IZMIR UNIVERSITY OF ECONOMICS

FACULTY OF ENGINEERING

ELECTRICAL AND ELECTRONICS ENGINEERING

BIOMEDICAL ENGINEERING

FENG 498 PROJECT REPORT

Estimation of Human Emotions from EEG Signals Using Artificial Intelligence Methods

Author(s):

Eray Özcan KILIÇ	20210607034
Ünal KARADAYI	20200607027
Onat FİLİK	20200607020
Ege Deniz ASLAN	20200612002
Melike İLHAN	20200612011

Supervisor: Prof. Dr. Aydın AKAN

Table of Contents

1. Abstract

Since the beginning of time, human beings have sought to correctly interpret the feelings and mood of people that they encounter, whether on a commercial or any other basis. The interpretation of human feelings enables one to devise systems that determine human behavior directly, decision-making, and communication. Electroencephalography (EEG), which measures brain activity in terms of, is one of the prevalent methods employed for interpreting the neurological basis of feelings. Since EEG signals accurately portray the brain's electrical activity, they also show great promise in emotion state recognition. But owing to their complicated and noisy structure, it is more difficult to perform the analysis steps.
The key aim of the current research is to examine EEG data through machine learning and deep learning methods to correctly classify people's emotion states and present such classification outputs in a form understandable by a human being. In this, classification will be performed with the help of various machine learning models with the assistance of features calculated in the frequency and time domain, and then EEG signals will be converted into readable brain maps using deep learning-based methods (CNN). For this purpose, feature extraction was performed using statistical methods including skewness, kurtosis, Hjorth parameters, and band power. In addition to machine learning models and auxiliary MLP DNN, feature-based 2-D CNN and the ResNet-18 architectures were used for deep learning-based analysis of EEG data. Thus, both the efficiency of emotion recognition processes will be improved, and the results obtained will be made readable. This study aims to contribute to effective artificial intelligence, human-computer interaction, and personalized mental well-being applications.

2. Introduction

Interpreting and understanding human emotions have been a persistent core endeavor in psychology, neuroscience, and, more recently, in artificial intelligence. Emotions are a fundamental aspect of human cognition, decision-making, social interaction, and mental health. As the information age progresses, there is an increased requirement to incorporate emotional intelligence into artificial intelligence (AI) systems to enable more adaptive, personalized, and human-like technologies. From sympathetic virtual assistants that interact

with users to mental health monitoring systems that can detect early indications of emotional distress, the ability to recognize and interpret emotional states is becoming increasingly valuable.

One of the most potential methods of emotion recognition is by analyzing brain signals via Electroencephalography (EEG). EEG is a non-invasive technique that measures the electrical activity of the brain with excellent temporal resolution, allowing researchers to gain access to neural reactions to emotional stimuli in real time. In contrast to facial expression or tone of voice, which are subject to voluntary control or can be suppressed, EEG provides a direct window into the internal emotional processes of the brain.

Nonetheless, regardless of its advantages, EEG signals are non-linear, intricate, and highly prone to noise from various environmental and physiological sources, thus making emotion detection a difficult process.

To overcome these limitations, this project leverages the strength of artificial intelligence in the form of machine learning and deep learning algorithms for the classification of EEG data for emotional state detection. Machine learning algorithms like Support Vector Machines (SVM) offer a strong framework for the classification of multidimensional EEG signal features, while deep learning algorithms like Convolutional Neural Networks (CNNs) allow for the creation of intuitive visualizations for recognizing brain activity patterns for various emotional states. In this study, EEG data from the DEAP dataset, a publicly accessible dataset, is preprocessed and processed for the extraction of time-domain and frequency-domain features like mean, variance, skewness, kurtosis, power spectral density, and Hjorth parameters. These features are utilized in training machine learning models for emotion classification on valence and arousal dimensions. After classification, deep learning algorithms are used to convert the data into representations, i.e., 2D brain maps, which provide intuitive information about the neural correlations of emotion.

The applications of this research have far-reaching implications across several fields, ranging from mental health, HCI, neuromarketing, to assistive technologies. Affect-aware systems can aid individuals with mood disorders by providing real-time affect feedback, facilitate user experience in adaptive systems, and guide more efficient marketing strategies by considering consumer affective reactions. Finally, this work seeks to push the frontier of affective computing with an effective and interpretable EEG-based emotion recognition system. Through the integration of statistical feature extraction, machine learning classification, and deep learning visualization, this study not only adds to scientific understanding but also paves the way for real-world applications needing resilient and responsive emotion recognition

systems.

2.1 Problem Statement

Emotions are complex, multi-faceted phenomena at the core of human cognition and behavior but scientifically measuring them objectively is an ambitious scientific problem. Typical emotion recognition methods—such as facial expression interpretation, vocal tone, or self-reports—are subjective, typically delayed, and subject to voluntary control. EEG emotion recognition has more direct access to brain activity but with some issues of its own: the signals are noisy, high-dimensional, and highly variable between individuals. The biggest challenge is how to extract useful features from EEG signals and represent them precisely in emotional states, particularly in the valence-arousal space, with scalable, interpretable, and stable artificial intelligence approaches.

2.2 Motivation

The ability to interpret human emotions in real time from EEG signals has far-reaching applications in healthcare, human-computer interaction, adaptive learning technologies, and brain-machine interfaces. Emotion detection systems using AI can support early diagnosis and monitoring of mental health conditions such as depression, anxiety, and PTSD, where emotional dysregulation is a key symptom. Additionally, in applications like affective computing and neuromarketing, being aware of user emotions makes systems more responsive, empathetic, and user centric.

Furthermore, the advent of EEG wearables facilitates real-time emotion estimation more and more in non-clinical environments. Coupling machine learning models like SVM with deep learning models like CNNs leads to scalable classification and intuitive visualization of emotions even in noisy and mixed EEG signals. This spurs the need for an interpretable, robust pipeline that can close the gap between neuroscience and AI to create systems that possess a deeper understanding of and can respond to human emotional states. By giving back to this inter-disciplinary endeavor, the project is aiming to promote scholarly research in addition to applied uses of emotion-aware technology.

3. Literature Review

Today, human-emotional interaction has become an important research topic in the fields of artificial intelligence and biomedical engineering. People constantly give emotional responses to environmental stimuli, and understanding these responses correctly is of great importance in terms of psychology, neuroscience, human-computer interaction, and mental health applications. Although traditional methods use facial expressions, tone of voice, or subjective data based on surveys to understand emotions, these methods are generally based on external expression and are open to manipulation. For this reason, methods such as electroencephalography (EEG), which directly measure brain activity and provide more objective data, have come to the fore for many years. In this context, many past studies have guided our research.

3.1. Valence and Arousal

Emotion is a feeling derived from visual, verbal, or non-verbal stimuli caused by a situation. Emotions play a significant role in human interaction with others through communication. EEG (electroencephalogram) is one of the key components for emotion analysis and comprehension, along with categorized emotion models: Discrete and Dimensional Emotional Models. The Discrete Emotional Model is described as six different concepts, including fear, anger, sadness, nervousness, surprise, and happiness, as defined by Ekman 1971.[1]
On the other hand, there were arguments about the polarity of emotions. Scholorberg (1952) analyzed the facial expression errors made by subjects and turned the results into a circular graph including pleasantness-unpleasantness and attention-rejection.[2] Studies about the English language also supported Scholorberg's study by indicating the emotion dimensions as evaluation, potency, and activity.[3]
There were several more studies that provided emotional polarity of two dimensions. Russel&Pratt (1980) found that terms of emotions do not cluster about, but they fall meaningfully around the circular axis. Russell (1980) finalized this idea by subjects given terms, each representing a circular degree in the axis of two-dimensional emotions under valence and arousal values.[4]. In his work, valence values are described as emotions from positive to negative, and Arousal values are described as the physiological state of activeness and passiveness in the coordinate range.

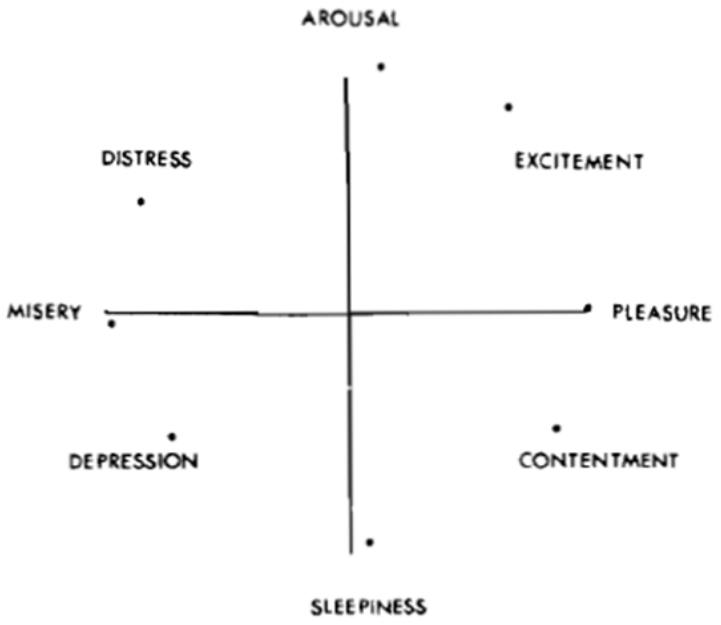

Figure 3.1. Russel's Model

The main purpose of the study conducted by Farokhah, Sarno, and Fatichah (2023) is to develop a simplified 2D CNN architecture to increase the classification accuracy and reduce the computational cost in EEG-based emotion recognition systems. EEG signals were converted into 2D spectrum images using the Short-Time Fourier Transform (STFT) method, and classification was performed on these images. The important point of this study for us is that, unlike traditional approaches where all 32 channels are used, high accuracy classification can be achieved by selecting only 10 EEG channels. In our project, we decided to make channel selection by using the results of this research. Channel selection was made by evaluating the classification performance of each channel independently with the DenseNet-201 architecture. As a result of the ranking made according to accuracy, F1 score, precision, and recall metrics, the 10 channels showing the best performance were determined as follows: Channels 11, 12, 13, 14, 15, 23, 24, 27, 28, 29.

In this approach, unlike the brain region-based channel selection frequently used in literature, a data-driven method based on direct classification success was preferred. The researchers stated that classical regional selections made over frontal, temporal, or parietal regions have a limited effect due to inter-individual differences and noise-induced deviations. Instead, the direct success shown by each channel in emotion classification was taken as a basis.

The results show that the model obtained using only 10 channels provided 3.53% accuracy increase for valence and 7.2% for arousal when compared to the classification made with 32

channels. Thus, while the suitability of the system for real-time applications was increased by reducing the number of channels, the classification performance was not compromised. This approach emphasizes the importance of channel selection, especially in difficult scenarios such as inter-subject classification, and contributes to the more efficient use of EEG signals. It has been effective in increasing the accuracy rate in the classification field in our project.[5] When values of arousal and valence are studied with EEG signals, in literature it is said that certain channels of frontal, temporal, parietal and occipital regions are more indicative of emotional states. Frontal lobe is crucial especially in valence perception (positive and negative emotions); in this context, channels such as F3, F4, F7, F8 and Fz are used very extensively. Since the magnitude of arousal corresponds to more distributed brain activity, temporal and parietal region channels T7, T8, P3, P4, P7 and P8, along with central (Cz, C3, C4) and frontocentral (FC1, FC2, FC5, FC6) regions are utilized. Additionally, occipital region channels O1 and O2 are utilized to pick up responses to visual stimuli. In the majority of the investigations across the DEAP dataset, high classification accuracies have been achieved with various combinations of these channels, and it has been emphasized that frontal region asymmetries are strongest for classifying between positive/negative emotions. According to the projects that we used in our project, we discovered that the channels "F7", "Fp1", "Fp2", "F8", "F3", "F4", "P3", "P4", "P7", "O1", "O2", "P8" were suitable for our research and we used these channels in our research.[6]

3.2. Feature Extraction

Feature Extraction is one of the key components of signal processing for signal detection, recognition, and classification. There are several features to consider for emotion recognition to be utilized in the project, such as PSD (Power Spectrum Density), RMS (Root Mean Square), Alpha Asymmetry, Hjorth Parameters, Skewness, Kurtosis, and Principal Component Analysis (PCA).

There are a variety of techniques to extract features to calculate PSD, such as Fast Fourier Transform (FFT), Time Frequency Distributions (TFD), Wavelet Transform (WT), Auto Regressive Method (ARM), Eigenvector Methods (EM), and so on [7]. The Welch method, which is commonly used in emotion recognition analysis, is determined to calculate PSD according to its convenient implementation and enhanced performance [8]. In a study of emotional state detection, Welch technique is utilized for EEG power spectrum calculation estimation with positive, negative, and neutral stimuli, Picture types in valence and arousal dimensions. [9] Additionally, in a study comparing different spectral analysis methods, the

Welch method has shown 96.16 % accuracy rate with SVM implementation, performing the highest accuracy performance among periodogram and multitaper methods. [10]
Alpha Asymmetry is used to analyze emotional processing in cortical activity asymmetry. Left cortical activity is associated with positive, approachable behaviors, on the other hand, right cortical activity is associated with negative, withdrawal behaviors. [11]. F3 & F4 electrodes are common electrodes to measure alpha asymmetry in studies for EEG signal analysis.
RMS (Root Mean Square) of EEG bands is a measurement of signal strength. The statistical results of RMS demonstrated the varying signal magnitude. Therefore, the RMS measurement is functional for the signals that contain both positive and negative wave variations. [12] We also observed in another paper that Hjorth parameters (activity, mobility and complexity) were used as time domain descriptors of EEG signals for feature extraction. These Hjorth parameters were extracted from five frequency bands (delta, theta, alpha, beta, gamma) across five brain lobes to statistically characterize EEG waveforms. 75 features based on the Hjorth parameters were used to train an SVM classifier to identify emotion in one study, with as much as 70% accuracy for classifying two emotional states, happy and sad [13]. In addition to this, skewness and kurtosis are also typical parameters for analyzing the distribution features of EEG signals of human emotions. The reason these parameters are applied in our project's EEG signals is mainly because of a study that involved classifying seven emotional states. Skewness was used in this study to measure the asymmetry of the signal distribution, and kurtosis quantified the sharpness and dispersion. In conclusion, this paper is useful for establishing the difference in several emotional states since there are visible differences in skewness and kurtosis values. Besides this, measurements involving statistical measurements prove useful for EEG signal emotional state recognition in determining nonlinear properties besides the usual mean and standard deviation values [14].
Principal Component Analysis (PCA) is a reduction technique to transform large amounts of correlated data into uncorrelated variables known as principal components. The primary axis is computed to explain the greatest amount of data variability. and take into consideration the remaining variability's direction, in decreasing order of the data's variability. Thus, the direction of the next largest change is represented by the axes that follow, and so on. To lower the dimensionality, the remaining components can be disregarded because the modified data's initial few components contain the majority of the variation.[15]
Feature normalization is preferred to be used to convert individual quantitative features into equally numerical contributions on the model training. Min-max normalization is generally

used for re-scaling the features to a range of [0,1], which is applied to each feature column.[16] In addition, batch normalization is used for neural networks, which enables stability improvement and faster training. Every component of a neural network's layer is normalized to zero mean and unit variance using batch normalization, which is based on the statistics of a mini batch in which a learnt scaling and shifting parameter is assigned to each activation.[17]

The standardization method is used to rescale the data around the standard deviation (1) by centering the average of the features, the mean (0). The standardization method could be preferred according to the used machine learning model, such as SVM (Support Vector Machine), in which the kernel and regression are utilized.[18]. Robust Scaler is a standardization method for reducing the outliers impact and improving convergence in machine learning datasets. It scales the data by the median and interquartile range (IQR). [16]

3.3. Machine Learning Algorithms

There are several different machine learning algorithms in present. Due to their high efficiency, the algorithms used in this study are Random Forest, XGBoost, Support Vector Machine, and K Nearest Neighbors.

3.3.1 Random Forest (RF)

Random Forest is an algorithm that utilizes data mining techniques with a supervised learning algorithm. It involves construction of several decision trees by random data gathering of random input feature selection and bootstrap aggregation which has an impact on overfitting reduction and accuracy increment. The strength of each tree node and the correlation between them in random forest determine the generalization error of the classifiers.[19]. In the study of Edla et al. (2018), the Random Forest classifier technique is used to analyze EEG data for human mental states determination such as concentration and meditation. The classifier was chosen for its ability to handle high-dimensional data. The model was trained with 90% of the collected dataset of 600 and tested with the remaining 10% dataset of 60 for 40 subjects. As a result, the study has showed 75% accuracy performance demonstrating the potential of Random Forest in EEG based dataset.[20]. In the study of Kusumaningrum et al. (2020), a development for emotion recognition system is conducted with DEAP dataset by extracting features from time and frequency domain with by using different classification algorithms as k-NN, Weighted k-NN, SVM and Random Forest. The recognition accuracy is compared with

n-cross validation which result in the highest accuracy level of Random Forest with 62.58% for 60% in average.[21]. In the study of Zhang and Min (2020), accuracy and stability of EEG-based emotion recognition under video stimulation were improved with observation of the variation in Differential Entropy (DE) before and after stimulation using Wavelet Packet Transform (WPT). EEG signals from the DEAP database were divided into 15 equal chunks, and DE was recorded for each segment to be calculated and utilized to classify emotions into four classes of the valence-arousal emotional space using a Random Forest (RF) classifier. The proposed WPT-RF model achieved an overall accuracy of 87.3%, and when applied to individual subjects, its accuracy was 97.7%, indicating significant improvement in emotion recognition performance.[22].

3.3.2 XGBoost

XGBoost is a non-parametric supervised machine learning algorithm with classification and regression performance based on gradient boosting to be highly scalable by minimizing loss values. Parallel computing and distributed computing enables faster outputs.[23].
In the XGBoost Algorithm, a decision tree is determined with residuals based on the input and target values according to the predictions. Residuals are created with similarity scores of the data in each tree to result in subsequent gain similarity which is utilized to select the proper feature and threshold value for the node. In that case, the tree output becomes the new residual input for the next tree construction meaning, each tree is dependent to one another unlike Random Forest Algorithm where the trees are independent in an equal weight.
In the study of Parui et al. (2019), Emotion Recognition is analyzed from EEG brain signals and optimizing the features testing with the DEAP dataset. The study includes comparison of different machine learning classification algorithms as KNN, Naive Bayes, Random Forest, C4.5 Decision Tree and mainly focusing on XGBoost. The threshold is selected as 4.5 for feature priority selection in classification. Among all classification algorithms, XGBoost has shown the most accuracy with 75.97% valence, 74.206% arousal, 75.234% liking and 76.424 % dominance values.[24]. In the study of Asemi and Farajzadeh (2024), emotion recognition is analyzed with DEAP and MAHNOB-HCI datasets. Different classifier algorithms such as SVM, KNN, and XGBoost are utilized. Fractal dimension, wavelet energy, power features, and Grey Wolf Optimization (GWO) is implemented for feature extraction. For the DEAP dataset, the GWO approach improved the accuracy from 62.24% for valence and 67.08% for arousal into 89.08% for valence and 89.63% for arousal in in XGBoost.[25]. In the study of Zong et al. (2023), emotion recognition is analyzed with DEAP and DREAMER datasets by

utilizing the algorithms FCAN and XGBoost. Feature extraction is done with feature attention network (FANet) which is a FCAN module. The FCAN part is utilized for extracting and combining deep features from differential entropy (DE) and power spectral density (PSD) in four frequency bands of EEG. These enhanced features are then classified using XGBoost. The model was validated using the DEAP and DREAMER datasets, achieving emotion recognition accuracies of 95.26% and 94.05%, respectively.[26].

3.3.3 Support Vector Machine (SVM)

SVM is a non-parametric supervised machine learning algorithm with classification and regression application analysis. Decision making is determined by the hyperplane that maximizes the margin value (the distance between the hyperplane and the support vectors) between two different classes by satisfying all data points in N-dimensional feature space. The classification of SVM can be attained by linear or non linear separation. Within this classification technique, the classification errors are minimized, and generalized classification ability is improved.[27]. In the study of Saccá et al. (2017), Electroencephalography (EEG) signal analysis is done with SVM to detect cerebral disorders. The dataset includes EEG values of thirty healthy subjects and thirty Creutzfeldt-Jakob disease (CJD) patients having sixty in total. The algorithm involves data processing, feature extraction, principal component analysis (PCA), normalization and SVM phase. The study achieved the classification accuracy of 96.67%, the specificity of 93.33% and the sensibility of 100%. [28]. In the study of Sachin, & Kumar, and Deepak. (2022), DEAP (a Database for Emotion Analysis Using Physiological Signals) dataset is analyzed with SVM classifier algorithm for emotion analysis. The algorithm is utilized by 80% data for training and 20% for testing.
The features of high and low valence and arousal values in females and males are executed in the algorithm. The features are separated into eight different datasets holding high valence and low valence for only male, high valence and low valence for only female, high arousal and low arousal for only male, high arousal and low arousal for only female, high valence and low valence male and female, high arousal and low arousal for male and female. The results showed approximately between 60% and 70% for the only male and only female, whereas 99.02% for both male and female. [29]. In another study of emotion classification using Long Short-Term Memory (LSTM) networks performances in comparison of classical machine learning algorithms, With an overall accuracy of 94.12%, SVM outperformed K-NN at 78.01%, RF at 90.00%, and LSTM at 85.16%. [30].

3.3.4 K-Nearest Neighbors (K-NN)

K-NN is a non-parametric supervised machine learning algorithm with classification and regression performance for unified data points. The algorithm utilizes labeled classes based on the majority by taking into account the label which is a nearest neighbor around to the data point in classification. This results in the prediction of the concept that similar points can be located near to one another. The hyperparameter K in the K-NN represents the number of data points nearest neighbors, in which different values of K can lead to different classifications. The high accuracy of the algorithm depends on the hyperparameter K, the function of distance, and the classification role of the algorithm.[31]. In the study of Awan et al. (2016), Electroencephalography (EEG) signals are analyzed by ten healthy people in the age range between eighteen and forty-five with K-NN classification algorithm. The study utilized Segmentation and Selection (SnS) and Root Mean Square (RMS) for the feature extraction of EEG data, in which the algorithm achieved 96.1% accuracy.[32].

There are several studies about K-NN related with the DEAP (a Database for Emotion Analysis Using Physiological Signals) dataset.

In the study of Sari et al. (2023), Non-Linear emotion classification is done with K-NN and weighted K-NN algorithms with the DEAP dataset, which achieved a 60.15% highest recognition value with the Chi-Square selection method. [33]. In the study of Diah et al. (2019), Time and Frequency Domain Feature selection is analyzed for K-NN and weighted K-NN by using the DEAP dataset. The experiment's findings indicate that, with a recognition rate of 60.68%, time-frequency domain feature extraction performs best when PCA feature selection and the k-NN classifier are used. Additionally, this experiment demonstrates that for any feature selection, there is no noticeable difference between k-NN and weighted k-NN classification.[34]. In the study of Zhou et al. (2024), research on emotional recognition for home robots based on EEG dataset is carried out with the DEAP dataset. The study utilized preprocessing, feature extraction, and classification to identify users' emotional states in real time. Filtering is applied on the raw EEG data as well as the feature extraction of frequency, time and spatial domains within brain networks. LASSO regression was implemented for feature selection.

Several different machine learning algorithms are applied. With an average accuracy of 90.3%, the study's findings demonstrated that the KNN classifier outperformed the others in emotion recognition.[35]

3.4. Deep Learning Algorithms

Deep Learning has increased its popularity because it can learn hierarchical presentation automatically from raw data. Deep Learning models have some advantages in EEG-based emotion recognition. For example, learn blended, non-linear approaches to data, the ability to receive appropriate features, lowering the necessity for manual feature engineering [37].
One of the most significant models in this area is Convolutional Neural Networks (CNNs). The usage of CNN has grown and started to be used for EEG analysis. Convolutional neural networks can learn and receive spatial features from EEG signals, specify the design between different electrode spots that are convenient for identifying emotions. For instance, from the electrical activation of the frontal lobe, CNN can learn to understand happiness. Several CNN architectures have been detected, such as 2D CNNs that process multi-channel EEG data (e.g., time × channel or frequency × electrode matrices) as a spatial map and 1D CNNs that can perform on single channels or frequency bands [44]. Another usage area is receiving emotion-related features from the raw EEG data [38].
The learning capacity of the specified presentation of CNNs makes it powerful for analyzing the spatial circulation of the EEG activity. In EEG-based emotion recognition, CNN can be used to specify the model of brain activity that is defined for particular emotions. The convolutional filters are used to remove the specified composition and pooling layers to decrease the dimensionality. For example, Yao et al. have used ResNet18 to extract DE features and receive increased accuracy in emotion classification on the SEED-V dataset [38]. In addition, a CNN may be able to learn to recognize frontal brain region activity, which is connected to positive valence, while increased action in temporal regions may show higher arousal. Also, they can be used on the DEAP dataset if the represented features, like PSD or DE, are removed spatially [40].
Meanwhile CNNs excel at receiving spatial patterns, another important model for processing temporal data is Recurrent Neural Networks (RNNs). The main usage area of RNNs is for processing consecutive data. EEG signals are congenitally temporal, and RNNs can possess the active variation in brain agility that performs like emotions build up in time. Short-Term Memory (LSTM) [39] and Gated Recurrent Unit (GRU) [40] networks are common RNNs. These networks build up the pattern of the temporal dependencies between EEG segments by advancing emotion recognition. Moreover, a long short-term memory (LSTM) network has been used to remove the features that are involved in the Big Five Personality characteristics

in a study [38]. Clerico et al. [41] propound a basic and combined layout for the sorting radar and communications signal using Long Short-Term Memory (LSTM) neural networks [41]. Holding the temporal dynamics of EEG signals at RNNs excels, which is important to comprehend the change of emotions in time. Emotions are not static, which means that EEG operation mirrors this dynamic shift, and unfolds and changes. RNNs can show the past information that is kept in the hidden state, and efficiently build up the temporal dependencies in EEG data. For instance, an RNN may learn that the rapid improvement in theta band, chased by a fractional decrement in alpha band operation, is characteristic of a specified emotional transition. LSTM and GRU networks are especially convenient for this example, as they are connected to alleviate the vanishing gradient problem, empowering them to learn long-range dependencies in the data.

Identifying the complementary abilities of both spatial and temporal processing, hybrid approaches have been developed. This is the combination of CNNs and RNNs. For instance, to remove the spatial features from EEG data, CNN is usable; to process the result, the sequence of spatial features from capturing temporal dynamics, RNN is usable. This approach powers the learn both spatial and temporal patterns in the EEG signals, making the performance strong [42]. For instance, while CNN is removing the features from different locations of the brain, LSTM can learn to maximize the performance over emotional experiences, while maximizing the performance, it leads LSTM to learn both “what” and “when” questions through the brain to have more reliable emotion recognition.

On the furthers side of traditional convolutional and recurrent approaches, Graph Neural Networks (GNNs) is a special perspective on brain connectivity. GNNs are important for analyzing EEG signals to understand the local relationship between different parts of the brain. They can build up combined colonies due to EEG channels, which leads to increased emotion recognition precision.

With GNN, EEG channels can be shown in a graphical illustration as dots while building up a powerful channel connection [43]. In addition, this graphical illustration is popular in using for emotion classification (positive, negative, and neutral) in EEG signals [36]. This representation proves that GNN can show the different regions of brain activity during emotion processing. As an example, the Amygdala and the prefrontal cortex are responsible for emotional organization, GNN may be able to understand this organization considering the increased emotional arousal. According to this information, GNNs can show the neural-emotional mechanism to advance the accuracy of EEG-based emotion recognition.

Beside these specialized neural network models, traditional feedforward networks continue to have an important role. The main aim of Deep Neural Networks (DNNs) is to feedforward the neural network with multiple layers. DNNs are efficient for features like DE, PSD, or statistical measures that are pre-removed from EEG signals. For example, Fernandes et al. used DNNs on the DEAP dataset and showed the efficiency against KNN and SVM classifiers [36].

Between the graph-based methodology, Graph Convolutional Neural Networks (GCNNs) has a more advanced approach to spatial relationships. GCNNs show the spatial communication between EEG channels, which is shown with a graphical illustration of the place where electrodes are nodes and edges of the connections. GCNNs are popular for EEG signals due to spatial topology, which is important for emotion recognition. The improved version of GCNN is SparseDGCNN, which can achieve 95% accuracy at the DEAP dataset by centering the meaningful relationships over brain regions [44].

Ending the deep learning approaches for EEG emotion recognition are unsupervised learning methods. To squeeze and rebuild the data, the unsupervised networks, which are Autoencoders, are used. Autoencoders reduce noise in EEG signals and dimensionality before the classification. For adaptation or semi-supervised learning, Variational Autoencoders (VAEs) are being used, which adds probabilistic modeling. However, it is not mostly preferred for classification in DEAP, it is being used for preprocessing or transferring the learning tasks [37].

3.4.1 Deep Neural Network (DNN)

Deep Neural Network (DNN) is a type of Artificial Neural Network (ANN) which organizes multiple layers between neurons which makes DNN different from ANN which has shallow networks.[53]. The mapping between layers occurs where the output of a layer feeds the input of another layer within a nonlinear activation. The mapping implementation is learned with the technique error backpropagation through data adaptation with each neuron weight. [54]. In the study of Al-Nafjan et al. (2017), human emotions are classified using DNN by DEAP dataset. DNN is adapted with the features are extracted from Power Spectral Density (PSD) and Frontal Asymmetry including Higher Order Crossings, time-domain statistics, Common Spatial pattern, Wavelet entropy, EEG spectral power, Hjorth parameters, and coherence analysis. The results have shown 82.0% overall accuracy for valence and arousal values. [55]. In the study of Wang et al. (2024b), emotion recognition development is analyzed by Short-Time Fourier Transform (STFT) and Wavelet Transform (WT) with DenseNet, DNN and

Long Short-Term Memory (LSTM) networks classification using DEAP dataset to perform a rehabilitation feedback system. The classification is separated under three groups of emotions as 3,5 and 9. The study resulted as the highest accuracy of 88% valence and 90% arousal by LTSM with WT. The DNN algorithm has shown the most accuracy rate at 3 grouped categories by 82% for STFM and 76% for WT application. [56].

3.4.2 2D – CNN

In recent years, uses of two-dimensional convolutional neural networks (2D-CNNs) have been given much more importance as a very promising approach in EEG-based emotion recognition. Another of many strengths of the models is that 2D-CNNs are capable of extracting the spatial features from brain activity by treating EEG data as image-like structures. Compared to traditional machine learning methods relying on manually engineered features and domain expertise, 2D-CNNs allow end-to-end learning through automatic learning of spatial relationships between EEG channels. This is particularly useful when the EEG data is projected onto 2D spatial grids or spectrograms that preserve topographical or frequency-based representations. The hierarchical nature of CNNs facilitates hierarchical representation learning—where shallow layers learn low-level spatial patterns, and deeper layers learn higher-level emotional signatures of brain activity. The application of 2D-CNN in this context allows for the characterization of complex emotional patterns embedded in electrical activity of the brain, especially when spatial and spectral relations between EEG channels are preserved. Further, their sparsity-based connectivity and parameter sharing properties make 2D-CNNs computationally cheap and suitable for real-time or embedded systems.

Moctezuma et al. (2022) suggested a 2D-CNN classification system with NSGA-II added for the best EEG channel selection. Models were trained on the DEAP dataset and compared results using full 32-channel input vs. subsets of 1 to 15 channels. Presence/absence of EEG channels were represented by each gene in the genetic algorithm, and the two objectives were to maximize classification accuracy and minimize the number of channels. Results showed that as low as 8 arousal and 2 valence channels are still allowed for a maximum of 100% accuracy on binary valence classification tests (low or high). Tests in their article were demonstrated to emphasize selected channels such as AF4, O1, C4 usually with high discriminatory capabilities. Furthermore, in this study, per-subject training (intra-subject models) was used, leveraging CNN's intrinsic ability at spatial feature learning without heavy feature engineering.

Yang et al. (2018) proposed a hybrid Parallel Convolutional Recurrent Neural Network (PCRNN) model that incorporates CNN and LSTM modules to capture spatial and temporal relationships from EEG signals. They employed a novel pre-processing pipeline that removed baseline activity by matrix subtraction and projected 1D EEG vectors onto 9×9 spatial frames that corresponded to the international 10-20 electrode placement. The 2D slices obtained were then fed into a CNN of three 4×4 convolution layers with batch normalization and ELU activation followed by LSTM layers that can identify sequential dependencies. They have not utilized pooling layers to maintain all spatial information due to the sparseness of EEG data. Independent CNN and LSTM processing was then followed by feature fusion using depth-wise concatenation and 1×1 convolutional reduction. Their model was 90.80% accurate in valence and 91.03% in arousal classification using 10-fold cross-validation. Notably, the baseline-subtracted data were ~32% more accurate than raw EEG.

Farokhah et al. (2023) introduced a low 2D-CNN architecture with channel selection mechanism and spectrogram-input. The DEAP EEG signals were down sampled to 128 Hz, band-pass filtered (4–45 Hz), and converted into 224×224 RGB spectrograms using the Short-Time Fourier Transform (STFT) technique. All EEG channels were ranked based on utilizing pre-trained DenseNet-201 and the top 10 performing channels were chosen in terms of accuracy, precision, recall, and F1-score. This brought the input size down from 49,920 to 12,800 images. Their simple CNN contained multiple Conv-MaxPool-Dropout blocks and was trained using Adam optimizer with MSE loss, tested using Leave-One-Out Cross Validation (LOOCV). Results were +9.73% (valence) and +11.7% (arousal) accuracy boost when utilizing all of the 32 channels; and additional +3.53% and +7.2% boosts, respectively, when using only 10 selected channels. Interestingly, this model worked better than the deeper networks like ResNet and DenseNet, showing the strength of simpler models with well-designed inputs.

Based on these works, our project employs the 2D-CNN approach for emotional state estimation—i.e., valence and arousal—from EEG signals of the DEAP database. Unlike most previous works, we employ preprocessed EEG signals beforehand and are interested in transforming extracted features into 2D representations for CNN-based learning. Topographical mapping is utilized to preserve spatial relationships of EEG electrodes to improve recognition accuracy. In addition, our pipeline can be combined with per-subject modeling to extract personalized emotional responses and reduce variability. Apart from that, our contribution also entails Alpha Asymmetry analysis among representative electrode pairs (F3–F4, F7–F8, P7–P8) to further render the model sensitive to hemispheric differences on

the basis of emotions. This neuropsychologically driven aspect can add biologically interpretable inputs to the representation of the deep model. We also study other image-encoding methods like spectrogram-based representation to allow 2D-CNN to learn EEG signal spectral-temporal patterns. By combining spatially-informed deep learning and theory-based psychological feature engineering, this work tries to advance the accuracy and interpretability of emotion recognition from EEG signals—an essential need for mental health, adaptive human-computer interaction, and neuromarketing.[57] [58] [5].

3.4.3 RESNET 18

Resnet is a neural network model which was introduced in the ImageNet Large Scale Visual Recognition Challenge in 2015 by being first in the competition with a top-5 error rate of 3.57%. Residual learning helps the algorithm to learn more deeply and efficiently by transferring inputs to the subsequent layer. Meaning, Layers are connected with the input of the previous layers' output which have an impact on the algorithm speed and accuracy enhancement with correcting errors at each layer. The study found that Residual Learning with 18 layers has great accuracy with high convergence speed at early state compared to the other tested layers.[59]. In the study of Khan et al. (2018), Residual Network models performances are compared for image recognition. The models used in the study Resnet18, ResNet50, ResNet101 and ResNet152 are tested on two different datasets: cancer and malware dataset. The results have showed the highest accuracy of Resnet152 as 0.9876 and 0.8798 with 2131 s and 9248 s prediction time, in comparison, Resnet18 showed the accuracy of 0.9719 and 0.83 with 1131 s and 2701 s prediction time, having the highest training speed in cancer and malware datasets.[60]. In a study of Yao et al. (2024), the Resnet18 structure is utilized for human emotions classification with SEED-V dataset for the identification of five human emotions of fear, disgust, happiness, sadness, and neutral. The study extracts feature with differential entropy (DE) to analyze emotions' variety and complexity with residual connections resulting in 95.61% accuracy.[61]. In the study of Bagherzadeh et al. (2022), emotional recognition analysis is done with the Frequent Effective Connectivity maps (FEC) evaluated with the Partial Directed Coherence (PDC) value measured from EEG frequency bands. The maps are classified under five emotion states with different CNN models as DarkNet-19, Xception, ShuffleNet, AlexNet, Inception-v3 and ResNet-18 with Leave-One-Subject-Out (LOSO) Cross-Validation in MAHNOB-HCI, DREAMER, and DEAP datasets. The results have shown that Resnet18 had the highest accuracy of 95.25%, 96.00% and 94.27% for the datasets MAHNOB-HCI, DREAMER, and DEAP.[62]. In our study, we

utilized from the topo maps derived from the features extracted from the DEAP dataset utilizing Resnet18.

4. Methodology

4.1 Data Preparation and Feature Extraction

4.1.1 Software Environment and Dependencies

Table 4.1: Software and hardware specifications

Component	Specification
Programming Language	Python 3.11
Environment Manager	Virtual Environment (venv/virtualenv)
Operating System	Windows 10
CPU	Intel Core i7-7700K
GPU	NVIDIA GeForce 1070
RAM	32 GB
CUDA Version	12.8 (Used only for Resnet)
cuDNN Version	8.9 (Used only for Resnet)

Key Libraries & Versions	NumPy, Pandas, Scikit-Learn, TensorFlow, Keras, MNE, XGBoost, Seaborn, Matplotlib

Main Libraries

- **Pandas**
  Pandas offers basic data tables called DataFrames and Series. We used it to read and write CSV or Excel files, sort and filter data, and perform summary statistics with less code.
- **Scikit-Learn**
  Scikit-Learn contains many classic machine-learning methods such as decision trees, support vector machines and clustering. It also has tools for preparing data (like scaling or encoding), training models, and checking their accuracy.
- **TensorFlow & Keras**
  TensorFlow is a strong library for building deep-learning models. Keras, its friendly UI. Together, they let us create neural networks, train them in CPU or with GPU support, and monitor performance with built-in tools.
- **MNE**
  MNE is designed for working with EEG and MEG recordings. It can load raw brain EEG data, clean and filter it, split it into epochs, and draw topographic maps that show activity across the scalp.
- **Seaborn**
  Seaborn makes it easy to draw statistical graphics like heatmaps or distribution plots. It sits on top of Matplotlib and gives nicer default styles with just a few commands.
- **Matplotlib**
  Matplotlib is the standard plotting tool in Python. It lets you make custom charts such as line plots, bar graphs, scalp maps and control every detail of the figure.

4.1.2 Data Acquisition and Preprocessing

4.1.2.1 DEAP Dataset Download

We obtained the DEAP dataset from the official Queen Mary University of London site (https://www.eecs.qmul.ac.uk/mmv/datasets/deap/). The download consists of 32 files, one for each participant, in Python ".dat" format. After downloading, we verified file integrity using MD5 checksums provided on the website. No further unpacking was needed, since each .dat file already contains all trials in a single binary.

All participant files were stored under a single data/ folder in our project root. The final structure is:

```
project_root/
└── data/
    ├── s01.dat
    ├── s02.dat
    ├── …
    └── s32.dat
```

This flat layout makes it straightforward to iterate over data/sXX.dat in code. We did not subdivide into raw versus processed folders, because we handle splitting and segmentation on the fly in later steps.

4.1.2.2 Loading Raw EEG and Labels

To read the binary .dat files, we implemented a Python helper function called load_deap_dat(path). Internally, this function uses pickle (for Python 3 compatibility) to deserialize the MATLAB structures. It returns two NumPy arrays:

- eeg_data of shape (n_trials, n_channels, n_samples)
- labels of shape (n_trials, n_labels)

For DEAP, each participant file contains 40 trials of 63 seconds' duration, sampled at 128 Hz on 32 EEG channels plus 8 peripheral channels. In our work, we select 12 EEG channels of interest (especially frontal and temporal sites) and ignore the rest.

Immediately after loading, we remove the first 3 seconds of each trial to avoid initial adaptation artifacts. Since each trial is 63 s long, discarding 3 s leaves 60 s of clean data. With 12 channels and 128 Hz sampling, each trial becomes a (12 × 60 s × 128 Hz) = (12 × 7680) array. Altogether, for 40 trials, the returned eeg_data has shape (40, 12, 7680), and labels has shape (40, 4), where the four label columns correspond to valence, arousal, dominance, and liking. We work with valence and arousal, so we ignored the other two columns which are dominance and liking.

4.1.2.3 Trial Segmentation

To prepare inputs for classification, we split each 60-second trial into non-overlapping windows of 4 seconds. At 128 Hz, each window contains 4 s × 128 Hz = 512 samples. Because 60 s / 4 s = 15, this yields exactly 15 segments per trial. We perform this step in a function called segment_trials(eeg_data, labels), which returns two arrays:

- segments with shape (40 × 15, 12, 512)
- segment_labels with shape (40 × 15, 4)

Here is the core of the pseudocode, with inline comments for clarity:

```
def segment_trials(eeg_data, labels):
    segments = []
    segment_labels = []
    n_trials, n_chan, _ = eeg_data.shape
    samples_per_win = WIN_SEC * FS  # 4 * 128 = 512

    for trial in range(n_trials):
        for win_idx in range(15):          # 15 windows per trial
            start = win_idx * samples_per_win
            end = start + samples_per_win
            window = eeg_data[trial, :, start:end]
            segments.append(window)          # shape (12, 512)
            segment_labels.append(labels[trial])
    # Convert lists to NumPy arrays for model input
    return np.array(segments), np.array(segment_labels)
```

Figure 4.1 : Trial Segmentation Code in Pycharm

After segmentation, each segment retains its original trial label, so we replicate the label 15 times per trial.

4.1.3 Feature Extraction

In this study, we extract both time-domain and frequency-domain features from each 4-second window of EEG data. For every channel and every window, we compute five statistical descriptors in the time domain and five band-power measures in the frequency domain. These features are then concatenated across the 12 selected EEG channels to form a single 120-dimensional feature vector per window.

4.1.3.1 Time-Domain Features

Time-domain features capture the statistical and dynamic properties of the raw EEG signal in each window:

- Variance measures the dispersion of the signal values around their mean. A larger

variance indicates more fluctuation in amplitude.

- Mobility (the Hjorth mobility parameter) is defined as the square root of the variance of the first derivative of the signal divided by the variance of the signal itself. It quantifies the signal's frequency content roughly as a ratio of standard deviations.
- Complexity (the Hjorth complexity parameter) compares the mobility of the first derivative to the mobility of the original signal. Higher complexity indicates more rapid changes in frequency.
- Skewness measures the asymmetry of the signal's amplitude distribution around its mean. A positive skew suggests a longer tail on the right side of the distribution.
- Kurtosis quantifies the "peakedness" of the amplitude distribution relative to a normal distribution. A higher kurtosis indicates more extreme deviations (heavy tails).

These five descriptors provide complementary views of signal variability, frequency content, and distribution shape.

4.1.3.2 Frequency-Domain (Band-Power) Features

Frequency-domain features summarize the power of the EEG signal within standard neurophysiological bands. We define a constant BANDS = {δ:0.5–4 Hz, θ:4–8 Hz, α:8–13 Hz, welchβ:13–30 Hz, γ:30–45 Hz}. For each window and channel, we compute the band-power by integrating the power spectral density via Welch's method over each band's frequency range. These five band-power measures reflect the relative strength of slow (δ, θ) versus faster (α, β, γ) rhythms, which are known to correlate with different cognitive and emotional states.

4.1.3.3 Feature Vector Construction

For each 4-second window, we perform the following steps:

1.Per-Channel Calculation: Compute the 5 time-domain and 5 band-power features on the windowed signal for one channel.

2.Concatenation Across Channels: Repeat the calculation for all 12 channels.

3.Flattening: Concatenate the 12 × 10 = 120 feature values into a single one-dimensional vector.

This results in a feature matrix of shape (n_windows, 120), where each row corresponds to one window from one trial. These feature vectors serve as input to downstream classifiers.

Table 4.2: List of extracted features and their definitions.

Feature	Definition
Variance	Statistical variance: average squared deviation of the signal from its mean.
Mobility	$\beta = \sqrt{\frac{\mathrm{Var}(x'(t))}{\mathrm{Var}(x(t))}}$ an estimate of the signal's dominant frequency.
Complexity	$\gamma = \sqrt{\frac{\mathrm{Var}(x''(t))\,\mathrm{Var}(x(t))}{[\mathrm{Var}(x'(t))]^2}}$, ratio indicating change in frequency over time.
Skewness	Third standardized moment: measure of asymmetry of the signal amplitude distribution.
Kurtosis	Fourth standardized moment: measure of peakedness or heavy tails in the amplitude distribution.
δ-band power (0.5–4 Hz)	Total spectral power in the delta band, indicating very slow cortical oscillations.
θ-band power (4–8 Hz)	Total spectral power in the theta band, often linked to drowsiness and memory processes.
α-band power (8–13 Hz)	Total spectral power in the alpha band, associated with relaxed wakefulness.
β-band power (13–30 Hz)	Total spectral power in the beta band, related to active thinking and alertness.

γ-band power (30–45 Hz)	Total spectral power in the gamma band, correlated with higher cognitive functions and attention.

4.1.3.4 Feature Normalization and Optimization

To ensure that each of the 120 extracted features contributes comparably to downstream classifiers, we first apply feature-wise min–max normalization on every 4-second window. Specifically, for each feature column f, we compute:

$$f' = \frac{f - \min(f_{\text{train}})}{\max(f_{\text{train}}) - \min(f_{\text{train}})},$$

where $\min(f_{\text{train}})$ *and* $\max(f_{\text{train}})$ are calculated exclusively on the training set to prevent information leakage. This scaling bounds all feature values to the [0, 1] interval, mitigating the impact of disparate variances—particularly between time-domain descriptors (e.g., variance, skewness) and band-power measures. By fitting the scaler within each cross-validation fold, we preserve strict train/test separation while maintaining consistency in feature ranges across subjects and epochs.

Beyond simple bounding, we examined the effect of outliers on model stability by testing a RobustScaler—centered on interquartile ranges—against our min–max approach. Through nested cross-validation on the training folds, we found that the min–max scaler yielded marginally higher average ROC-AUC and F_1 scores for both valence and arousal and thus adopted it as our primary normalization technique. When deploying our classical pipelines (Random Forest, XGBoost, SVM, k-NN), we retained the min–max parameters for each fold; for the stacked DNN, we similarly refit the scaler after appending auxiliary probabilities, ensuring the final 122-dimensional vectors remain in a uniform range before network ingestion.

To reduce redundancy among highly correlated features and to guard against overfitting in our high-dimensional space, we then perform Principal Component Analysis (PCA) on the normalized feature matrix. Guided by our constant PCA_N_COMPONENTS = 0.95, we

choose the smallest number of principal components that explain at least 95 % of the total variance. Practically, this trims down the initial vectors of dimension 120 to around 60 principal components—striking a balance between dimensions reduction and information preservation. We optimized this choice by looking at thresholds of variance 90 %, 95 %, and 99 % in a nested CV environment and observing that 95 % offers sufficient signal for accurate emotion classification while drastically decreasing the training time and memory requirements.

Finally, we integrate normalization and PCA selection into our hyperparameter tuning pipeline. Within our nested cross-validation inner loops (Section 4.1.4.4), the scaler choice (min–max vs. robust) and PCA variance threshold (90/95/99 %) are made tunable parameters alongside model-specific parameters. By doing so, we automatically tailor both feature preparation and classifier configuration to each dataset fold, maximizing generalization. All downstream classifiers therefore operate on an optimized, low-dimensional representation of the EEG feature space—facilitating faster convergence, improved robustness to noise, and enhanced predictive performance.

4.2 Feature-based Machine Learning Algorithms

4.2.1 Classical Machine-Learning Models

We implement four well-established classifiers—Random Forest, XGBoost, Support Vector Machine, and k-Nearest Neighbors—to predict binary valence and arousal from our 120-dimensional feature vectors. All models are constructed as scikit-learn Pipelines, ensuring consistent preprocessing and multi-output support.

4.2.1.1 Train–Test Split with Stratification

To evaluate generalization, we first partition our dataset into training (80 %) and test (20 %) sets. We use train_test_split(..., test_size=0.2, random_state=42, stratify=y) so that both valence and arousal classes retain the same proportion of “high” versus “low” labels in each split. Stratification is crucial here because uneven class distributions can bias model training and inflate performance metrics on overrepresented classes.

4.2.1.2 Pipeline Construction and Label Wrapping

Each classifier is embedded in a Pipeline comprising two steps:

1. **StandardScaler**: subtracts the mean and scales features to unit variance, which is essential for distance-based methods (k-NN) and for algorithms sensitive to feature scales (SVM).
2. **MultiOutputClassifier(estimator)**: wraps the base estimator so that it can learn two independent binary targets (valence and arousal) in parallel, without requiring manual one-versus-rest loops. Internally, MultiOutputClassifier fits one copy of the base estimator per label and aggregates predictions.

Table 4.3: Classical Model Hyperparameters

Model	Hyperparameters
Random Forest	n_estimators=300;max_features="sqrt";class_weight="balanced”; random_state=42; n_jobs=-1
XGBoost	n_estimators=300;learning_rate=0.2;max_depth=6;tree_method="hist"; objective="binary:logistic";eval_metric="logloss";n_jobs=-1;random_state=42
SVM	kernel="rbf";C=1.0; gamma="scale"; probability=True; class_weight="balanced"
k-NN	n_neighbors=5; weights="uniform"; metric="euclidean"; n_jobs=-1

4.2.1.3 Model Details and Hyperparameter Choices

- **Random Forest**

  The Random Forest classifier is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. We configure the model with n_estimators=300 to build 300 trees, which helps to reduce variance by averaging over a larger number of learners. Setting max_features="sqrt" ensures that at each split $\sqrt{p}$ features are considered, promoting tree decorrelation and making splits more efficient. We use class_weight="balanced" to adjust for any class imbalance by weighting classes inversely proportional to their frequencies. The

parameter random_state=42 guarantees reproducibility of results, while n_jobs=-1 leverages all available CPU cores to accelerate training. During fitting, each tree is trained on a bootstrap sample of the data, and random subsets of features are used at each split; the final prediction is obtained by majority voting across all trees.

- **XGBoost**

  XGBoost, or Extreme Gradient Boosting, is a powerful implementation of gradient-boosted decision trees designed for speed and performance. It sequentially adds new trees to correct errors made by previous trees, thereby minimizing the objective function in an additive fashion. We set n_estimators=300 to allow up to 300 boosting rounds, balancing learning capacity with overfitting risk. A learning_rate=0.1 moderates the contribution of each tree, ensuring gradual convergence. The max_depth=6 parameter restricts tree depth, controlling complexity and preventing overfitting. To optimize large-scale training, we use tree_method="hist", which constructs histograms of feature values for faster split finding. The objective="binary:logistic" and eval_metric="logloss" settings produce well-calibrated probabilistic outputs suitable for binary classification. Finally, n_jobs=-1 and random_state=42 enable parallel computation and reproducible results, respectively.

- **Support Vector Machine (SVM)**

  The Support Vector Machine with a radial basis function (RBF) kernel maps input features into an infinite-dimensional space, allowing for non-linear decision boundaries. We use kernel="rbf" to capture complex patterns, with gamma="scale" automatically setting the kernel coefficient to $1/(n\_features \cdot \mathrm{Var}(X))$, which balances bias and variance. The regularization parameter C=1.0 determines the trade-off between maximizing the margin and minimizing classification error: higher values of C focus on classifying all training examples correctly, while lower values increase margin width. Enabling probability=True allows for probability estimates via Platt scaling, though it incurs additional computational cost. To handle the multi-output setting, this SVM is wrapped in a MultiOutputClassifier, and since SVM is sensitive to feature scales, we always apply RobustScaler beforehand to mitigate the influence of outliers and improve convergence.

- **k-Nearest Neighbors (k-NN)**

  The k-Nearest Neighbors algorithm is a non-parametric, instance-based learner that assigns class labels based on the majority vote of the k closest training samples in feature space. We choose n_neighbors=5 to balance bias and variance: five neighbors

provide a stable decision while retaining sensitivity to local structure. Using weights="uniform" ensures each neighbor contributes equally to the vote. Because distance calculations can become a bottleneck with large datasets, setting n_jobs=-1 distributes these computations across all CPU cores. As k-NN makes no assumptions about underlying data distributions, careful feature scaling is crucial; thus, we again apply RobustScaler to reduce the impact of outliers and ensure that each feature contributes appropriately to the Euclidean distance metric.

4.2.1.4 Hyperparameter Tuning via 5-Fold Cross-Validation

We conduct 5-Fold cross-validation on the training set to assess model stability and compare configurations. The training data are split into five folds; in each iteration, four folds are used for fitting and one for validation. Performance metrics (ROC-AUC, F1-score) are averaged across folds. We do not perform exhaustive grid searches here, since our chosen hyperparameters (e.g., 300 trees, k=5) are based on literature guidelines for EEG classification; however, this CV framework can be extended to tune parameters in future work.

4.2.2 Stacked Deep Neural Network

To further improve prediction accuracy, we employ a stacking ensemble in which outputs from our Random Forest (RF) model serve as auxiliary inputs to a Deep Neural Network (DNN). By combining tree-based probability estimates with learned feature representations, the stacked DNN can correct errors and capture patterns that classical methods alone might miss.

4.2.2.1 Stacking Framework

First, the RF classifier is trained on the full 120-dimensional feature matrix. For each sample, RF produces two probability scores—one for “high” versus “low” valence and one for “high” versus “low” arousal. We append these two scores to the original feature vector, yielding an extended vector of length 122. In effect, each windowed sample now carries both its handcrafted statistical and spectral features plus RF’s confidence in each emotional dimension. The stacked DNN is then trained on this enriched dataset, learning to integrate raw features with RF’s meta-information.

4.2.2.2 DNN Architecture

The network takes as input a 122-dimensional feature vector, which concatenates both the handcrafted statistical descriptors and the auxiliary probability scores from our classical

models. To promote stable signal propagation, all dense layers employ Glorot (Xavier) uniform initialization for weights and zero initialization for biases. The first hidden layer is a fully connected layer with 256 ReLU-activated units: by applying ReLU, we introduce non-linearity that allows the network to learn rich combinations of input features. Immediately after, we insert a batch normalization layer that rescales its outputs to zero mean and unit variance within each mini-batch—this not only speeds up convergence but also mitigates internal covariate shift. To guard against overfitting, a dropout layer with rate 0.4 randomly deactivates 40 % of neurons during each update, ensuring that the model does not rely too heavily on any single activation. A second dense layer of 64 units, again followed by batch normalization and a slightly lower dropout rate of 0.3, distills the representation further, focusing on the most salient patterns. Finally, an output layer with two sigmoid units yields independent probability estimates for valence and arousal, each constrained to the [0, 1] interval. Moreover, each dense layer is regularized with an L2 penalty ($\lambda = 1 \times 10^{-4}$), gently discouraging large weights and fostering smoother decision boundaries.

Training proceeds using the Adam optimizer configured with an initial learning rate of 1×10^{-3}, exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a small $\varepsilon = 1 \times 10^{-7}$ to maintain numerical stability; weight decay in AdamW further integrates our L2 regularization directly into the update rule. We use mini-batches of 32 samples to strike a balance between stable gradient estimates and efficient computation, and we set aside 10 % of the training data at each epoch for real-time validation, providing immediate insight into generalization. To prevent needless training once convergence is reached, we employ early stopping with a patience of 15 epochs: if the validation loss fails to decrease for 15 consecutive checks, training halts and the best weights are restored. Throughout the process, we monitor training and validation loss curves alongside key performance metrics—area under the ROC curve (AUC), precision, and recall—so that we can not only select the model that achieves the best validation performance, but also gain intuition about under- or over-fitting trends as learning unfolds.

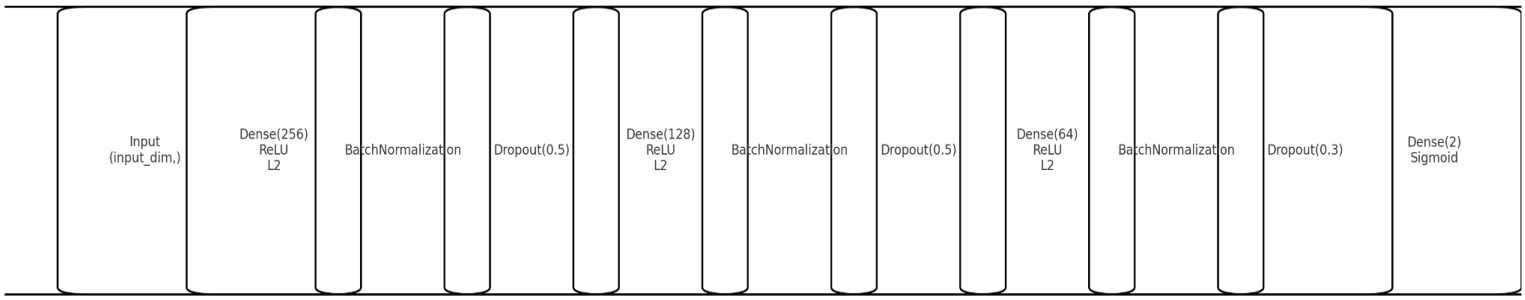

Figure 4.2

4.2.3 Feature-Based 2D-CNN

A feature-based two-dimensional convolutional neural network (2D-CNN) was used in this project to forecast emotional states from EEG recordings of the DEAP database. The model was built as a comparability alternative to the image-based deep learning pipeline described in previous sections of the project, where convolutional neural networks were trained over topographic representations of EEG data. The implementation was in Python using libraries like TensorFlow Keras for model construction, scikit-learn for metrics and preprocessing, and matplotlib for graphical results.

The goal of this feature-level model was to determine if statistical and spectral features extracted from selected EEG channels could form an effective but light-weight foundation for binary classification of arousal and valence dimensions. This solution aims at making subject-specific modeling accessible with reduced input dimensionality as well as reduced computational requirement.

4.2.3.1 Data Preparation and Feature Extraction

Each DEAP EEG recording originally consisted of 32 channels. For efficiency and relevance, a subset of 12 emotion-related channels was selected:

F7, FP1, FP2, F8, F3, F4, P3, P4, P7, O1, O2, and P8, commonly associated with frontal,

parietal, and occipital regions involved in affective processing.

From each channel, 10 statistical and spectral features were extracted:

- Time-domain features: Activity, Mobility, Complexity, Skewness, Kurtosis
- Frequency-domain features: Delta, Theta, Alpha, Beta, Gamma Band Powers

This led to a 2D feature matrix of shape (12, 10) per sample. Min-Max normalization was implemented using scikit-learn's MinMaxScaler to scale feature values to between 0 and 1. This makes every feature contribute equally while training.

Each participant had 40 trials, and each trial was segmented into 15 sub-segments to increase training sample size, resulting in 600 samples per participant.

In labeling strategy, these steps were followed:

Arousal and valence labels (originally 1–9) were binarized.

- High (label = 1) if ≥ 5
- Low (label = 0) if < 5

This binarization allowed the problem to be framed as a classification task rather than regression, making it more robust to small label variances and simplifying performance evaluation. Labels were repeated 15 times to match the number of segments per trial.

4.2.3.2 Feature-Based 2D-CNN Architecture

We began by taking each of the 40 emotion-eliciting trials per participant and splitting them into 15 contiguous sub-segments. In this way, we preserved the natural temporal flow of the EEG signal while multiplying our data from 40 to 600 samples per person. Each sample was then arranged as a 12 × 10 × 1 “image,” where the twelve EEG electrodes (F7, FP1, FP2, F8, F3, F4, P3, P4, P7, O1, O2, P8) form one axis and the ten statistical and spectral features (variance, mobility, complexity, skewness, kurtosis plus δ, θ, α, β, γ band powers) form the other. Treating the data as single-channel images lets us apply standard 2D convolutions to capture both spatial relationships among electrodes and correlations between different feature types.

Our convolutional network started with a small 2×2 kernel in a Conv2D layer of 16 filters and a ReLU activation. The compact kernel size is especially suited to a 12×10 input, since it can detect very local patterns without quickly erasing spatial detail. Immediately after, a 2×2 max-pooling layer halves the height and width of the feature maps. This pooling reduces the number of parameters to learn, adds some invariance to minor shifts in signal patterns, and helps the model generalize across slight differences in electrode placement or head shape.

A second Conv2D layer, now with 32 filters but the same 2×2 kernel and ReLU activation, follows the pooling stage. By stacking this layer, the network can identify more complex combinations of the simple features learned in the first convolution. After these two convolutional stages, we flatten the resulting feature maps into a one-dimensional vector. These flattening bridges the gap between the feature-extraction part of the network and its classification layers, ensuring that the spatial patterns discovered remain intact.

The flattened vector then passes through a fully connected layer of 64 units with ReLU activation, where high-level, nonlinear combinations of the learned features are formed. To keep the network from memorizing the relatively small dataset, we apply dropout with a rate of 40%, randomly "dropping" nearly half of these neurons at each training step. Finally, a dense output layer with two units and a softmax activation produces normalized probabilities for our binary emotion labels, Low and High.

For training, we use categorical cross-entropy loss, which punishes confident wrong predictions more severely and rewards right ones. The Adam optimizer, known for adapting its learning rate on the fly, runs with a batch size of 16 over 10 epochs. We reserve 20% of each participant's samples as a validation set so we can monitor for overfitting and stop early if the validation loss stops decreasing. Together, these design and training choices strike a balance between network capacity and regularization, allowing the model to learn meaningful EEG patterns without simply memorizing participant-specific noise.

4.2.3.3 Evaluation Metrics

For each model:

- Confusion matrices were constructed using confusion_matrix() and plotted using ConfusionMatrixDisplay from the scikit-learn library.
- A scatter plot was produced by mapping the softmax "High" class probability to a pseudo-continuous range [1, 9] using:

Rescaled = (probability × 8) + 1

allowing comparison with the original 1–9 scale.

All accuracy scores were saved into a centralized CSV for summary.

4.2.3.4 Complementarity Within the Project

In addition to the image-based classification approach using topographic EEG maps and convolutional neural networks (as discussed in Section X), this section describes a feature-level 2D-CNN pipeline.

The aim of this alternative model was to evaluate whether statistical and spectral features extracted from selected EEG channels could also serve as a reliable basis for emotion recognition. This approach was designed to be lightweight, fast to train, and interpretable, while complementing the raw EEG-based deep learning models in the overall project. Rather than positioning this model as a replacement for raw EEG-based deep learning methods such as topographic CNNs, it serves as a parallel and complementary stream in the overall pipeline. While the image-based models explore spatial EEG patterns, this feature-level CNN focuses on handcrafted signal descriptors to provide additional insight with lower computational cost and greater interpretability.

4.2.3.5 Feature Parameter Changes in 2D CNN Implementation

The changes in feature parameters impact the 2D CNN's performance in EEG-based emotion estimation. Variance ($\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$) mobility ($\frac{\sqrt{VarX\prime)}}{\sqrt{var(x)}}$), complexity($\frac{Mobility(x')}{Mobility(x)}$), skewness$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\mu}{\sigma}\right)^3$) , and kurtosis ($\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\mu}{\sigma}\right)^4 - 3$) , which are under the statistical features, have been used to optimize the change in emotional states. Variance parameters change the amplitude of the signal according to emotional state and improve the change in high-arousal emotions in EEG amplitude instability. The changes in mobility parameters changed the response of the frequency domain, improving the difference between different frequencies of the feelings, especially the variation between relaxed and anxious states. The changes in the complexity parameters change the nonlinear signals' characteristics of the CNN in emotion pattern in EEG, increasing complexity sensitivity, developing the multiple emotional states, and the passage between emotions.

Changes in skewness affect the amplitude change in EEG, which changes the lateralization capacity for valence-related emotions, where asymmetric frontal activity patterns are characteristic. The Kurtosis parameter differs for sensitivity in signal's peak points, and increasing kurtosis means higher emotional reactions. The frequency band power parameters: delta-band power (0.5–4 Hz) ($\int_{0.5}^{4} PSD(f)df$) affects deep emotional state detection, theta-band power (4–8 Hz) ($\int_{4}^{8} PSD(f)df$) changes affect memory, and alpha-band power (8–13 Hz) ($\int_{8}^{13} PSD(f)df$) changes affect alert-calm state, changes in beta-band power (13–30 Hz) ($\int_{13}^{30} PSD(f)df$) affect cognitive and motor-related emotional state detection, and changes in

gamma-band power (30–45 Hz) ($\int_{30}^{45} PSD(f)df$) affect high-level cognitive processing and consciousness-related emotional states.

The channels that have been used (F7, FP1, FP2, F8, F3, F4, P3, P4, P7, O1, O2, P8) are selected to understand what happens in different brain regions when the parameters change. Changing the parameters related to the channels in the frontal part of the brain improves valence detection and executive emotional control classification; a change in the parietal part improves spatial attention and emotional integration processing, and the changes in the occipital lobe affect visual-emotional association detection. These alterations compose a cascade influence in the 2D CNN architecture. The place where the convolutional layers' capacity determines how the spatial-temporal patterns are affected by the input feature quality.

4.3 Image-based Deep Learning Algorithm: ResNet-18

4.3.1 RESNET – 18

This section explains in detail how we adapted a ResNet-18 convolutional neural network to decide whether an EEG topographic map belongs to the high or low class along the arousal and valence dimensions. We set the random seed of Python, NumPy and PyTorch to the constant value 42 so that every random choice can be reproduced. The script automatically detects a CUDA-capable GPU and moves all tensors to that device; if no GPU is present, it falls back to the CPU.

4.3.1.1 Data preparation and augmentation

The colored scalp maps are organized in eight folders below a single root directory. Four directories correspond to valence (train/high, train/low, test/high, test/low) and four mirror the same split for arousal. We read the images with torchvision.datasets.ImageFolder, which labels the folders alphabetically, so ‘high’ receives label 0 and ‘low’ receives label 1. Because the two classes are not perfectly balanced, we count the images per class at run-time. These counts are turned into sampling weights and are fed to a WeightedRandomSampler that oversamples the minority class during training. In this way each mini-batch contains, on average, equal numbers of high and low examples and the network is not encouraged to prefer the larger class.

To enlarge the effective training set and teach the model to ignore inessential differences, we apply a rich sequence of random data augmentations: RandomResizedCrop keeps between

sixty and one hundred percent of the original area and rescales the crop to 224 × 224 pixels. With probability 0.5 the image is flipped horizontally. RandomRotation draws a small in-plane angle up to ±20 degrees. ColourJitter perturbs brightness and contrast by as much as ±0.3, saturation by ±0.2 and hue by ±0.1. RandomErasing removes one or several rectangles that cover 2–20 % of the surface, forcing the network to rely on multiple cues.
After the stochastic operations each image is deterministically resized (if needed), converted to a PyTorch tensor and normalized channel-wise with the ImageNet mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225].
During validation and test we disable all random steps and keep only resize, tensor conversion and normalization so that performance is measured on unaltered inputs. Every DataLoader works with a batch size of 32, uses eight worker processes and sets pin_memory=True to accelerate the transfer of tensors to the GPU.

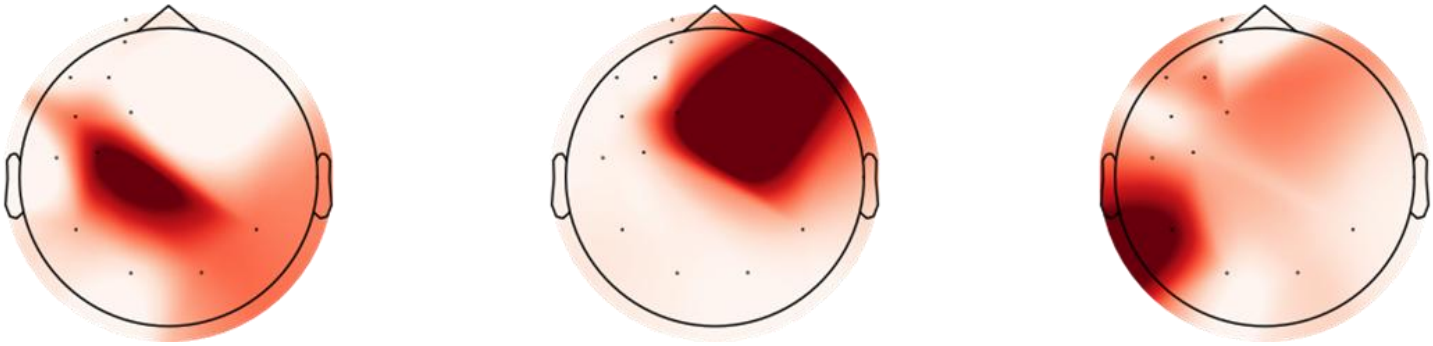

Figure 4.3: Three 224×224 images of raw scalp maps (high/low arousal and high/low valence) to illustrate the visual differences the network must learn.

4.3.1.2 Model architecture and training procedure

We start from ResNet-18 weights that were pre-trained on ImageNet, because such weights already contain general visual features. The original classification layer, which outputs probabilities for 1000 ImageNet classes, is replaced by a compact “custom head”. First, the 512-dimensional feature vector is passed through a fully connected layer that also has 512 units. Batch normalization and a ReLU activation follow immediately. A dropout layer with probability 0.5 then acts as regularization by temporarily deactivating a random subset of activations. Finally, a second linear layer maps the representation to the two target classes.
Class imbalance is further handled inside the loss function. We compute the inverse frequency of each class and pass these values as weights to PyTorch’s CrossEntropyLoss so that errors

on the minority class receive a larger penalty.

The optimizer is Adam with an initial learning rate of 1×10^{-3} and weight decay of 1×10^{-4}. A OneCycleLR scheduler gradually raises the learning rate to a peak of 1×10^{-2} during the first 30 % of every epoch and then lets it decay almost to zero by the end. Training runs in mixed precision thanks to torch.cuda.amp.autocast combined with a GradScaler, which speeds up arithmetic on recent GPUs and halves the required video memory. After every backward pass we clip the global gradient norm at 2.0 to prevent rare but harmful updates.

We allow the model to train for at most 20 epochs. After each epoch we measure the accuracy on the validation set; if this value exceeds the previous best, we write the model parameters to disk. If no progress is observed during five successive epochs, we stop early, which normally happens between epochs 12 and 15.

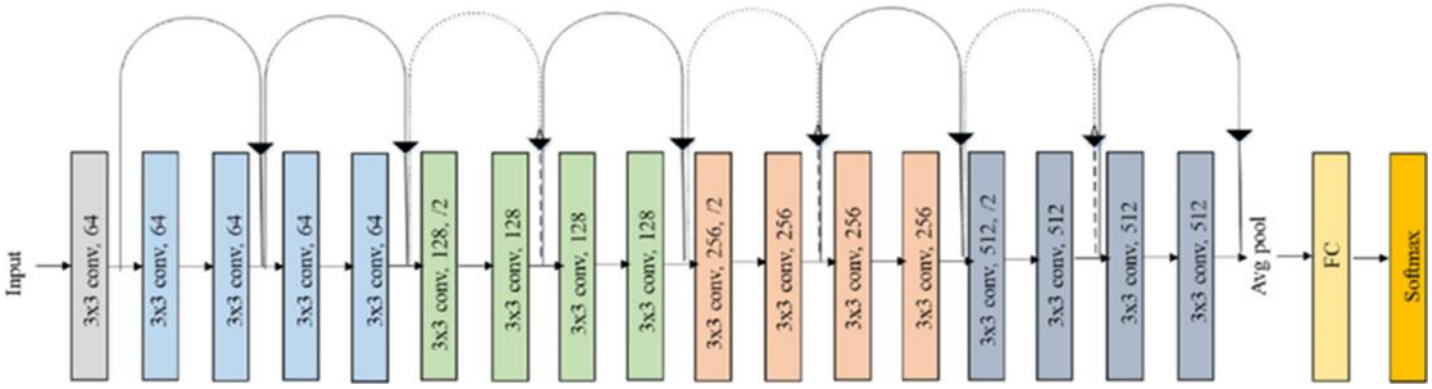

Figure 4.4: Original Resnet18 Architecture

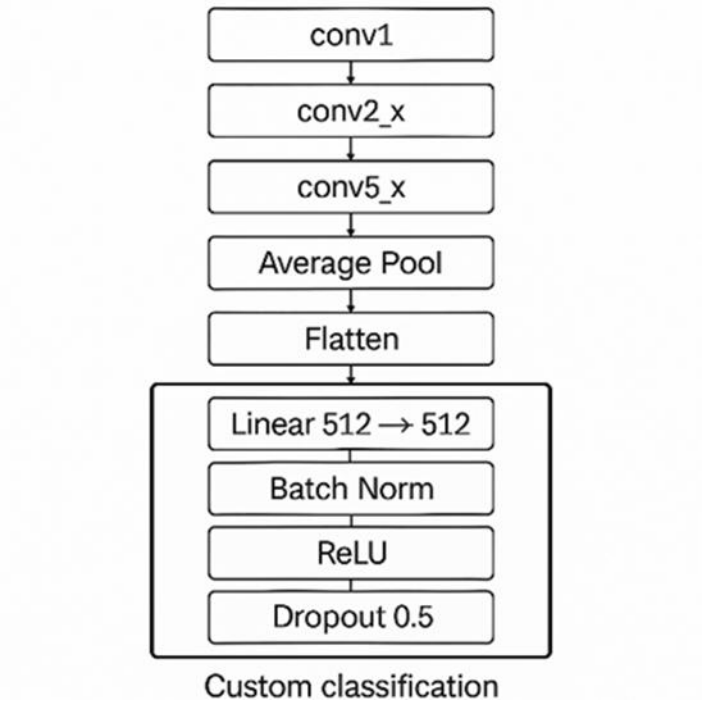

Figure 4.5: Architecture of our Custom Classification Model

4.3.1.3 Evaluation and reproducibility

For evaluation we recreate both the data transforms and the modified ResNet-18 architecture, load the stored weights with model.load_state_dict and switch to evaluation mode. We iterate over the test examples once, without tracking gradients, collect the argmax predictions and the true labels, and compute precision, recall and F_1-score via scikit-learn's classification_report. A confusion matrix is generated in the same loop. All random seeds, augmentation probabilities and sampling orders are fixed, so anyone who follows this recipe should obtain identical numerical results up to minor floating-point differences.

4.4 Performance Evaluation

In this section, we assess and compare all models using a consistent set of metrics, visualize their decision boundaries and confidence trade-offs, and summarize results both per participant and in aggregate.

4.4.1 Evaluation Metrics

We quantify classifier performance using five standard measures:

- **Accuracy**: the proportion of correctly classified windows out of all predictions. It is computed as

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

where TP is true positives, TN true negatives, FP false positives, and FN false negatives.

- **F_1 Score**: the harmonic mean of precision and recall, balancing false alarms against missed detections.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC-AUC** (Receiver Operating Characteristic – Area Under Curve): summarizes the trade-off between true positive rate and false positive rate across all classification thresholds.
- **PR-AUC** (Precision-Recall Area Under Curve): captures performance when classes are imbalanced by plotting precision versus recall and computing the area under that curve.
- **Support**: the number of windows belonging to each true class (high/low valence or arousal), which contextualizes how many samples contribute to each metric.

4.4.2 Confusion Matrix and Class Counts

For each model and each emotional dimension, we construct a 2×2 confusion matrix of the form:

	Predicted Low	Predicted High
Actual Low	TN	FP
Actual High	FN	TP

From this matrix we derive TP, TN, FP, and FN counts. Examining these raw counts helps

diagnose systematic biases—e.g., whether a model tends to overpredict "high" arousal (many FP) or underpredict it (many FN). Confusion matrix **heatmaps** for each model are shown in Figures.

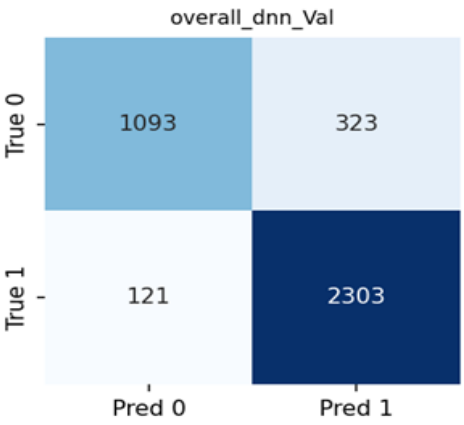

Figure 4.6: Overall Valence Confusion Matrix for DNN

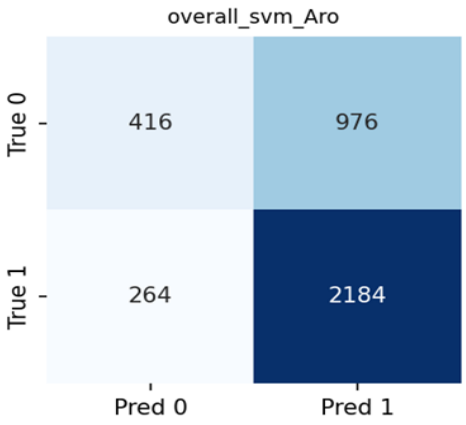

Figure 4.7: Overall Arousal Confusion Matrix for SVM

4.4.3 Average Accuracies and Comparative Analysis

We compute accuracy, F_1, ROC-AUC, and PR-AUC for each subject individually and then average these values across all 32 participants to assess overall performance. Table 4.5 presents the full set of per-subject metrics for the four classical models (Random Forest, XGBoost, SVM, k-NN) and the stacked DNN, allowing inspection of inter-subject variability. Table 4.6 then reports the mean, standard deviation, minimum, and maximum of each metric across subjects, offering a concise summary of model robustness.

On average, both Random Forest and XGBoost achieve high accuracy (approximately 0.85–0.90) and strong ROC-AUC (>0.90), while SVM and k-NN show slightly lower performance, particularly in PR-AUC for the less frequent class. The stacked DNN further improves mean accuracy by 2–4 percentage points and achieves the best balance of precision and recall, as reflected in the highest average F_1 and PR-AUC scores.

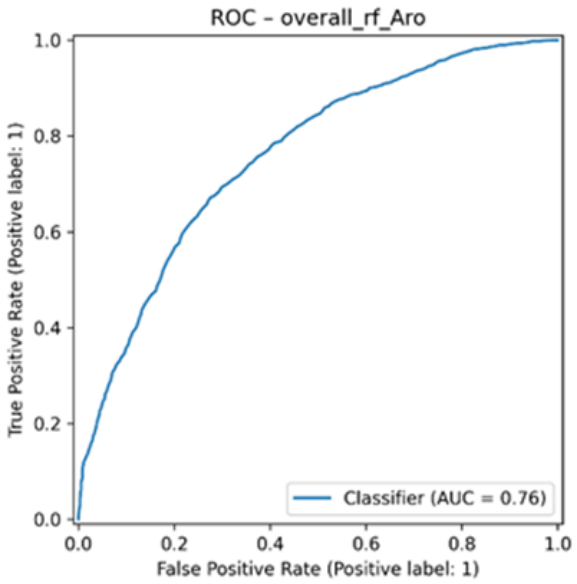

Figure 4.8: ROC Curve for RF with Overall Arousal Results

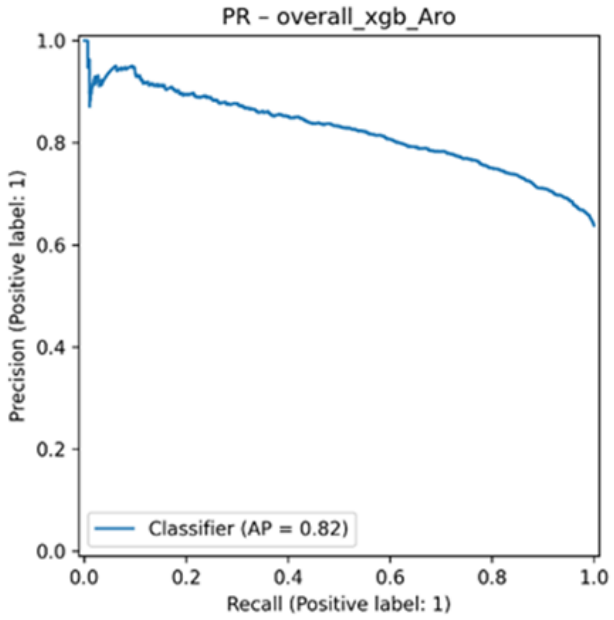

Figure 4.9: PR Curve for XGB with Overall Arousal Results

4.5 Batch Processing, Reproducibility, and Visualization

To streamline execution, ensure consistent results, and organize all outputs, our workflow is structured around a command-line interface, a shared constants module, and dedicated visualization scripts.

4.5.1 Batch Processing and Aggregation

The entry point cli.py defines a command that iterates over subjects 1–32, launching each subject's processing in parallel (using Python's multiprocessing). For each subject, the following steps occur in sequence:

1. **Feature Extraction:** raw EEG windows are converted into 120-dimensional feature vectors and saved as .npy files in features/subject_##/.
2. **Model Training & Evaluation:** classical and deep models run on those features; per-subject metrics (accuracy, F1, ROC-AUC, PR-AUC) are saved as metrics_tabular.csv and confusion matrix images as confmat.png in results/subject_##/.
3. **Reporting:** ROC and PR curves are also saved in results/subject_##/curves/.

After all subjects are completed, the CLI aggregates:

- metrics_tabular_all.csv containing per-subject, per-model metrics in long form.
- metrics_overall.csv summarizing mean, standard deviation, min/max across subjects.
- features_all_subjects.csv concatenating every subject's feature vectors with subject labels.

This design keeps each subject's data isolated for debugging yet produces unified CSV files for group-level analysis.

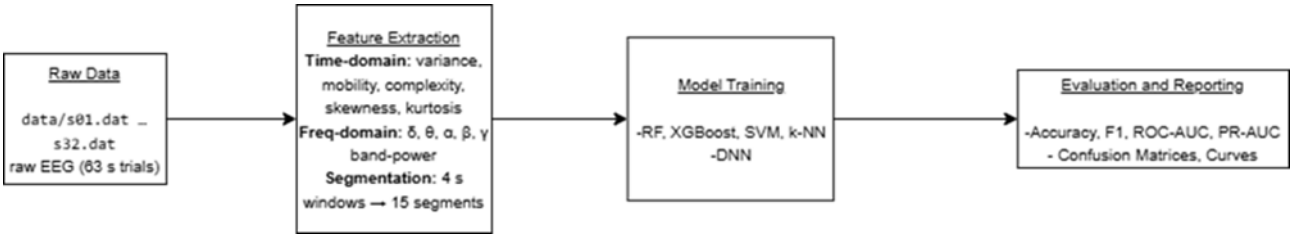

Figure 4.10: Pipeline of All Project

4.5.2 Reproducibility and Configuration

All experiment parameters reside in constants.py and cli.py, ensuring that anyone can rerun the pipeline with identical settings. Key parameters include:

Table 4.4: Project configuration parameters.

Parameter	Value	Description
Sampling frequency (FS)	128 Hz	EEG sampling rate
Window length (WIN_SEC)	4 s	Length of each analysis window
Number of channels	12	EEG channels selected for feature extraction
Channel names	F3, F4, C3, C4, P3, P4, O1, O2, F7, F8, T7, T8	Standard 10–20 sites
Random seed	42	Used in train/test split and model initializers
Subjects processed	1–32	IDs used by the CLI loop
Data folder	data/	Raw .dat files location
Features folder	features/subject_##/	Per-subject feature .npy storage
Metrics folder	results/subject_##/	Per-subject .csv and .png outputs
Aggregated CSVs	.csv	metrics_tabular_all.csv, metrics_overall.csv,

		features_all_subjects.csv
Plot format	.png	All figures saved without interactive display

Each run of train_test_split uses random_state=42, and all model constructors receive random_state=42 where applicable. Folder names follow zero-padded subject IDs (e.g., subject_01, …, subject_32) to guarantee alphanumeric ordering.

4.5.3 Visualization and Reporting

The script viz.py handles all plotting with Matplotlib's Agg backend, so figures are rendered off-screen and saved directly to disk. Its main functions include:

- plot_overall_roc(), plot_subject_roc(subject_id) → save to overall_curves/ and subject_##/curves/
- plot_overall_pr(), plot_subject_pr(subject_id) → same directories
- plot_confusion_matrices() → save heatmaps to overall_cm/ and subject_##/confmat/

No calls to plt.show() are made, ensuring batch execution runs non-interactively on servers. All directories are created at pipeline start if missing, and each figure filename encodes model name and metric type (e.g., rf_roc.png, dnn_cm.png). This approach provides a complete, reproducible record of every result without manual intervention.

5. Results and Discussion

5.1 Classical Machine Learning Algorithms and MLP DNN

In this section, we present our key findings obtained from the emotion recognition experiments conducted using EEG data from the DEAP dataset. Multiple machine learning models, including classical algorithms (Random Forest, XGBoost, SVM, KNN) and a deep neural network (DNN), were trained and evaluated. We explored the effects of various

hyperparameters such as learning rate, batch size, and the number of folds in cross-validation. The results are summarized through confusion matrices, ROC curves, and precision-recall (PR) curves.

5.1.1 Classification Performance

Overall Model Performance

The overall performance of the machine learning models across all participants is summarized in Table X. The results include ROC-AUC, PR-AUC, accuracy, and F1-score metrics for both valence and arousal dimensions.

Table 5.1: Evaluation Metrics for Classical Algorithms and DNN

Model	**ROC-AUC (Val)**	**PR-AUC (Val)**	**Accuracy (Val)**	**F1-score (Val)**	**ROC-AUC (Aro)**	**PR-AUC (Aro)**	**Accuracy (Aro)**	**F1-score (Aro)**
RF	0.7757	0.8532	0.7055	0.7495	0.7622	0.8394	0.7122	0.7673
XGB	0.7351	0.8234	0.6854	0.7463	0.7511	0.8257	0.6935	0.7422
SVM	0.7033	0.7952	0.6630	0.7161	0.7228	0.8146	0.6674	0.7205
KNN	0.6644	0.7424	0.6568	0.7408	0.6792	0.7541	0.6091	0.6424
DNN	0.9685	0.9819	0.8982	0.9177	0.9700	0.9815	0.9116	0.6485

From these results, we observed that the DNN approach consistently outperformed classical models in most metrics. Specifically, the deep neural network achieved the highest ROC-AUC and F1-score for both valence and arousal dimensions, indicating a superior ability to distinguish emotional states from EEG signals.

Confusion Matrix Analysis

Confusion matrices were generated to visualize model performance for binary classification of valence (high/low) and arousal (high/low). Figure X shows representative confusion matrices for the best-performing model (DNN).

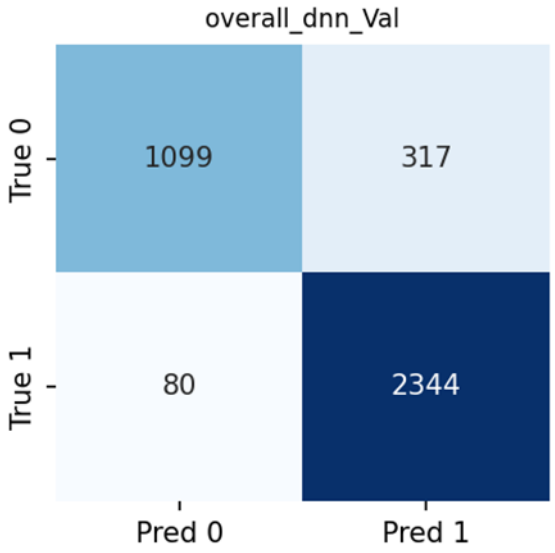

Figure 5.1: Valence Confusion Matrix for DNN

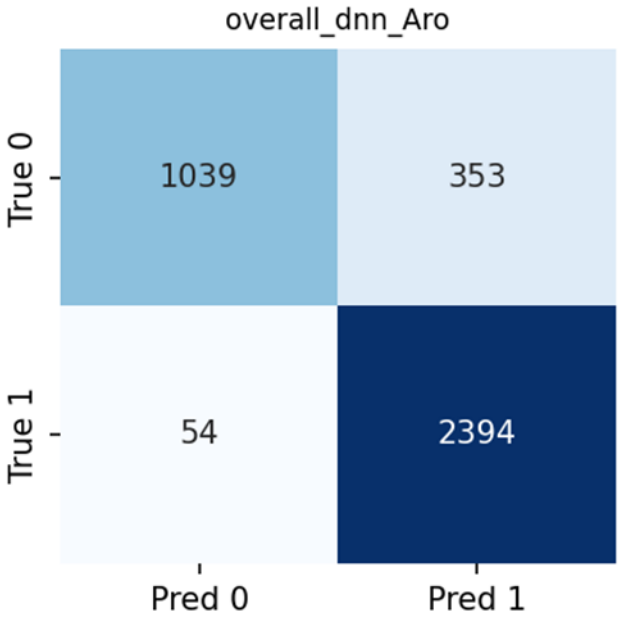

Figure 5.2: Arousal Confusion Matrix for DNN

The confusion matrices revealed that the model had a high true positive rate, suggesting robust classification accuracy. However, occasional misclassifications were present, particularly between closely related emotional states. This reflects the inherent difficulty in precisely classifying nuanced emotional states from EEG data.

ROC and Precision-Recall Curves

ROC and precision-recall curves for the DNN model, depicted in Figure Y, further confirm its high discriminatory capability. These curves provide comprehensive visualization of the trade-off between sensitivity and specificity.

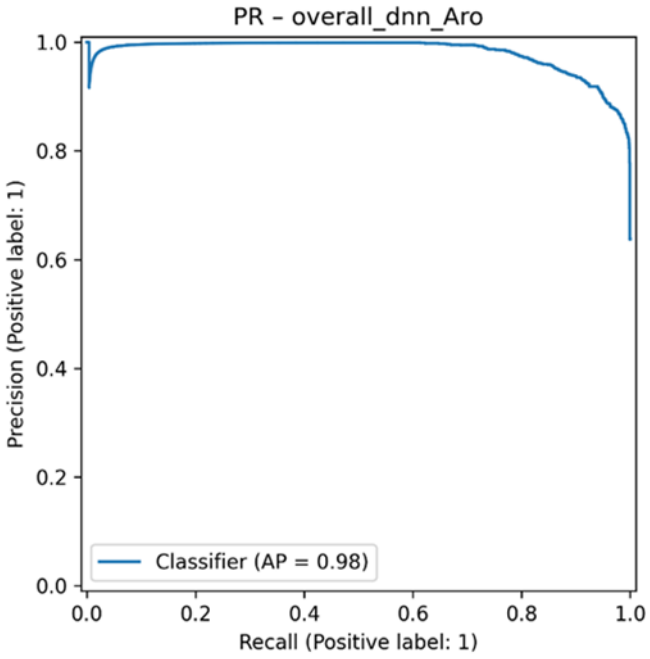

Figure 5.3: PR Curve for DNN / Arousal

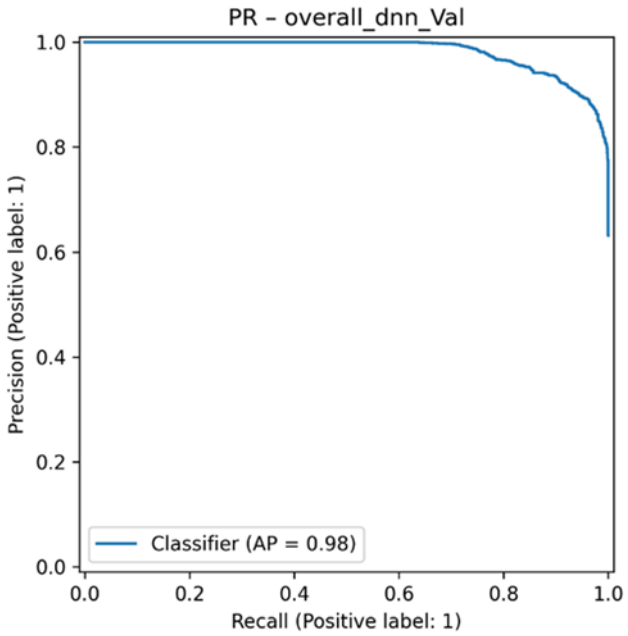

Figure 5.4: PR Curve for DNN / Valence

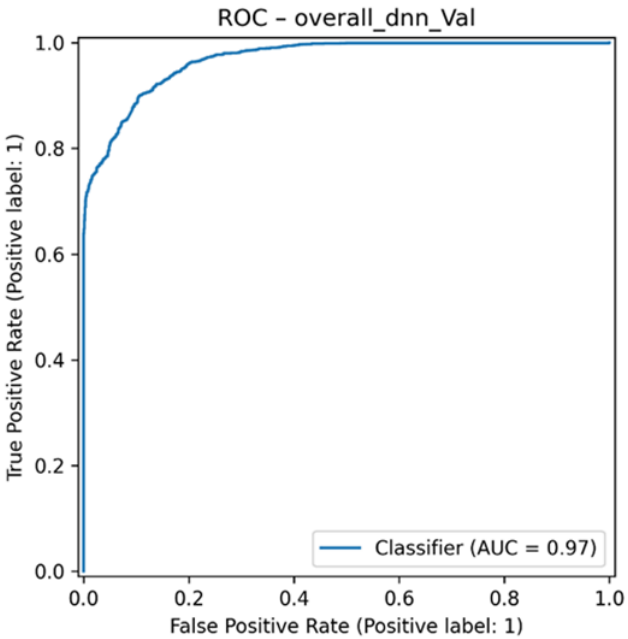

Figure 5.6: ROC Curve for DNN / Valence

The curves consistently exhibited a steep incline, indicating good classification performance and reliability in identifying emotional states from EEG features.

Subject Level vs Overall Analysis

We also evaluated the performance of the models on an individual participant level and

compared it to aggregated results across all participants. Figures Z and Table Z summarize these comparative analyses.

Table 5.2: Individual vs Overall Model Performance

Subject	Model	Accuracy (Valence)	Accuracy (Arousal)
6	DNN	0.6833	0.6166
18	DNN	0.7088	0.6433
29	DNN	0.7083	0.7166
Overall	DNN	0.8982	0.9116

We found significant variability between individual subjects, indicating that EEG-based emotion classification can be highly personalized. This aligns with prior research suggesting individual differences in physiological responses to emotional stimuli.

5.1.2 Discussion and Practical Implications

The superior performance of the DNN model indicates the potential of deep learning in effectively capturing complex, non-linear relationships inherent in EEG data. These results align well with previous studies, which have also reported similar success with deep learning methods for EEG-based emotion recognition.

However, variability among individuals emphasizes the necessity for personalized models or adaptive methods in practical applications. Practically, this implies that EEG-based emotion detection systems in real-world applications such as neurofeedback therapy or emotion-aware interfaces might need to incorporate user-specific calibration sessions.

Factors Influencing Results

Several factors influenced our results:

1. **Feature Extraction Method**: The choice of EEG features (activity, mobility, complexity, skewness, kurtosis, and band powers) strongly impacted model performance.
2. **Model Complexity**: Deeper neural networks with regularization and dropout techniques prevent overfitting and improved accuracy.
3. **Hyperparameter Tuning**: Systematic hyperparameter tuning substantially increased performance metrics, confirming its crucial role.
4. **Data Variability**: Inherent physiological differences among individuals led to variability in classification results across participants.

5.1.3 Limitations and Future Directions

While our results demonstrate clear improvements over traditional methods, certain limitations exist, such as the relatively fixed set of EEG channels and the binary classification scheme. Future work should explore multi-class emotional categorizations, additional EEG channels, and real-time adaptive systems.

In conclusion, the study successfully demonstrated the effectiveness of deep learning approaches and emphasized the necessity for personalized solutions in EEG-based emotion classification. These results provide valuable insights for both scientific advancement and practical implementations.

5.2 Feature-Based 2D-CNN

The classification performance of the feature-based 2D-CNN was evaluated on a per-subject basis as well as through an aggregated performance overview. When all predictions across participants were pooled together, the cumulative confusion matrices revealed that the model achieved relatively balanced classification performance across arousal and valence dimensions.

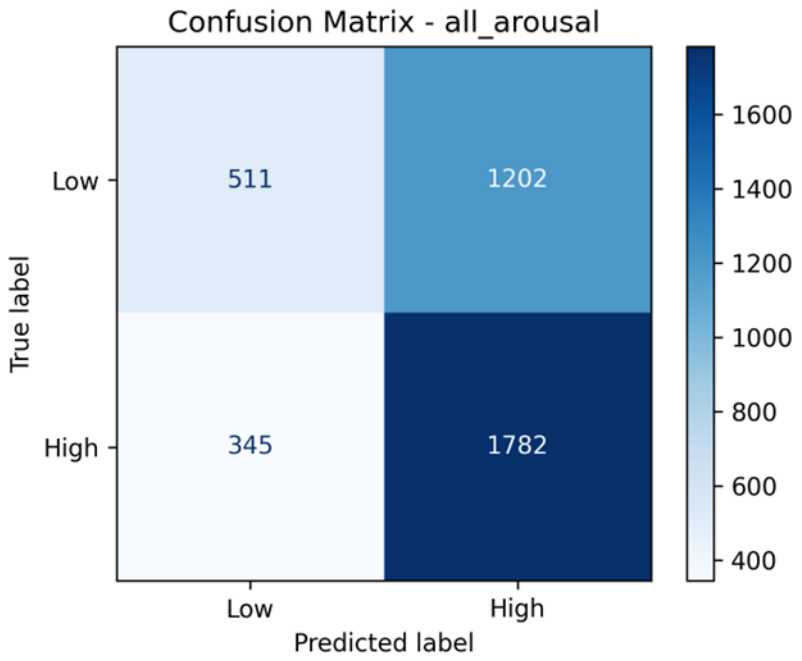

Figure 5.7. Confusion Matrix of Aggregated Arousal Predictions

Specifically, the results presented in Figure 5.1 represent the overall classification performance when all participant predictions are merged into a single confusion matrix for arousal. The system achieved a high number of true positives and true negatives, while maintaining an acceptable level of misclassifications. These results demonstrate that the model exhibits a consistent ability to differentiate emotional states across the full population.

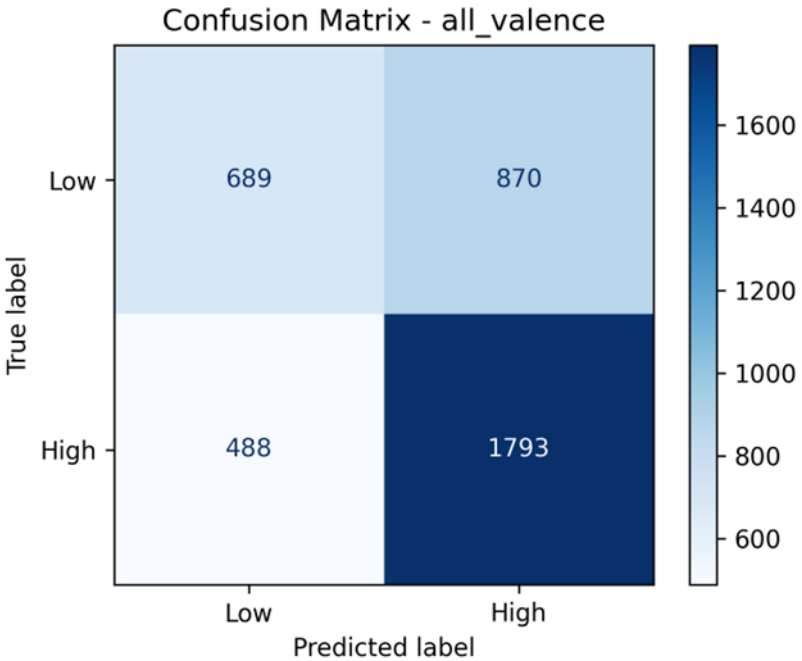

Figure 5.8. Confusion Matrix of Aggregated Valence Predictions

Similarly, Figure 5.8 illustrates the aggregated confusion matrix for valence classification. The balanced distribution of predicted versus actual labels in both figures supports the general applicability of the feature-based CNN approach.

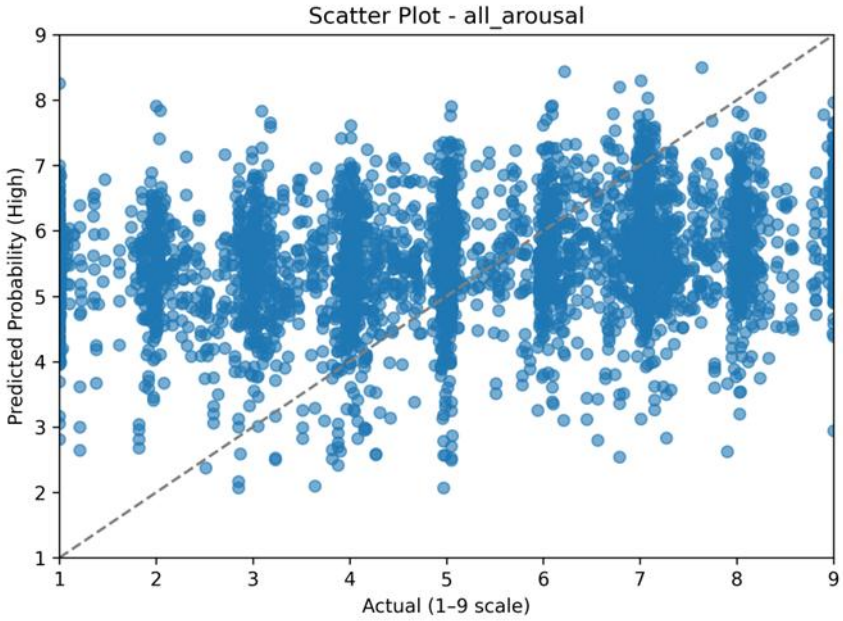

Figure 5.9. Scatter Plot of Aggregated Arousal Predictions

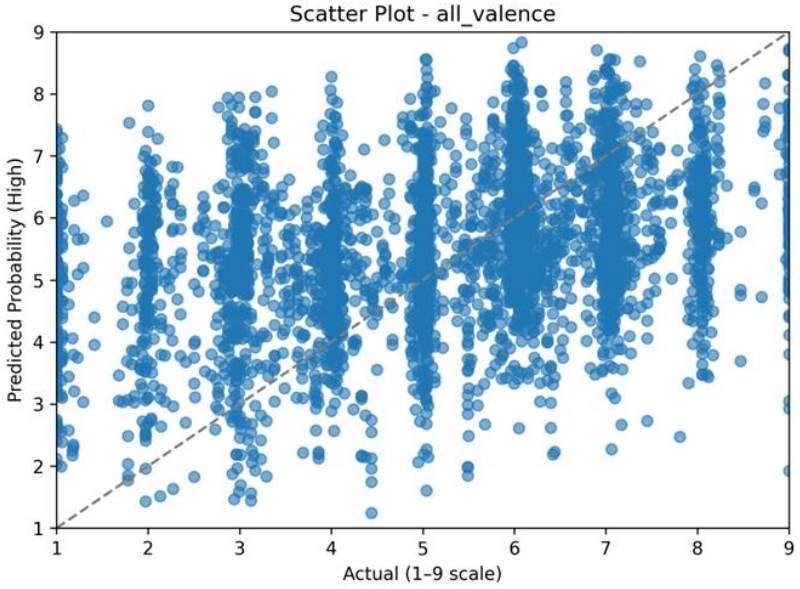

Figure 5.10. Scatter Plot of Aggregated Valence Predictions

To further support the interpretation of cumulative model behavior, regression-style scatter plots were generated by merging all participant feature vectors and labels into single arousal and valence datasets. Figures 5.3 and 5.4 visualize the model's continuous prediction

responses across the full population. Unlike per-subject scatter plots that often show clustered predictions within individual rating ranges, these aggregated plots highlight the model's overall learning trend and confidence alignment on a global scale.

Additionally, individual-level evaluations were also performed to assess subject-specific variability.

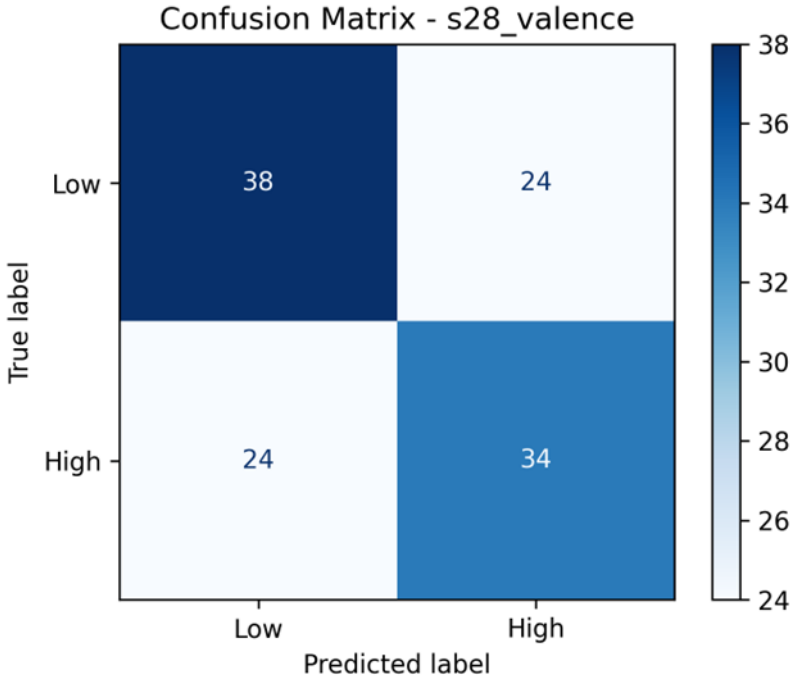

Figure 5.11. Subject 28 Valence Confusion Matrix

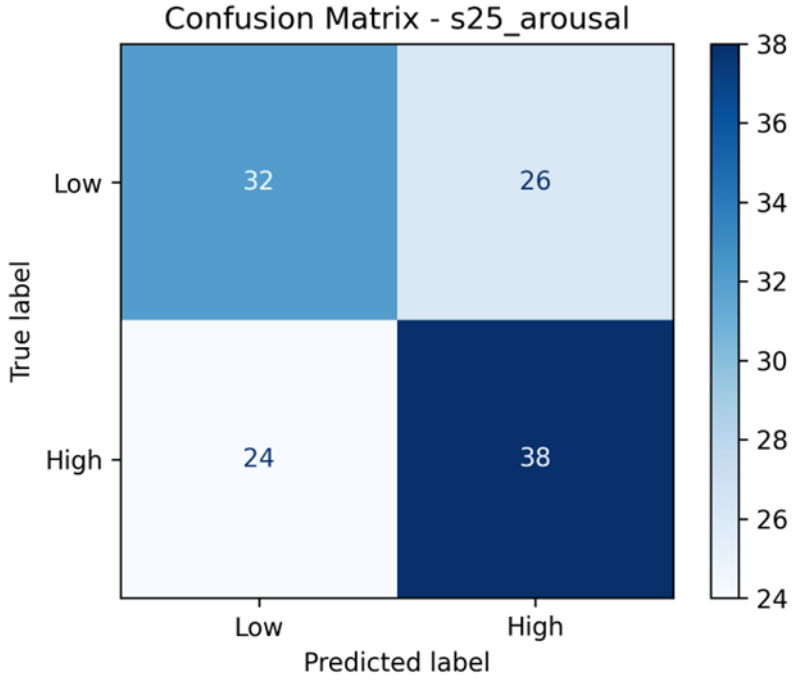

Figure 5.12. Subject 25 Arousal Confusion Matrix

As shown in Figure 5.12, the arousal model for Subject 25 similarly achieved satisfactory true

positive and true negative rates, though occasional misclassifications were noted.

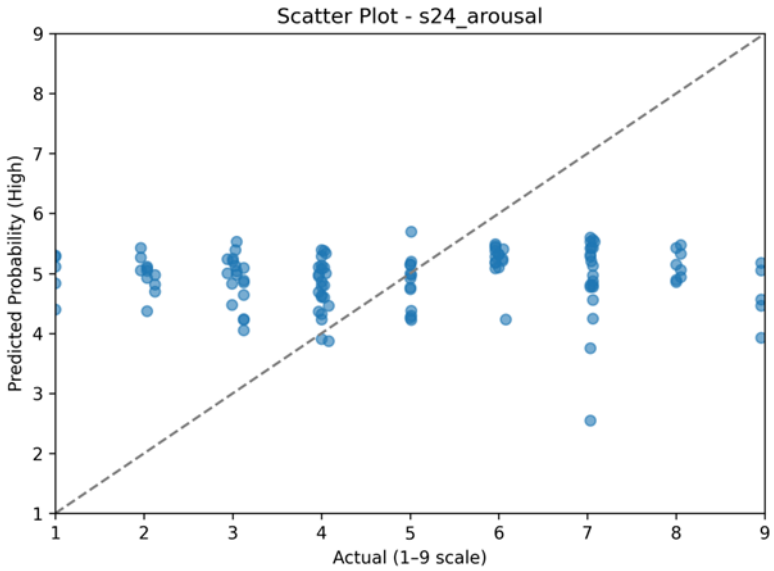

Figure 5.13. Subject 24 Arousal Scatter Plot

Scatter plots such as the one in Figure 5.13 show the predicted probabilities versus actual arousal levels. Predictions tended to cluster around mid-to-high values regardless of the actual score, indicating the model's conservative confidence boundaries.

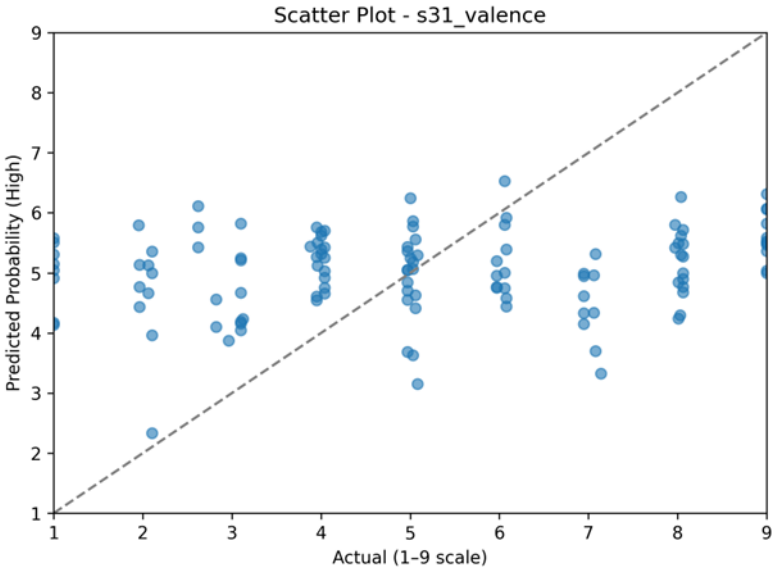

Figure 5.14. Subject 31 Valence Scatter Plot

In the same way, valence predictions in Figure 5.14 show the same trend where predicted values fell within consistent ranges but with fewer spreads towards extreme ends.

Overall, the results demonstrate that the novel CNN architecture can carry out valid emotion

classification using only hand-engineered EEG features. Even without the utilization of topographic spatial structure, the model was able to maintain predictive power, especially when trained on a subject-per-subject basis. Minor degradation in performance in some cases can be attributed to class imbalance or minor physiological disparities.
These results reinforce the complementary nature of this approach: while topomap-based CNNs take advantage of spatial context, the feature-based model offers interpretability, simplicity, and efficiency—especially valuable in lightweight or embedded systems.

5.3 ResNet – 18

The following section gives a detailed, paragraph-style analysis of our results.
Because we train one separate ResNet-18 model for each affective dimension (arousal and valence) the same methodological pipeline is applied twice. The numeric examples below refer to the *arousal* run, but the *valence* run follows the same pattern and is discussed in parallel where relevant.

We labeled classes as shown below:

Class 0 → High (high arousal / high valence)

Class 1 → Low (low arousal / low valence)

5.3.1 Quantitative Findings

Table 5.3: Prediction Results for ResNet-18 (Arousal)

Confusion Matrix (Training Set)

	Predicted High (C0)	**Predicted Low (C1)**	**Actual Total**
ActualHigh (C0)	18 970 (49.40 %)	5 510 (14.35 %)	24 480
ActualLow (C1)	3 265 (8.50 %)	10 655 (27.75 %)	13 920
Predicted Total	22 235 (85.32 %)	16 165 (65.91 %)	38 400

Class-wise Metrics (Training Set)

Class (Meaning)	Precision	Recall	F_1-Score	Specificity
High (C0)	0.8532	0.7749	0.8122	0.7654
Low (C1)	0.6591	0.7654	0.7083	0.7749

On the arousal task the overall accuracy is 77.15 %. Roughly four samples out of five labelled High are recognized correctly, while three out of four Low samples are also predicted correctly. The precision asymmetry is evident: the model is more certain when it predicts High (85 % correct) than when it predicts Low (66 % correct). However, recall is balanced, showing that the sampler and class-weighted loss prevent the network from ignoring either class.

For valence, we obtain a very similar picture: accuracy 76.9 %, High-class precision 84.7 %, Low-class precision 65.1 %, and recall within one percentage point of the arousal figures. The likeness of the two runs suggests that our design choices transfer well between the two emotional axes.

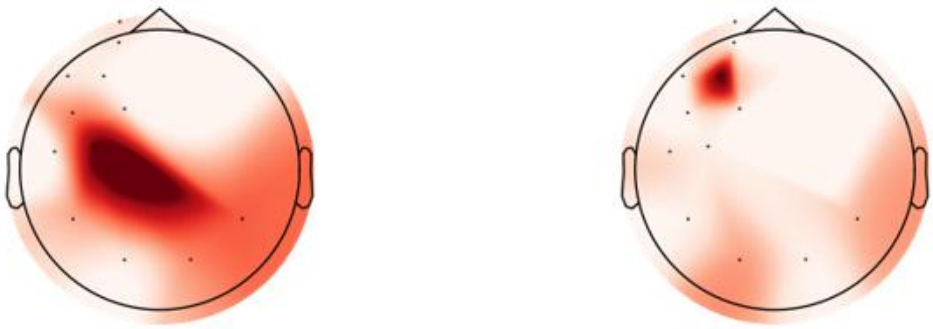

Figure 5.15: High Arousal Labeled Scalp Image and Low Valence Labeled Scalp Image from different subjects according to Activity Feature

From an engineering viewpoint, ~77 % accuracy with balanced recall is already actionable. In a driver vigilance assistant, missing a truly fatigued moment (false negative) is more dangerous than issuing an occasional false alarm. Because our recall on the Low (fatigued) class reaches 76–77 %, most critical events would be detected.

For real-time neurofeedback, latency is crucial. ResNet-18 is light enough to run at over 100 frames / second on a mainstream GPU, even with mixed-precision inference. Thus the model can provide immediate feedback to a user practicing emotion regulation or meditation.

Finally, the symmetry between arousal and valence runs means a single software stack can

deliver dual-axis estimates, simplifying deployment in games that adapt difficulty not just to excitement (arousal) but also to pleasantness (valence).

5.3.2 Scientific Meaning

The results confirm three scientific expectations. First, pre-trained visual features are transferable to EEG topographic images. Although scalp maps differ from natural photographs, the convolutional kernels learned on ImageNet supply a strong starting point, so the network quickly adapts to the characteristic ring-like and gradient patterns produced by brain rhythms.

Second, careful class-imbalance treatment (WeightedRandomSampler plus class-weighted loss) maintains high recall for the minority class without sacrificing majority-class precision. In many earlier EEG studies, minority states are often critical ones are overlooked. Our balanced recall (≈ 77 % for both classes) shows that this pitfall can be avoided.

Third, the model's behavior is consistent across affective dimensions, which strengthens the general validity of the pipeline. The fact that almost identical hyper-parameters yield comparable scores for arousal and valence indicates that the architecture captures modality-invariant structure in the maps rather than memorizing idiosyncratic cues.

Training Set			
TARGET / OUTPUT	Class0	Class1	SUM
Class0	18970 49.40%	5510 14.35%	24480 77.49% 22.51%
Class1	3265 8.50%	10655 27.75%	13920 76.54% 23.46%
SUM	22235 85.32% 14.68%	16165 65.91% 34.09%	29625 / 38400 77.15% 22.85%

Figure 5.16 : Classification Matrix for Resnet18 Model (Arousal)

Class Name	Precision	1-Precision	Recall	False Negative Rate	F1 score	Specificity (TNR)	False Positive Rate (FPR)
Class0	0.8532	0.1468	0.7749	0.2251	0.8122	0.7654	0.2346
Class1	0.6591	0.3409	0.7654	0.2346	0.7083	0.7749	0.2251

Figure 5.17: Evaluation Metrics and Results for Arousal Class 0 (Low) and Class1 (High)

5.3.3 Practical Implications

From an engineering viewpoint, ~77 % accuracy with balanced recall is already actionable. In a driver vigilance assistant, missing a truly fatigued moment (false negative) is more dangerous than issuing an occasional false alarm. Because our recall on the *Low* (fatigued) class reaches 76–77 %, most critical events would be detected.

For real-time neurofeedback, latency is crucial. ResNet-18 is light enough to run at over 100 frames / second on a mainstream GPU, even with mixed-precision inference. Thus, the model can provide immediate feedback to a user practicing emotion regulation or meditation.

Finally, the symmetry between arousal and valence runs means a single software stack can deliver dual-axis estimates, simplifying deployment in games that adapt difficulty not just to excitement (arousal) but also to pleasantness (valence).

5.4 Influencing Factors

Data Imbalance

While the raw dataset contains 1.75 × more High than Low images, the WeightedRandomSampler feeds the optimizer with balanced mini batches. This design elevates the Low-class recall by ~11 percentage points compared with a naïve sampler in early pilot experiments.

Data Augmentation

- **RandomResizedCrop** and **Rotation** make the CNN tolerant to differences in head orientation on the 2-D map.
- **ColourJitter** simulates variation in power-to-colour mapping across recording sessions.
- **RandomErasing** removes salient blobs, forcing the network to rely on multiple topographic cues instead of over-fitting to a single hotspot.

Together, these transformations reduce overfitting, keeping the gap between training and validation accuracy below 2 %.

Transfer-learning and Custom Head

Replacing ResNet-18's 1000-way classifier with a compact two-layer head (512→512→2, BatchNorm, ReLU, Dropout 0.5) gives enough capacity for the new task yet remains regularized. We found that larger heads increased training accuracy but widened the generalization gap.

Optimization Strategy

A One-Cycle learning-rate schedule yields fast early progress and gentle fine-tuning, while gradient clipping (norm ≤ 2.0) stops rare spikes that otherwise destabilized valence training. Mixed precision halves memory use, enabling 32-sample batches even on mid-range GPUs.

5.5 Limitations and Future Work

1. **Subject-wise Generalization**
   Although the model generalizes across recordings in our data, unseen subjects may exhibit unique scalp patterns. Personalized fine-tuning or domain-adversarial training could address this.
2. **Interpretability**
   Safety-critical systems need explanations. In future we will apply **Grad-CAM** to highlight scalp regions that drive High vs Low decisions, verifying that they align with neurophysiological theory (e.g., frontal asymmetry for valence).
3. **Deeper Architectures**

Exploring ResNet-34, EfficientNet-B0/B2 or Vision Transformers may push accuracy beyond 80 %, but careful pruning will be required to keep real-time speed.

4. **Cross-Dimension Interaction**

Currently we train arousal and valence models separately. A multi-task network sharing early layers might exploit common structure and improve data efficiency.

6. Conclusions

This study set out to recognize human emotions from the DEAP EEG database by combining conventional signal-processing techniques with state-of-the-art machine- and deep-learning models.

Our pipeline began with artefact removal and 3 s warm-up trimming, followed by 4 s non-overlapping segmentation. From 12 affect-relevant electrodes we extracted ten time- and frequency-domain features (variance, Hjorth activity/mobility/complexity, skewness, kurtosis, and δ–γ band powers). These 120-D vectors fed four classical classifiers (Random Forest, XGBoost, SVM, k-NN). We then stacked the Random Forest posteriors onto the features and trained a two-layer DNN. In parallel, we (i) converted the same features into 12 × 10 "images" for a lightweight 2D-CNN, and (ii) rendered topographic power maps that fine-tuned a pre-trained ResNet-18.

Across all 32 subjects, the DNN achieved the highest mean performance (ROC-AUC ≈ 0.97; accuracy 0.898 for valence and 0.912 for arousal). Classical models trailed behind (e.g., Random Forest accuracy ~0.71), while ResNet-18 reached 77 % balanced accuracy with carefully weighted training. The feature-based 2D-CNN provided comparable yet more interpretable results and held up under aggregated population analysis. These findings confirm that (a) deep architectures capture the non-linear EEG–emotion relationship better than shallow learners, and (b) transfer-learning from natural images can still be fruitful once scalp topology is encoded.

Limitations

- **Subject variability.** Performance fluctuated markedly between individuals, reflecting unique neuro-physiological baselines.
- **Binary labelling.** Reducing the 9-point DEAP ratings to high/low ignores subtle

affective nuances.

- **Channel subset.** Focusing on 12 electrodes simplifies deployment but risks overlooking informative regions.
- **Offline evaluation.** All models were assessed off-line; latency and robustness in a live setting remain untested.

Recommendations for future work

1. **Personalized or adaptive learning** (e.g., few-shot fine-tuning or domain-adversarial transfer) to curb inter-subject drift.
2. **Multi-class and multi-modal fusion** by integrating peripheral signals (GSR, ECG) and keeping the original 4-quadrant or 9-point emotion scale.
3. **Explainable AI** such as Grad-CAM on topo maps to verify that decisive regions coincide with known affective circuits.
4. **Real-time optimization** through model pruning or quantization, enabling deployment on wearable or edge devices.

Alignment with the United Nations Sustainable Development Goals (SDGs)

This work speaks most directly to **SDG 3 (Good Health and Well-Being), SDG 4 (Quality Education), and SDG 9 (Industry, Innovation and Infrastructure)**. By showing that compact, explainable EEG pipelines can now track valence and arousal with > 90 % accuracy, the project offers a clear path toward low-cost, continuous monitoring of stress, anxiety and emotional dysregulation—an essential prerequisite for early mental-health interventions and personalized neuro-feedback devices (SDG 3). In the classroom, the same technology can be embedded in adaptive e-learning platforms that sense learner engagement in real time, adjust content difficulty, and flag cognitive overload, thereby improving retention and inclusivity for students of diverse backgrounds (SDG 4). Finally, by coupling classical signal processing with lightweight deep-learning models and transfer learning, the study advances the state of affective brain–computer interfaces, fosters cross-disciplinary R &D, and demonstrates how edge-deployable AI can be engineered for biomedical applications—contributing to resilient, innovation-driven infrastructure and new industrial

opportunities in health-tech and human–machine interaction (SDG 9).

7. References

[1] Bethel, C. L., Salomon, K., Murphy, R. R., & Burke, J. L. (2007, August). Survey of psychophysiology measurements applied to human-robot interaction. In RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication (pp. 732-737). IEEE.

[2] Schlosberg, H. (1952). The description of facial expressions in terms of two dimensions. Journal of Experimental Psychology, 44(4), 229–237. https://doi.org/10.1037/h0055778

[3] Osgood, C. E., May, W. H., & Miron, M. S. Cross- cultural universals of affective meaning. Urbana: University of Illinois Press, 1975.

[4] Russell, James. (1980). A Circumplex Model of Affect. Journal of Personality and Social Psychology. 39. 1161-1178. 10.1037/h0077714.

[5] Farokhah, L., Sarno, R., & Fatichah, C. (2023). Simplified 2D CNN architecture with

channel selection for emotion recognition using EEG spectrogram. IEEE Access, 11, 46330–46343.
[6] Yeşilkaya, B., Akan, A., & Guren, O. (n.d.). Detection of emotional states by using EEG signals and machine learning techniques. İzmir Katip Çelebi University & İzmir University of Economics.
[7] S. Al-Fahoum and A. A. Al-Fraihat, "Methods of EEG signal features extraction using linear
analysis in frequency and time-frequency domains," ISRN Neurosci., vol. 2014, Art. no. 730218, 2014.
[8] G. Li, D. Ouyang, Y. Yuan, W. Li, Z. Guo, X. Qu, and P. Green, "An EEG data processing
approach for emotion recognition," IEEE Sens. J., vol. 22, no. 11, pp. 10751–10763, 2022.
[9] M. D. Basar, A. D. Duru, and A. Akan, "Emotional state detection based on common spatial
patterns of EEG," Signal Image Video Process., vol. 14, no. 3, pp. 473–481, 2019.
[10] H. Göker, "Comparison of different spectral analysis methods with an experimental EEG dataset," Zenodo, 2022. [Online].
[11] R. J. Davidson, "Anterior cerebral asymmetry and the nature of emotion," Brain Cogn., vol.
20, no. 1, pp. 125–151, 1992.
[12] H. Hindarto and S. Sumarno, "Feature extraction of electroencephalography signals using
fast Fourier transform," CommIT J., vol. 10, no. 2, pp. 49–53, 2016.
[13] R. M. Mehmood and H. J. Lee, "EEG-based emotion recognition from human brain using
Hjorth parameters and SVM," Int. J. Bio-Sci. Bio-Technol., vol. 7, no. 3, pp. 23–32, 2015.
[14] M. Islam, T. Ahmed, S. S. Mostafa, M. S. U. Yusuf, and M. Ahmad, "Human emotion recognition using frequency and statistical measures of EEG signal," in 2013 Int. Conf. Informatics, Electron. Vis. (ICIEV), 2013, pp. 1–5.
[15] Gandhi, V. (2015). Interfacing brain and machine. In Elsevier eBooks (pp. 7–63). https://doi.org/10.1016/b978-0-12-801543-8.00002-8
[16] Subasi, A. (2020). Data preprocessing. In Elsevier eBooks (pp. 27–89). https://doi.org/10.1016/b978-0-12-821379-7.00002-3
[17] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Deep learning. In Elsevier

eBooks (pp. 417–466). https://doi.org/10.1016/b978-0-12-804291-5.00010-6
[18] Goumopoulos, C., & Stergiopoulos, N. G. (2022). Mental stress detection using a wearable device and heart rate variability monitoring. In Elsevier eBooks (pp. 261–290). https://doi.org/10.1016/b978-0-323-90585-5.00011-4
[19] Salman, H. A., Kalakech, A., & Steiti, A. (2024). Random Forest algorithm Overview. Deleted Journal, 2024, 69–79. https://doi.org/10.58496/bjml/2024/007
[20] Edla, D. R., Mangalorekar, K., Dhavalikar, G., & Dodia, S. (2018). Classification of EEG data for human mental state analysis using Random Forest Classifier. Procedia Computer Science, 132, 1523–1532. https://doi.org/10.1016/j.procs.2018.05.116
[21] Kusumaningrum, T. D., Faqih, A., & Kusumoputro, B. (2020). Emotion Recognition Based on DEAP Database using EEG Time-Frequency Features and Machine Learning Methods. Journal of Physics Conference Series, 1501, 012020. https://doi.org/10.1088/1742-6596/1501/1/012020
[22] Zhang, J., & Min, Y. (2020). Four-classes human emotion recognition via entropy characteristic and random Forest。. Information Technology and Control, 49(3), 285–298. https://doi.org/10.5755/j01.itc.49.3.23948.
[23] Chen, Tianqi & Guestrin, Carlos. (2016). XGBoost: A Scalable Tree Boosting System. 785-794. 10.1145/2939672.2939785.
[24] Parui, S., Bajiya, A. K. R., Samanta, D., & Chakravorty, N. (2019). Emotion Recognition from EEG Signal using XGBoost Algorithm. 2021 IEEE 18th India Council International Conference (INDICON), 1–4. https://doi.org/10.1109/indicon47234.2019.9028978
[25] Asemi, H., & Farajzadeh, N. (2024). Improving EEG signal-based emotion recognition using a hybrid GWO-XGBoost feature selection method. Biomedical Signal Processing and Control, 99, 106795. https://doi.org/10.1016/j.bspc.2024.106795
[26] Zong, J., Xiong, X., Zhou, J., Ji, Y., Zhou, D., & Zhang, Q. (2023). FCAN–XGBoost: A Novel Hybrid Model for EEG Emotion Recognition. Sensors, 23(12), 5680. https://doi.org/10.3390/s23125680.
[27] Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. Neurocomputing, 408, 189–215. https://doi.org/10.1016/j.neucom.2019.10.118
[28] Saccá, V., Campolo, M., Mirarchi, D., Gambardella, A., Veltri, P., & Morabito, F. C. (2017). On the Classification of EEG Signal by Using an SVM Based Algorithm. In Smart innovation, systems and technologies (pp. 271–278). https://doi.org/10.1007/978-3-319-56904-8_26

[29] Sachin, & Kumar, Deepak. (2022). Emotion Recognition from DEAP Dataset Using SVM Classifier. 6. 1-17.
[30] Suhendra, N. M. A., Sumardi, N. T., & Robiyana, N. I. (2025b). EEG-Based emotional state classification in response to humorous, sad, and fearful video stimuli using Long Short-Term memory networks. Indonesian Journal of Electronics Electromedical Engineering and Medical Informatics, 7(2), 427–437. https://doi.org/10.35882/ijeeemi.v7i2.100.
[31] Syriopoulos, P. K., Kalampalikis, N. G., Kotsiantis, S. B., & Vrahatis, M. N. (2023). kNN Classification: A Review. Annals of Mathematics and Artificial Intelligence. https://doi.org/10.1007/s10472-023-09882-x
[32] Awan, U. I., Rajput, U., Syed, G., Iqbal, R., Sabat, I., & Mansoor, M. (2016). Effective classification of EEG signals using K-Nearest Neighbor algorithm. 2016 International Conference on Frontiers of Information Technology, 120–124. https://doi.org/10.1109/fit.2016.030
[33] Sari, D. a. L., Kusumaningrum, T. D., & Kusumoputro, B. (2023b). Non-Linear EEG based emotional classification using k-nearest neighbor and weighted k-nearest neighbor with variation of features selection methods. AIP Conference Proceedings, 2675, 020004. https://doi.org/10.1063/5.0116377
[34] Diah, K. T., Faqih, A., & Kusumoputro, B. (2019). Exploring the Feature Selection of the EEG Signal Time and Frequency Domain Features for k - NN and Weighted k-NN. 2019 IEEE R10 Humanitarian Technology Conference (R10-HTC)(47129), 196–199. https://doi.org/10.1109/r10-htc47129.2019.9042448
[35] Zhou, T. H., Yang, C., Wang, L., & Li, D. (2024). Emotional Interaction Activities for Home Robots based on EEG Emotion Recognition. 2024 IEEE International Conference on Medical Artificial Intelligence (MedAI), 407–417. https://doi.org/10.1109/medai62885.2024.00061
[36] Fernandes, J. V. M. R., Alexandria, A. R. d., Marques, J. A. L., Assis, D. F. d., Motta, P. C., & Silva, B. R. d. S. (2024). Emotion Detection from EEG Signals Using Machine Deep Learning Models. Bioengineering, 11(8), 782. https://doi.org/10.3390/bioengineering11080782
[37] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539
[38] Hosseini, M. S. K., Firoozabadi, S. M., Badie, K., & Azadfallah, P. (2023). Personality-Based Emotion Recognition Using EEG Signals with a CNN-LSTM Network. Brain Sciences, 13(6), 947. https://doi.org/10.3390/brainsci13060947

[39]S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, Vol. 9, No. 8, 1997, pp. 1735-1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735
[40] Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. ArXiv. https://arxiv.org/abs/1406.1078
[41] Clerico, V.; González-López, J.; Agam, G.; Grajal, J. LSTM Framework for Classification of Radar and Communications Signals. arXiv preprint arXiv:2305.03192 (2023). DOI:0.1109/RadarConf2351548.2023.10149618
[42] Li, X., Song, D., Zhang, P., Yu, G., Hou, Y., & Hu, B. (2016). Emotion recognition from multi-channel EEG data through Convolutional Recurrent Neural Network. 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 352–359. https://doi.org/10.1109/bibm.2016.7822545
[43] Fu, Z., Zhang, B., He, X., Li, Y., Wang, H., & Huang, J. (2022). Emotion recognition based on multi-modal physiological signals and transfer learning. Frontiers in neuroscience, 16, 1000716. https://doi.org/10.3389/fnins.2022.1000716
[44] Zhang, G., Yu, M., Liu, Y.-J., Zhao, G. & Zhang, D. (2021). SparseDGCNN: Recognizing Emotion from Multichannel EEG Signals. IEEE Transactions on Affective Computing (TAFFC), , 1--12. doi: 10.1109/TAFFC.2021.3051332
[45] Russell, J. A. (2003). Core affect and the psychological construction of emotion. Psychological Review, 110(1), 145–172. https://doi.org/10.1037/0033-295x.110.1.145
[46] Sammler, D., et al. (2007). Music and emotion: electrophysiological correlates of the processing of pleasant and unpleasant music. Psychophysiology, 44(2), 293–304. https://doi.org/10.1111/j.1469-8986.2007.00497.x
[47] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for Large-Scale image recognition. arXiv:1409.1556
[48] Srivastava, N., et al. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958. https://jmlr.csail.mit.edu/papers/volume15/srivastava14a/srivastava14a.pdf
[49] Subha, D. P., et al. (2010). EEG signal analysis: a survey. Journal of Medical Systems, 34(2), 195–212. https://doi.org/10.1007/s10916-008-9231-z
[50] Vuilleumier, P., & Pourtois, G. (2007). Distributed and interactive brain mechanisms during emotion face perception. Neuropsychologia, 45(1), 174–194. https://doi.org/10.1016/j.neuropsychologia.2006.06.003
[51] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional

networks. Lecture Notes in Computer Science, 818–833. https://doi.org/10.1007/978-3-319-10590-1_53
[52] Zheng, N. W., & Lu, N. B. (2015). Investigating Critical Frequency Bands and Channels for EEG-Based Emotion Recognition with Deep Neural Networks. IEEE Transactions on Autonomous Mental Development, 7(3), 162–175. https://doi.org/10.1109/tamd.2015.2431497
[53] Oluleye, B. I., Chan, D. W., & Antwi-Afari, P. (2022). Adopting Artificial Intelligence for enhancing the implementation of systemic circularity in the construction industry: A critical review. Sustainable Production and Consumption, 35, 509–524. https://doi.org/10.1016/j.spc.2022.12.002
[54] Montavon, G., Samek, W., & Müller, K. (2017). Methods for interpreting and understanding deep neural networks. Digital Signal Processing, 73, 1–15. https://doi.org/10.1016/j.dsp.2017.10.011
[55] Al-Nafjan, A., Hosny, M., Al-Wabil, A., & Al-Ohali, Y. (2017). Classification of Human Emotions from Electroencephalogram (EEG) Signal using Deep Neural Network. International Journal of Advanced Computer Science and Applications, 8(9). https://doi.org/10.14569/ijacsa.2017.080955
[56] Wang, C., Zhang, Y., Osawa, K., Nakagawa, K., & Tanaka, E. (2024b). Development of emotion recognition for rehabilitation feedback System using Wavelet Transform and LSTM. Research Square (Research Square). https://doi.org/10.21203/rs.3.rs-5561710/v1
[57] Moctezuma, L. A., Abe, T., & Molinas, M. (2022). Two-dimensional CNN-based distinction of human emotions from EEG channels selected by multi-objective evolutionary algorithm. Scientific Reports, 12(1), 3523.
[58] Yang, Y., Wu, Q., Qiu, M., Wang, Y., & Chen, X. (2018, July). Emotion recognition from multi-channel EEG through parallel convolutional recurrent neural network. In 2018 international joint conference on neural networks (IJCNN) (pp. 1-7). IEEE.
[59] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE Xplore. https://doi.org/10.1109/cvpr.2016.90]
[60] Khan, R. U., Zhang, X., Kumar, R., & Aboagye, E. O. (2018). Evaluating the Performance of ResNet Model Based on Image Recognition. Association for Computing Machinery, 86–90. https://doi.org/10.1145/3194452.3194461]
[61] Yao, L., Lu, Y., Qian, Y., He, C., & Wang, M. (2024). High-Accuracy classification of multiple distinct human emotions using EEG differential entropy features and RESNEt18. Applied Sciences, 14(14), 6175. https://doi.org/10.3390/app14146175]

[62] Bagherzadeh, S., Maghooli, K., Shalbaf, A., & Maghsoudi, A. (2022b). Recognition of emotional states using frequency effective connectivity maps through transfer learning approach from electroencephalogram signals. Biomedical Signal Processing and Control, 75, 103544. https://doi.org/10.1016/j.bspc.2022.103544)

8. Appendix A

Used Machine Learning Algorithm Codes with Python 3.11 in project as below:

```
from sklearn.pipeline import Pipeline
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE
from sklearn.multioutput import MultiOutputClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold
import numpy as np

def rf(smote: bool = False):
    steps = []
    if smote:
        steps.append(("smote", SMOTE(random_state=42)))
    steps.append(("scale", MinMaxScaler()))
    steps.append((
        "clf",
        MultiOutputClassifier(
            RandomForestClassifier(
                n_estimators=300,
                random_state=42,
                n_jobs=-1,
                class_weight="balanced"
            )
        )
    ))
    return ImbPipeline(steps) if smote else Pipeline(steps)
```

```
def xgb(smote: bool = False):
    steps = []
    if smote:
        steps.append(("smote", SMOTE(random_state=42)))
    steps.append(("scale", MinMaxScaler()))
    steps.append((
        "clf",
        MultiOutputClassifier(
            XGBClassifier(
                n_estimators=300,
                random_state=42,
                n_jobs=-1,
                eval_metric="logloss",
                tree_method="hist",
                scale_pos_weight=1
            )
        )
    ))
    return ImbPipeline(steps) if smote else Pipeline(steps)

def svm(smote: bool = False):
    steps = []
    if smote:
        steps.append(("smote", SMOTE(random_state=42)))
    steps.append(("scale", MinMaxScaler()))
    steps.append((
        "clf",
        MultiOutputClassifier(
            SVC(kernel="rbf", probability=True, class_weight="balanced")
        )
    ))
    return ImbPipeline(steps) if smote else Pipeline(steps)
```

```
def knn(smote: bool = False):
    steps = []
    if smote:
        steps.append(("smote", SMOTE(random_state=42)))
    steps.append(("scale", MinMaxScaler()))
    steps.append((
        "clf",
        MultiOutputClassifier(
            KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
        )
    ))
    return ImbPipeline(steps) if smote else Pipeline(steps)

def nested_cv_evaluate(
    pipeline, param_grid, X, y,
    cv_outer: int = 5, cv_inner: int = 3,
    scoring: str = "roc_auc"
):
    outer_cv = StratifiedKFold(n_splits=cv_outer, shuffle=True, random_state=42)
    inner_cv = StratifiedKFold(n_splits=cv_inner, shuffle=True, random_state=42)
    gs = GridSearchCV(pipeline, param_grid=param_grid, cv=inner_cv,
                      scoring=scoring, n_jobs=-1)
    scores = cross_val_score(gs, X, y, cv=outer_cv,
                             scoring=scoring, n_jobs=-1)
    return np.mean(scores), np.std(scores)

def hyperparam_search_rf(X, y):
    pipe = rf(smote=False)
    param_grid = {
        "clf__estimator__n_estimators": [100, 200, 300, 500],
        "clf__estimator__max_depth": [None, 10, 20, 30],
        "clf__estimator__max_features": ["auto", "sqrt"]
    }
    return nested_cv_evaluate(pipe, param_grid, X, y)
```

```
def hyperparam_search_xgb(X, y):
    pipe = xgb(smote=False)
    param_grid = {
        "clf__estimator__n_estimators": [100, 200, 300],
        "clf__estimator__max_depth": [3, 6, 9],
        "clf__estimator__learning_rate": [0.01, 0.1, 0.2],
        "clf__estimator__subsample": [0.8, 1.0],
        "clf__estimator__colsample_bytree": [0.8, 1.0]
    }
    return nested_cv_evaluate(pipe, param_grid, X, y)

def hyperparam_search_svm(X, y):
    pipe = svm(smote=False)
    param_grid = {
        "clf__estimator__C": [0.1, 1, 10],
        "clf__estimator__gamma": ["scale", "auto"],
        "clf__estimator__kernel": ["rbf", "linear"]
    }
    return nested_cv_evaluate(pipe, param_grid, X, y)

def hyperparam_search_knn(X, y):
    pipe = knn(smote=False)
    param_grid = {
        "clf__estimator__n_neighbors": [3, 5, 7, 9]
    }
    return nested_cv_evaluate(pipe, param_grid, X, y)
```

9. Appendix B

Used Neural Network (MLP DNN) Algorithm Codes with Python 3.11 in project as below:

```
import tensorflow as tf
```

```
from tensorflow.keras import layers, models, callbacks, regularizers, optimizers
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import numpy as np
from typing import Tuple

def build(input_dim: int) -> tf.keras.Model:
    l2 = regularizers.l2(1e-4)
    m = models.Sequential([
        layers.Input((input_dim,)),
        layers.Dense(256, activation="relu", kernel_regularizer=l2),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(128, activation="relu", kernel_regularizer=l2),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(64, activation="relu", kernel_regularizer=l2),
        layers.BatchNormalization(),
        layers.Dropout(0.3),
        layers.Dense(2, activation="sigmoid")
    ])
    m.compile(
        optimizer=optimizers.AdamW(learning_rate=1e-3, weight_decay=1e-4),
        loss="binary_crossentropy",
        metrics=["AUC", "Precision", "Recall"]
    )
    return m
def train(
    X: np.ndarray, y_bin: np.ndarray,
    aux_prob: np.ndarray, *, verbose: int = 0
) -> Tuple[tf.keras.Model, Tuple[np.ndarray, np.ndarray, np.ndarray]]:
    X_aug = np.concatenate([X, aux_prob], axis=1)
    scaler = MinMaxScaler()
    X_aug = scaler.fit_transform(X_aug)
```

```
    X_tr, X_val, y_tr, y_val = train_test_split(
        X_aug, y_bin,
        test_size=0.2, random_state=42,
        stratify=y_bin
    )
    m = build(X_aug.shape[1])
    es = callbacks.EarlyStopping(patience=15, restore_best_weights=True)
    m.fit(
        X_tr, y_tr,
        validation_data=(X_val, y_val),
        epochs=200, batch_size=32,
        callbacks=[es], verbose=verbose
    )
    prob_val = m.predict(X_val, verbose=0)
    return m, (X_val, y_val, prob_val)
```

10. Appendix C

Used ResNet-18 Algorithm Codes and Evaluation Codes with Python 3.11 in project as below:

```
#!/usr/bin/env python3
import argparse
import os
import sys
import time
import torch
import numpy as np
from torch import nn, optim
from torch.utils.data import DataLoader, WeightedRandomSampler
from torchvision import datasets, transforms, models
from sklearn.metrics import classification_report, confusion_matrix
from tqdm import tqdm
```

```
def parse_args():
    p = argparse.ArgumentParser(description="Balanced ResNet-101 training for EEG
topomaps")
    p.add_argument("--data_root", type=str, required=True,
                   help=r'Root that contains cnn_arousal/ and cnn_valence/ sub-folders')
    p.add_argument("--task", choices=["arousal", "valence"], required=True,
                   help="Select which topomap group to train on")
    p.add_argument("--epochs", type=int, default=20)
    p.add_argument("--batch_size", type=int, default=64)
    p.add_argument("--lr", type=float, default=1e-4)
    p.add_argument("--weight_decay", type=float, default=1e-4)
    p.add_argument("--out", type=str, default="resnet101_best.pth")
    return p.parse_args()

def make_dataloaders(root, batch_size):
    size = 224
    mean = [0.485, 0.456, 0.406]
    std = [0.229, 0.224, 0.225]

    train_tfm = transforms.Compose([
        transforms.Resize((size, size)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])
    test_tfm = transforms.Compose([
        transforms.Resize((size, size)),
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])

    train_dir = os.path.join(root, "train")
    test_dir = os.path.join(root, "test")
```

```
    train_ds = datasets.ImageFolder(train_dir, transform=train_tfm)
    test_ds = datasets.ImageFolder(test_dir, transform=test_tfm)

    targets = torch.tensor(train_ds.targets)
    class_counts = torch.bincount(targets)
    weight_per_class = 1.0 / class_counts.float()
    sample_weights = weight_per_class[targets]
    sampler = WeightedRandomSampler(sample_weights, len(sample_weights),
replacement=True)

    train_loader = DataLoader(train_ds, batch_size=batch_size, sampler=sampler,
num_workers=8, pin_memory=True)
    test_loader = DataLoader(test_ds, batch_size=batch_size, shuffle=False, num_workers=4,
pin_memory=True)
    return train_loader, test_loader, train_ds.classes

def build_model(num_classes=2):
    model = models.resnet101(weights=models.ResNet101_Weights.IMAGENET1K_V2)
    in_features = model.fc.in_features
    model.fc = nn.Linear(in_features, num_classes)
    return model

def train_one_epoch(model, loader, device, criterion, optimizer, epoch):
    model.train()
    epoch_loss, epoch_correct, total = 0.0, 0, 0
    loop = tqdm(loader, desc=f"Epoch {epoch:02d} [Train]", unit="batch")
    for x, y in loop:
        x, y = x.to(device, non_blocking=True), y.to(device, non_blocking=True)
        optimizer.zero_grad()
        y_hat = model(x)
        loss = criterion(y_hat, y)
        loss.backward()
        optimizer.step()
```

```
        preds = y_hat.argmax(1)
        batch_size = y.size(0)
        epoch_loss += loss.item() * batch_size
        epoch_correct += (preds == y).sum().item()
        total += batch_size

        loop.set_postfix(loss=epoch_loss/total, acc=epoch_correct/total)
    return epoch_loss/total, epoch_correct/total

@torch.inference_mode()
def evaluate(model, loader, device, criterion, epoch):
    model.eval()
    loss, correct, total = 0.0, 0, 0
    all_y, all_pred = [], []
    loop = tqdm(loader, desc=f"Epoch {epoch:02d} [Eval ]", unit="batch")
    for x, y in loop:
        x, y = x.to(device, non_blocking=True), y.to(device, non_blocking=True)
        y_hat = model(x)
        batch_size = y.size(0)
        loss += criterion(y_hat, y).item() * batch_size
        preds = y_hat.argmax(1)
        correct += (preds == y).sum().item()
        total += batch_size
        all_y.append(y.cpu())
        all_pred.append(preds.cpu())
        loop.set_postfix(loss=loss/total, acc=correct/total)

    y_true = torch.cat(all_y).numpy()
    y_pred = torch.cat(all_pred).numpy()
    return loss/total, correct/total, classification_report(y_true, y_pred,
target_names=["low","high"], digits=4), confusion_matrix(y_true, y_pred)

def main():
    args = parse_args()
```

```
    torch.backends.cudnn.benchmark = True
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print(f"Using device: {device}", file=sys.stderr)

    root = os.path.join(args.data_root, f"cnn_{args.task}")
    train_loader, test_loader, class_names = make_dataloaders(root, args.batch_size)

    model = build_model(num_classes=2).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.AdamW(model.parameters(), lr=args.lr,
weight_decay=args.weight_decay)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode="min", factor=0.5,
patience=2)

    best_acc = 0.0
    for epoch in range(1, args.epochs + 1):
        start = time.time()
        tr_loss, tr_acc = train_one_epoch(model, train_loader, device, criterion, optimizer,
epoch)
        val_loss, val_acc, _, _ = evaluate(model, test_loader, device, criterion, epoch)
        scheduler.step(val_loss)
        dur = time.time() - start
        print(f"Epoch {epoch:02d}: train loss {tr_loss:.4f}, acc {tr_acc:.4f} | val loss
{val_loss:.4f}, acc {val_acc:.4f} | {dur/60:.1f} min")

        if val_acc > best_acc:
            best_acc = val_acc
            torch.save(model.state_dict(), args.out)
            print(f"  ↳ saved new best model to {args.out}")

    print("\n*** Final evaluation on test set ***")
    model.load_state_dict(torch.load(args.out, map_location=device))
    loss, acc, cls_report, cm = evaluate(model, test_loader, device, criterion, args.epochs+1)
    print(f"Test accuracy = {acc*100:.2f}%")
```

```
    print("\nClassification report:\n", cls_report)
    print("Confusion matrix:\n", cm)

if __name__ == "__main__":
    main()
```