randomly sets a fraction of input units to zero during training to prevent overfitting. Higher dropout rates increase this effect, improving generalization but potentially harming model capacity;
- Label smoothing epsilon: A technique that softens the true labels during training to prevent the model from becoming overly confident. It can enhance the robustness of noise in the labels;
- Attention heads: The number of attention heads in the transformer. More heads allow the model to capture different aspects of the data by attending to different parts of the input;
- Encoders: The number of transformer encoder layers. More layers can capture more complex relationships in the data but increase computational requirements;
- Decoders: The number of transformer decoder layers. More layers provide a deeper decoding process but add complexity;
- Hidden size: The dimensionality of the hidden layers and the attention mechanism. Larger sizes can capture more detail but increase the number of parameters and computational cost;
- Merge mels typology: It determines how the mel spectrogram data is merged. *samples* keeps the sample dimension intact while *channels* retains channel information, affecting how features are aggregated;
- Data augmentation: Different combinations of SpecAugment, additive noise, and flipping to create a more diverse training set, further improving robustness.

Ablation	Variant	Params (M)	MACs (M)	Accuracy (%)
Baseline	-	1.84	24.06	55.64
Preprocessing	Sampling rate 128 → 192	1.84	24.06	55.71
	Sampling rate 128 → 256	1.84	24.06	61.88
	Mels 8 → 16	1.84	24.52	59.52
	Mels 8 → 32	1.84	25.44	59.01
	Mels window size 0.2 → 0.1	1.84	24.55	54.91
	Mels window size 0.2 → 0.05	1.84	25.47	63.14
	Mels window stride = half size	1.84	24.55	57.60
Training	Learning rate 1e-4 → 5e-5	1.84	24.06	57.40
	Learning rate 1e-4 → 1e-5	1.84	24.06	55.63
	Regularization 0.05 → 0.1	1.84	24.06	56.60
	Regularization 0.05 → 0.2	1.84	24.06	60.55
	Dropout 0 → 0.1	1.84	24.06	56.08
	Dropout 0 → 0.2	1.84	24.06	54.15
	Label smoothing 0 → 0.1	1.84	24.06	49.75
	Label smoothing 0 → 0.2	1.84	24.06	46.05
Model	Attention heads 4 → 8	1.84	24.06	56.82
	Attention heads 4 → 12	1.84	24.06	56.88
	Encoders 2 → 4	3.40	47.67	57.60
	Encoders 2 → 6	5.00	71.28	57.78
	Decoders 0 → 2	3.95	37.18	55.62
	Decoders 0 → 4	6.05	50.29	54.85
	Decoders 0 → 6	8.16	63.41	55.06
	Hidden size 256 → 512	6.83	95.32	54.57
	Merge Mels typology: channels → samples	1.84	27.21	60.17
Data Augmentation	Spec augment only	1.84	24.06	55.64
	Additive noise only	1.84	24.06	57.01
	Flipping only	1.84	24.06	55.64
	Spec augment and additive noise	1.84	24.06	52.46
	Spec augment and flipping	1.84	24.06	55.64
	Additive noise and flipping	1.84	24.06	59.07
	Spec augment, additive noise, and flipping	1.84	24.06	58.76

Table 5.3. Ablation study of the proposed method on the AMIGOS dataset and the LOO validation scheme.

For the final tests described in Section 5.4, the combination of hyperparameters that yielded the highest accuracy was selected. Although increasing individual parameters often improved performance compared to the baseline, combining all the enhancements led to overfitting under the leave-one-out validation scheme. Consequently, it was decided to reduce the complexity of the EEG Transformer model by using 8 attention heads and 4 encoder layers, instead of 16 and 6, respectively. This decision was based on the observation that the performance difference between the optimal and reduced configurations was marginal. The reduced complexity not only mitigates overfitting but also shortens training times, providing a more practical and efficient solution without significant loss in accuracy.

While an ablation study provides insights into the influence of individual hyperparameters on model performance, it does not guarantee that the identified best hyperparameters are truly optimal. Testing all possible combinations of hyperparameters to find the absolute best configuration would be ideal, but it is impractical due to the immense computational time required given the number of parameters involved in this work. Therefore, we use the ablation study to understand how performance varies when adjusting one parameter at a time, and then combine the best settings from these individual tests to create an improved model configuration.

5.6 Additional tests

Another experimental approach involved incorporating a **triplet loss** mechanism. The primary objective was to train a model capable of distinguishing between EEG signals from the same subject while differentiating them from signals belonging to different subjects. This approach necessitated a modification of the data loader to create triplets in the following structure:

- **Anchor**: A generic EEG window from the dataset;
- **Positive**: Another EEG window from the same subject as the anchor;
- **Negative**: An EEG window from a different subject than the anchor.

For each window in the dataset, a corresponding triplet was created, resulting in n triplets where n is the total number of windows. Each window served as an anchor exactly once. During the training phase, the model was fed with these triplets, producing three distinct outputs: one for the anchor, one for the positive sample, and one for the negative sample.

The triplet loss was computed using the **TripletMarginLoss** function, which aims to minimize the distance between the anchor and the positive sample while maximizing the distance between the anchor and the negative sample. Additionally, *BCEWithLogitsLoss* (Binary Cross-Entropy Loss with Logits) was applied separately to the outputs of the anchor, positive, and negative samples, thus calculating three individual binary losses.

The total loss for each triplet was the sum of the triplet loss and the three binary cross-entropy losses, combining these four components to drive the backpropagation process. This comprehensive loss function aimed to enhance the model's ability to generalize by making it more robust in distinguishing between EEG signals from different subjects.

However, despite the theoretical advantages, this approach did not lead to an improvement in the model's performance. The increased complexity of using triplets required a significantly larger amount of training data. Each triplet demanded multiple samples, which, coupled with the necessity of calculating four distinct losses instead of just one, substantially increased the computational load and data requirements. Consequently, the expected benefits in generalization were not realized, and the performance remained unchanged.

Chapter 6

Conclusion

This work, titled "TA-EPE: A Transformer-based Approach to EEG Personality Estimation", set out to enhance personality estimation from EEG data using a deep learning approach based on Transformer architectures. The proposed model was evaluated on two prominent datasets, AMIGOS and ASCERTAIN, using various validation schemes to ensure a comprehensive comparison with existing works. The results demonstrated that our approach significantly outperformed traditional methods, especially when using a fixed mean for label discretization, which reduced label variability and noise, thus improving the model's ability to generalize across different subjects.

Despite the promising results, there are several limitations to this study. Predicting personality traits from EEG signals is inherently challenging due to the complexity and variability of brain activity. Additionally, the datasets used are relatively small, which may affect the model's robustness and generalizability. Moreover, Transformer models are computationally intensive, requiring substantial resources for training.

Future work should focus on addressing these limitations. One potential direction is to apply the proposed method to larger datasets, although such datasets with personality labels are currently scarce. Alternatively, data augmentation techniques could be explored to augment the existing datasets and improve the model's performance. Further research could also investigate more efficient Transformer architectures or hybrid models (e.g. Transformers and CNNs) to reduce computational demands while maintaining high accuracy. Overall, this thesis contributes to the growing field of EEG-based personality estimation and opens avenues for further advancements in AI-driven human behavior analysis.

Acknowledgements

Throughout this journey, I have had the opportunity to meet many people, from professors to fellow students, with whom I shared this incredible experience.

I would like to express my deepest gratitude to Prof. Luigi Cinque and Dr. Romeo Lanzino for their constant support, which allowed me to approach this thesis in the best possible way.

I am also grateful to my friends, who have been by my side over the years, sharing both the joys and the inevitable challenges along the way.

Finally, a special thanks goes to my family, who have supported me every step of the way. Thanks to their encouragement, I found the strength to push my limits and reach this milestone.

Traduzione italiana Durante questo percorso ho avuto l'opportunità di incontrare molte persone, dai professori ai colleghi di corso, con cui ho condiviso questa straordinaria esperienza.

Desidero esprimere la mia più profonda gratitudine al Prof. Luigi Cinque e al Dr. Romeo Lanzino per il costante supporto che mi hanno offerto, permettendomi di affrontare al meglio questa tesi.

Un ringraziamento va anche ai miei amici, che mi hanno accompagnato in questi anni, condividendo con me sia le gioie che le inevitabili sfide di questo cammino.

Infine, un pensiero speciale va alla mia famiglia, che mi ha sostenuto in ogni momento. Grazie al loro incoraggiamento, ho trovato la forza di spingermi oltre i miei limiti e giungere fino a questo traguardo.

The future belongs to those who believe in the beauty of their dreams.

Bibliography

[1] *AdamW Optimizer*. https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html.

[2] Takio Kurita Akinori Hidaka. "Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks". In: (2017). DOI: https://doi.org/10.5687/sss.2017.160.

[3] Abeer Al-Nafjan Aldayel Mashael Mourad Ykhlef. "Deep Learning for EEG-Based Preference Classification in Neuromarketing". In: *Applied Sciences* 10 (2020). DOI: https://doi.org/10.3390/app10041525.

[4] Tawfik Al Hadhrami Algarni Mona Faisal Saeed. "Deep Learning-Based Approach for Emotion Recognition Using Electroencephalography (EEG) Signals Using Bi-Directional Long Short-Term Memory (Bi-LSTM)". In: *Sensors* (2022). DOI: https://doi.org/10.3390/s22082976.

[5] Semen Kurkin Alla Chepurova Alexander Hramov. "Motor Imagery: How to Assess, Improve Its Performance, and Apply It for Psychosis Diagnostics". In: *Diagnostics (Basel)* (2022). DOI: https://doi.org/10.3390/diagnostics12040949.

[6] Semen Kurkin Alla Chepurova Alexander Hramov. "Transformed common spatial pattern for motor imagery-based brain-computer interfaces". In: *Frontiers in Neuroscience* 17 (2023). DOI: https://doi.org/10.3389/fnins.2023.1116721.

[7] Niki Parmar Ashish Vaswani Noam Shazeer. "Attention Is All You Need". In: (2017). DOI: https://doi.org/10.48550/arXiv.1706.03762.

[8] Gregory J. Boyle. "Myers-Briggs Type Indicator (MBTI): Some psychometric limitations". In: *Australian Psychologist* 30 (1995). DOI: https://doi.org/10.1111/j.1742-9544.1995.tb01750.x.

[9] Mei Wang Chaofei Yu. "Survey of emotion recognition methods using EEG information". In: *Cognitive Robotics* 2 (2022), pp. 132–146. DOI: https://doi.org/10.1016/j.cogr.2022.06.001.

[10] Davide Rivolta Davide Borra Francesco Bossi. "Deep learning applied to EEG source-data reveals both ventral and dorsal visual stream involvement in holistic processing of social stimuli". In: *Sci Rep.* 13 (2023). DOI: https://doi.org/10.1038/s41598-023-34487-z.

[11] *DEAP: A dataset for emotion analysis using eeg, physiological and video signals*. https://www.eecs.qmul.ac.uk/mmv/datasets/deap/.

[12] Xiaoming Liu Difei Liu Shen Liu. "Interactive Brain Activity: Review and Progress on EEG-Based Hyperscanning in Social Interactions". In: *Front Psychol* 12 (2021). DOI: https://doi.org/10.3389/fpsyg.2021.681900.

[13] Aditi Sakalle Harshit Bhardwaj Pradeep Tomar. "EEG based personality prediction using genetic programming". In: *Asian J Control* 25 (2023). DOI: https://doi.org/10.1002/asjc.3019.

[14] Aditi Sakalle Harshit Bhardwaj Pradeep Tomar. "EEG-Based Personality Prediction Using Fast Fourier Transform and DeepLSTM Model". In: *Computational Intelligence and Neuroscience* (2021). DOI: https://doi.org/10.1155/2021/6524858.

[15] Scott Huettel Hilke Plassmann Vinod Venkatraman. "Consumer Neuroscience: Applications, Challenges, and Possible Solutions". In: *Journal of Marketing Research* 52 (2015). DOI: https://doi.org/10.1509/jmr.14.00.

[16] Kambiz Badie Hosseini Mohammad Saleh Khajeh Seyed Mohammad Firoozabadi. "Personality-Based Emotion Recognition Using EEG Signals with a CNN-LSTM Network". In: *Brain Sciences* 13 (2023). DOI: https://doi.org/10.3390/brainsci13060947.

[17] Jian-Peng An Jia-Yi Guo Qing Cai. "A Transformer based neural network for emotion recognition and visualizations of crucial EEG channels". In: *Physica A Statistical Mechanics and its Applications* 603 (2022). DOI: https://doi.org/10.1016/j.physa.2022.127700.

[18] Edward F. Chang Joseph G. Makin David A. Moses. *Machine translation of cortical activity to text with an encoder–decoder framework*. Tech. rep. 2020. DOI: https://doi.org/10.1038/s41593-020-0608-8.

[19] Nicu Sebe Juan Abdon Miranda-Correa Mojtaba Khomami Abadi. "AMIGOS: A Dataset for Affect, Personality and Mood Research on Individuals and Groups". In: (2017). DOI: https://doi.org/10.48550/arXiv.1702.02510.

[20] Shaoqing Ren Kaiming He Xiangyu Zhang. "Deep Residual Learning for Image Recognition". In: (2015). DOI: https://doi.org/10.48550/arXiv.1512.03385.

[21] Shahera Hossain Khondoker Murad Hossain Md. Ariful Islam. "Status of deep learning for EEG-based brain–computer interface applications". In: *Front. Comput. Neurosci.* (2023). DOI: https://doi.org/10.3389/fncom.2022.1006763.

[22] *A Complete Guide to K-Nearest Neighbors*. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering.

[23] Jianhai Zhang Li Zhu Chongwei Su. "EEG-based approach for recognizing human social emotion perception". In: *Advanced Engineering Informatics* 46 (2020). DOI: https://doi.org/10.1016/j.aei.2020.101191.

[24] *LSTM networks*. https://www.linkedin.com/pulse/lstm-networks-abdullah-al-rahman/.

[25] Rosalia Dacosta-Aguayo Manousos A. Klados Panagiota Konstantinidi. "Automatic Recognition of Personality Profiles Using EEG Functional Connectivity during Emotional Processing". In: *Brain Sciences* 10 (2020). DOI: https://doi.org/10.3390/brainsci10050278.

[26] *Max Pooling*. https://paperswithcode.com/method/max-pooling.

[27] Kitti Kaiboriboon Mehdi Bagheri Hamaneh Numthip Chitravas. "Automated removal of EKG artifact from EEG data using independent component analysis and continuous wavelet transformation". In: *IEEE Trans Biomed Eng.* (2014). DOI: https://doi.org/10.1109/TBME.2013.2295173.

[28] Rui Na Mengxi Dai Dezhi Zheng. "EEG Classification of Motor Imagery Using a Novel Deep Learning Framework". In: *Sensors (Basel)* 19 (2019). DOI: https://doi.org/10.3390/s19030551.

[29] *Machine learning vs. neural networks: What's the difference?* https://www.techtarget.com/searchenterpriseai/answer/Machine-learning-vs-neural-networks-Whats-the-difference.

[30] *The Myers Briggs personality assessment.* https://prosper.liverpool.ac.uk/postdoc-resources/reflect/myers-briggs-type-indicator-personality-assessment/.

[31] *Naive Bayes Algorithms: A Complete Guide for Beginners.* https://www.analyticsvidhya.com/blog/2023/01/naive-bayes-algorithms-a-complete-guide-for-beginners/.

[32] Mojtaba Khomami Abadi Ramanathan Subramanian Julia Wache. "ASCERTAIN: Emotion and Personality Recognition Using Commercial Sensors". In: (2016). DOI: https://doi.org/10.1109/TAFFC.2016.2625250.

[33] Paul Costa Robert R. McCrae. "Reinterpreting the Myers-Briggs Type Indicator from the perspective of the five-factor model of personality". In: *Journal of Personality* 57 (1989). DOI: https://doi.org/10.1111/j.1467-6494.1989.tb00759.x.

[34] William La Cava Ryan J. Urbanowicz Melissa Meeker. "Relief-based feature selection: Introduction and review". In: *Journal of Biomedical Informatics* 85 (2018). DOI: https://doi.org/10.1016/j.jbi.2018.07.014.

[35] Leilah Schubert. *AI and EEG Transform Silent Thoughts to Text.* Tech. rep. University of Technology Sydney, 2023. URL: https://neurosciencenews.com/ai-eeg-thoughts-to-text-25343.

[36] Morteza Mousakhani Somayeh Raiesdana. "An EEG-Based Neuromarketing Approach for Analyzing the Preference of an Electric Car". In: (2022). DOI: https://doi.org/10.1155/2022/9002101.

[37] Rajiv Bajpai Soujanya Poria Erik Cambria. "A review of affective computing: From unimodal analysis to multimodal fusion". In: *Information Fusion* (2017). DOI: https://doi.org/10.1016/j.inffus.2017.02.003.

[38] *Guide on Support Vector Machine (SVM) Algorithm.* https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/.

[39] *The Big Five personality assessment.* https://prosper.liverpool.ac.uk/postdoc-resources/reflect/the-big-five/.

[40] Vincent Tran. "Positive Affect Negative Affect Scale (PANAS)". In: *Encyclopedia of Behavioral Medicine* (2013). DOI: https://doi.org/10.1007/978-1-4419-1005-9_978.

[41] Yvonne Tran. "EEG Signal Processing for Biomedical Applications". In: *Sensors (Basel)* (2022). DOI: https://doi.org/10.3390/s22249754.

[42] Pierluigi Reali Veronika Guleva Alessandra Calcagno. "Personality traits classification from EEG signals using EEGNet". In: *21st Mediterranean Electrotechnical Conference (MELECON)* (2022). DOI: https://doi.org/10.1109/MELECON53508.2022.9843118.

[43] *VGG Very Deep Convolutional Networks (VGGNet) – What you need to know*. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/.

[44] Paul A. Pavlou Vinod Venkatraman Angelika Dimoka. "Predicting Advertising success beyond Traditional Measures: New Insights from Neurophysiological Methods and Market Response Modeling". In: *Journal of Marketing Research* 52 (2015). DOI: https://doi.org/10.1509/jmr.13.0593.

[45] Heng Ji Zhenhailong Wang. "Open Vocabulary Electroencephalography-To-Text Decoding and Zero-shot Sentiment Classification". In: (2021). DOI: https://doi.org/10.48550/arXiv.2112.02690.