NeuroVibe: A Mobile EEG Platform for Real-Time Aesthetic Perception Analysis and Comparative Study of Mobile vs. Edge Computing

Valentina Bertei, Mirco Concu, Alex Sgammato, Francesco Tarchi
v.bertei@studenti.unipi.it, m.concu@studenti.unipi.it, a.sgammato1@studenti.unipi.it, f.tarchi3@studenti.unipi.it

ABSTRACT

This project presents the development of a mobile application in Kotlin, NeuroVibe, designed to interface with an EEG (electroencephalogram) reader via Wi-Fi, enabling the wireless collection of real-time brainwave data. The core objective is to capture EEG signals from users as they view a series of paintings and analyze the obtained data using an AI-based classification model. By combining mobile development, wireless data acquisition, and machine learning, the application creates an interactive platform for studying aesthetic perception through neurophysiological responses.
Rather than evaluating the classification accuracy of the AI models employed, this study focuses on comparing the energy efficiency of two deployment strategies: on-device (mobile) processing and edge-based inference. The findings demonstrate that delegating model inference to a remote edge server results in significantly lower CPU load and energy usage on the mobile device, with a slight increase in Wi-Fi activity. These outcomes suggest that edge computing offers a promising, energy-efficient solution for real-time EEG data analysis in mobile contexts.

1 Introduction

This project involves the development of a mobile application designed for use in different realistic settings, such as museum settings, aimed at exploring the intersection of art and human emotion through the lens of neuroaesthetics. Neuroaesthetics is a field of study that investigates the neural and physiological basis of aesthetic experiences, seeking to understand how the brain responds to art and beauty. The application connects to an EEG (electroencephalogram) device to collect real-time brain activity data from application users, as they view various artworks displayed on the app. By analyzing this physiological data, the system aims to infer the user's emotional and aesthetic response specifically, whether a piece of art is perceived as beautiful or not. The collected EEG signals are processed using an AI classification model capable of detecting subtle variations associated with emotional arousal and aesthetic appreciation. This approach provides a novel, data-driven way to understand how individuals react to visual art, offering insights into personal and collective experiences of beauty. The application not only enhances user engagement but also contributes to the broader study of neuroaesthetics by enabling in situ, real-time measurement of aesthetic response in naturalistic environments like museums.
However, continuously using a mobile phone to collect, process, and analyze EEG data throughout the entire museum visit can lead to significant power consumption, posing a challenge for current-generation mobile devices. To address this limitation, the project explores the use of edge computing as a solution for more energy-efficient processing. Edge computing allows data to be processed locally -either on the device itself or on a nearby low-power processing unit- reducing the need for constant communication with cloud servers or overloading the mobile device processor. This setup not only helps lower energy consumption but also minimizes latency. As part of the project, we implemented and compared two system configurations: one where all EEG data is processed directly on the smartphone, and another where computation is offloaded to an edge device. The comparison focuses on metrics such as energy usage and processing time. Through this analysis, the project aims to assess the feasibility of edge computing for real-time neuroaesthetic evaluation in mobile environments, contributing to more sustainable and scalable applications of physiological computing in public and cultural spaces.

2 Related Works

Our work is inspired by the study [1] of M. Palmieri, M. Avvenuti, F. Marcelloni and A. Vecchio, titled "***Recognizing Special Artpieces through EEG: A Journey in Neuroaesthetics Classification***". In this research, three approaches are proposed to classify neuroaesthetic responses to art contemplation by analyzing EEG signals. These methods, evaluated employing a public dataset collected using mobile tools from museum visitors at an art exhibit, address typical challenges in EEG analysis such as small dataset size and class imbalance through data segmentation and sampling techniques. In this work, authors highlight the potential of deep learning methods, particularly convolutional neural networks (CNNs), to improve generalizable model accuracy, although these models pose challenges in interpretability. They suggest the integration of explainable AI (XAI) techniques to enhance model transparency and usability by domain experts. Building upon these findings, our work extends the exploration of EEG feature extraction and classification in neuroaesthetic contexts, focusing on adapting the model developed by M. Palmieri *et al.* to support the evaluation of arbitrary artworks or visual stimuli. We further

employ this model to compare edge-based and mobile-based architectures, analyzing their classification performance and estimating energy consumption using two neural network configurations, one with 100 neurons and another with 1000 neurons. Moreover, our approach draws inspiration from the system proposed by Bateson and Asghar in "***Development and Evaluation of a Smartphone-Based Electroencephalography (EEG) System***" [2], where the authors designed a general-purpose EEG platform capable of acquiring real-time brain signals via Wi-Fi and integrating them directly into a mobile application. Their system demonstrated the feasibility of using a smartphone both as a recording and stimulus delivery device, validating its signal reliability in both resting and movement conditions. This study underlines the potential of mobile EEG for real-time, user-centered neuroscience applications, a concept closely aligned with our work. NeuroVibe builds upon this vision by integrating wireless EEG data collection with a Kotlin-based mobile interface and leveraging AI models to classify aesthetic responses to visual stimuli. While Bateson and Asghar focused primarily on validating signal quality and ERP detection, our work extends this paradigm toward a more applied neuroaesthetic scenario, exploring not only real-time processing but also the impact of different deployment architectures (edge vs mobile) on energy consumption and system responsiveness.
Crucially, our evaluation of the energy-performance trade-offs between local and remote model inference was strongly inspired by the findings of Hu *et al*. in ***"Quantifying the Impact of Edge Computing on Mobile Applications"*** [3], who demonstrated that offloading computation to an edge server -rather than to a distant cloud or executing locally-can significantly reduce response times and energy usage in mobile applications, especially those with interactive and latency-sensitive components. Their work provided a solid foundation for framing our comparison between mobile and edge deployments within a broader performance-aware context.

3 Architecture

The architecture of the project is composed of two main components: a **mobile part** and an **edge part**, designed to work together for efficient data collection and processing in real-time. The **mobile part** is responsible for interfacing with the EEG device via Wi-Fi to collect physiological signals from the user as they view artworks. The mobile application is developed in Kotlin and optimized for Android devices, ensuring compatibility and responsiveness during usage. The **edge part** is a lightweight computing unit placed within the museum environment, capable of receiving raw EEG data from the mobile device. By offloading computationally intensive tasks to the edge device, the system reduces the power consumption of the smartphone while maintaining fast response times and data privacy. This distributed architecture enables real-time analysis without relying on cloud infrastructure.

3.1 MindRove headset

MindRove arc is the EEG headset used in this project to capture brain activity data from users in real-time. The MindRove device records raw EEG signals directly from the scalp. It is equipped with 6 active channels and 2 reference channels, for a total of 8 electrodes. These raw signals can later be processed to extract features potentially related to cognitive and emotional states. The electrodes are positioned over the top of the head and are placed at the following locations according to the international 10–10 system: **C5, C3, C1, C2, C4, and C6**. These positions are commonly associated with the brain regions which are relevant in studies of aesthetic perception and affective responses. MindRove uses **semi-dry electrodes**, eliminating the need for conductive gels, which makes it more user-friendly, less invasive, and suitable for prolonged use in public settings. The headset is powered by a **rechargeable lithium-polymer (LiPo) battery**, offering sufficient autonomy for several (4-6) hours of data acquisition. Its wireless communication capabilities allow it to transmit EEG data in real-time to the mobile device or edge processor, forming the foundation for the system's neuroaesthetic evaluation pipeline.

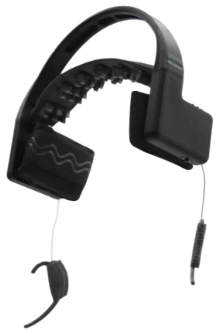

Figure 1: MindRove Headset ARC.

3.2 MindRove configuration electrodes

The machine learning model used for aesthetic classification was originally trained on EEG data collected using the full **10–20 international system**, which includes **8 electrode positions** distributed across the scalp. This setup allows for a more comprehensive spatial representation of brain activity, particularly useful for capturing neural responses associated with visual and emotional stimuli. However, the **MindRove EEG headset** used in our application includes only **6 active electrode channels** positioned over the central area of the scalp (C5, C3, C1, C2, C4, and C6), and lacks two of the positions expected by the model. To address this limitation and ensure compatibility, we apply an **electrode mapping strategy**, which allows us to estimate or adapt the missing channels based on the available ones. This mapping ensures that the input feature vector maintains the same structure as the one used during model training. The adaptation strategy may include interpolation, averaging of nearby channels, or substitution based on symmetrical positioning. This adjustment allows us to leverage a pre-trained model with minimal loss in accuracy while using a more practical and portable EEG device in real-world settings. The **channel mapping strategy** reassigns available MindRove channels to approximate the required 10–20 system positions as follows:

- **C3 ← C3** (channel 2)
- **C4 ← C4** (channel 5)
- **F3 ← C5** (channel 1)
- **Fz ← C1** (channel 3)
- **F4 ← C6** (channel 6)
- **P3 ← C1** (channel 3)
- **Pz ← C2** (channel 4)
- **P4 ← C2** (channel 4)

This mapping allows the model to maintain its expected input structure by reusing or approximating certain signals, especially for the parietal (P3, Pz, P4) and frontal (F3, Fz, F4) regions. While this adaptation introduces a degree of approximation, it enables effective use of a lightweight, mobile EEG headset while retaining compatibility with the pre-trained model.

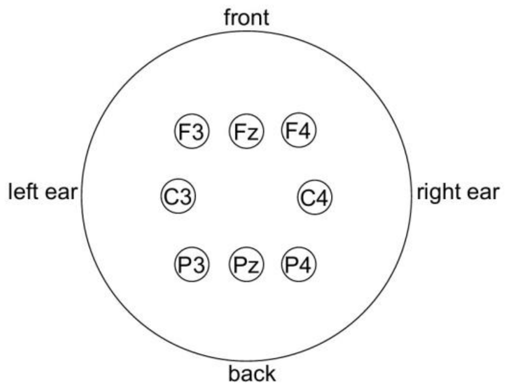

Figure 2: 10-20 system used in [1] for training the model.

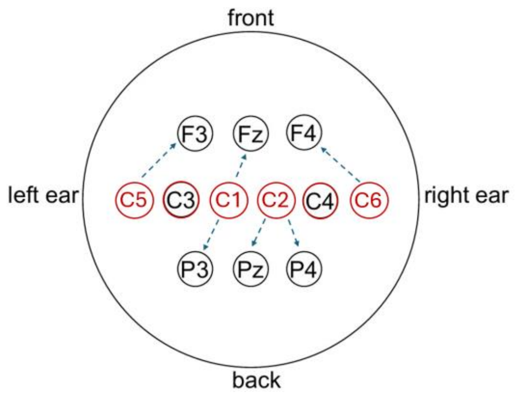

Figure 3: Mapping from a 10-10 (MindRove) electrode system to a 10-20 system. MindRove electrodes are highlighted in red.

3.3 Mobile architecture

The mobile architecture of this project is structured into three main stages: **data processing**, **feature extraction**, and **model prediction**. This modular design enables efficient handling and analysis of EEG data collected during the user app usage.

- **Data Processing Stage**
 This stage is responsible for acquiring raw EEG signals from the MindRove headset. The data is transmitted wirelessly to the system, where it undergoes initial preprocessing steps such as filtering, artifact removal, and normalization. These steps ensure that the input is clean and consistent for further analysis.
- **Feature Extraction Stage**
 Once the data is preprocessed, relevant features are extracted to capture meaningful patterns associated with the user's cognitive and emotional responses. These features form a compact representation of the brain activity linked to aesthetic perception.
- **Model Prediction Stage**
 In the final stage, the extracted features are fed into a machine learning model trained to classify whether the user perceives an image as beautiful or not. The model can run either on the mobile device or be offloaded to an edge computing unit, depending on the configuration, enabling flexibility in performance and energy efficiency.

3.4 Edge architecture

In the edge computing configuration, the system is designed to offload computational tasks from the mobile device to a lightweight, local server. This setup allows for energy-efficient processing while maintaining low latency and data privacy. The mobile application collects raw EEG data from the MindRove headset and formats it into a **CSV (Comma-Separated Values)** structure, containing time-stamped signal values from each electrode channel. The application then sends this CSV-formatted data to a **Python-based server** running locally on the edge device. Communication is established through an **HTTP POST request**, with the server built using the **Flask** web framework. The POST request contains the EEG data as a payload, which the server receives and processes in real-time. Upon receiving the data, the server passes the features to the trained classification model. The model evaluates the input and determines whether the EEG signals indicate a perception of the viewed image as **beautiful** or **not beautiful**. The server then returns the classification result as a response to the mobile application, which can use this feedback to update the user interface or store the outcome for further analysis. This edge-based architecture provides a scalable and efficient solution for real-time neuroaesthetic classification in a mobile environment.

4 User interface

The user interface (UI) of the Kotlin application was designed to provide a streamlined and intuitive workflow for EEG data collection and model evaluation using MindRove device. The interface ensures that the user can easily establish connectivity, collect EEG signals, and observe model inference results with minimal interaction and clear feedback.
The design of the single elements (most of them) extends the Material 3 styles, giving the app a look in accordance with the last Android design guidelines.

4.1 Connectivity with Mindrove

At launch, the application checks whether it is connected to the Mindrove EEG device via Wi-Fi. If a connection is not detected, the application prompts the user with a Wi-Fi setup dialog, guiding them to connect to the appropriate network. This step is essential, as all EEG-related functionality is dependent on a successful connection to Mindrove.

4.2 Start EEG Session

Once the device is connected, the user can press the "Start EEG" button. This action marks the beginning of the EEG data collection session. Pressing this button initializes the necessary communication protocols and prepares the system to receive and process EEG signals.

4.3 Painting-Based Stimulus and Data Collection

The application is designed to present a series of seven distinct paintings as visual stimuli. For each painting:

- The image is displayed on the screen.
- EEG data is collected individually per iteration, capturing brain activity in response to that specific painting.

4.4 Model Inference and Result Display

Following the data collection phase for each painting, the application presents four buttons, each representing a different combination of machine learning model and offloading strategy. These include variations such as local vs. remote inference and different model architectures.
Upon selecting a button, the application:

- sends the collected EEG data to the corresponding processing pipeline.
- executes the classification or inference task.
- displays the result on-screen, showing how the selected model/offloading configuration interpreted the EEG response to the specific painting.

This interaction is visually illustrated in Figure [4], where the sequence of painting presentation, data collection, and result visualization is shown, while in Figure [5] the button selection and the results are shown.

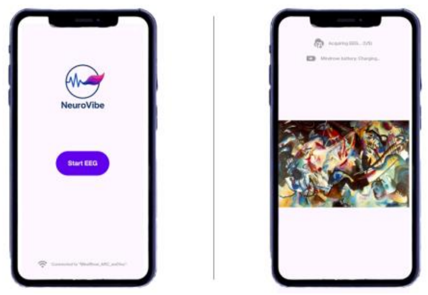

Figure 4: main menu on the left, painting view on the right.

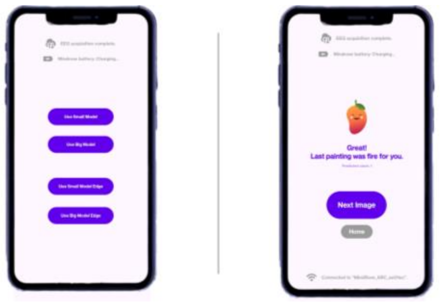

Figure 5: model choice menu on the left, analysis result on the right.

4.5 Night version of the app

The app supports also Android night mode. The entire app UI changes colors of its elements (background, app logo, buttons, texts, ...) with relation to the currently chosen Android system theme (light or dark).

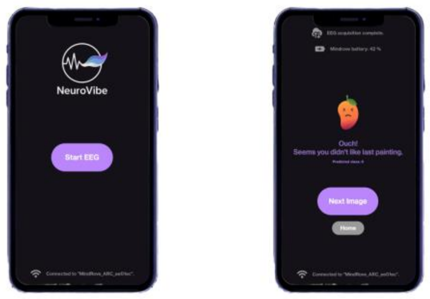

Figure 6: on the left the main menu, on the right a normal run (night mode).

5 Data Processing

The data collected from the MindRove EEG headset must undergo a structured processing workflow to be usable by the

classification model. Raw EEG signals are continuous, multi-channel time-series data that need to be segmented, organized, and formatted appropriately for analysis. To handle this, we use a custom class called `SensorData` (provided by MindRove library), which is designed to represent each EEG channel as well as the built-in motion sensors. This class is responsible for receiving and organizing the incoming data streams, associating each data point with a timestamp and channel label. The system captures EEG data while the user observes a painting for a total of 10 seconds. This duration is divided into five segments of 2 seconds each to enable fine-grained analysis and temporal resolution. Each 2-second segment is saved as a separate CSV file, containing raw EEG values for all channels over the duration of the recording. These CSV files serve as individual input units for further processing and classification. This segmentation approach ensures consistency in data size and timing, allowing the model to operate on standardized inputs. It also helps in capturing transient neural responses that may vary over the viewing period, providing more reliable and interpretable predictions from the AI model.

5.1 Data processing Pipeline

The **data processing pipeline** is a critical component that prepares raw EEG signals for input into the machine learning model. The pipeline ensures that the data is clean, consistent, and representative of meaningful brain activity related to aesthetic perception. It consists of the following sequential steps:

- **Data Retrieval**:
  The process begins by loading the EEG data from the CSV files generated during the recording session. Each file corresponds to a 2-second EEG segment and contains time-stamped values for each EEG channel. For each painting viewed by the user, five 2-second segment files are retrieved and processed.
- **Normalization**:
  To ensure consistency across different channels and recording sessions, the data is normalized. This step scales the signal values so that they share a common range or distribution, reducing variability caused by device calibration differences, user-specific signal amplitude, or session-based noise.
- **Filtering**:
  A fir **bandpass filter** is applied to retain only the EEG frequency components in the **1 to 50 Hz** range and order 50. This step removes irrelevant low-frequency drift and high-frequency noise (e.g. electrical interference), preserving the core brainwave frequencies (delta, theta, alpha, beta, and low gamma) associated with cognitive and emotional states.
- **Feature Extraction**:
  Following the filtering process, meaningful features are extracted from the EEG signal and used as input to the AI model, allowing it to determine whether the user perceives the artwork as aesthetically pleasing or not.

5.2 Feature Extraction

In the final stage of the data processing pipeline, we extract a set of descriptive features from each 2-second EEG segment to capture the essential characteristics of the brain signal. These features provide a compact and informative representation of the data for the classification model. The following features are computed for each EEG channel:

- **Shannon Entropy**: Measures the complexity and unpredictability of the signal, which can reflect the cognitive load and neural variability during aesthetic evaluation. It was computed as follows.

$$H = -\sum_{i=1}^{b} p_i \cdot \ln(p_i)$$

Where:
- b = bins number.
- p_i = estimated probability of bin i (defined as ${}^{n_i}/_{N}$).
- n_i = number of values falling into bin i.
- N = total number of values in the sequence.

- **Mean**: Represents the average amplitude of the EEG signal, providing a baseline measure of neural activity over the segment. It was computed as follows.

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- **Kurtosis**: Describes the "tailedness" of the signal distribution, indicating the presence of sharp peaks or outliers, which may be related to sudden neural responses. It was computed as follows.

$$\text{Kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s}\right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

Where:
- n = sample size (number of data points).
- $\bar{x}$ = sample mean (average of the sample data).

- **Minimum (Min)**: Captures the lowest signal value in the segment, useful for understanding the range of electrical activity.

$$\text{Min} = \min(x_1, x_2, \ldots, x_n)$$

- **Maximum (Max)**: Captures the highest signal value in the segment, complementing the minimum to define the amplitude span.

$$\text{Max} = \max(x_1, x_2, \ldots, x_n)$$

- **Standard Deviation**: Quantifies the amount of variation or dispersion in the EEG signal, offering insight into the signal's stability and fluctuation over time. It was computed as follows.

$$\text{Standard Deviation} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

The feature extraction process was carried out using the Kotlin Math library for mean, standard deviation, minimum, maximum, and kurtosis. Shannon entropy, not provided by the library, was computed using a custom implementation.
These features are computed independently for each EEG channel, and their aggregated values constitute a single row in a CSV file. This row is subsequently used as input to the machine learning model for classification.

6 Machine Learning Model

The classification component of our system is based on a **pre-trained neural network model** originally developed in the research paper [1]. This model was trained on EEG data collected with a standard 10-20 electrode layout to distinguish whether users perceive visual stimuli (artworks) as beautiful or not. To adapt this model to our application context -where we use the **MindRove headset** with only 6 available channels- we implemented a **channel mapping strategy** that reassigns our inputs to approximate the original 8-channel structure expected by the model. In our implementation, we evaluate **two variations** of pre-trained architecture:

- **Small Model**: A compact version with **100 neurons**. This model is suitable for real-time classification on mobile or low-power devices, where resource constraints are critical.
- **Big Model**: A more complex version with **1,000 neurons**. This configuration aims to maximize classification accuracy, leveraging higher computational power.

Since the model is pre-trained and we apply a mapped version of the EEG input, our goal is **not to assess classification performance**, but rather to **evaluate and compare the energy consumption** of the small and large models in different deployment scenarios.

7 Edge component

To support efficient offloading of computational tasks from the mobile device, we developed an **edge computing component** designed to perform real-time inference using pre-trained machine learning models. This component is implemented as a **Python script** that utilizes the **TensorFlow** library for neural network operations and **Flask** to handle communication via HTTP. The mobile application, after performing the initial data acquisition, preprocessing, and feature extraction, sends the resulting data as a **CSV file** to the edge server. This data represents a 10-second EEG recording divided into five 2-second segments, already transformed into the feature vector expected by the model.

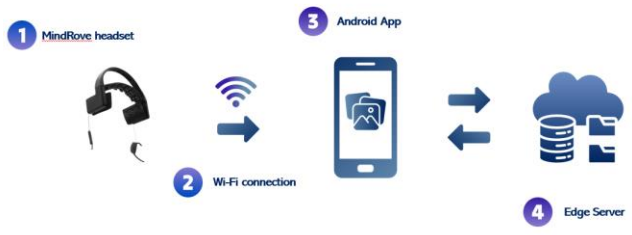

Figure 7: Edge architecture

The edge component functions as follows:

- A **Flask server** listens to **HTTP POST requests** from the mobile device.
- Upon receiving a CSV file, the server loads the data and passes it through the selected pre-trained model (either the small or large version).
- The model performs **inference**, classifying the user's response to the visual stimulus (e.g., whether the image was perceived as beautiful or not).
- The resulting **classification label** is then returned to the mobile application in HTTP response.

This architecture allows for a clear separation between lightweight mobile processing and heavier model inference, enabling energy-efficient deployment while preserving model accuracy and response time.

8 Evaluation and Result

This section presents the outcomes of the EEG data collection process and evaluates the performance of different models and offloading strategies. Key aspects such latency, CPU and Wi-Fi utilization time and user experience were analyzed to assess the effectiveness and responsiveness of the system.

8.1 Evaluation tools: *adb batterystats*

To objectively evaluate the energy consumption and resource usage of the application, we utilized the Android Debug Bridge (ADB) tool to collect system-level statistics. This allowed us to measure CPU and Wi-Fi utilization with high precision, focusing specifically on our application behavior during each test iteration. Each model was tested in four separate trials on both mobile and edge devices, resulting in a total of sixteen experiments.

8.2 Measurement Procedure

To ensure consistent and isolated measurements, we followed a standardized procedure for each test.

1) **Battery Stats Reset**

Before starting any measurement, we executed:

```
adb shell dumpsys batterystats --reset
```

This cleared all existing battery statistics, ensuring that only the data from the upcoming session would be recorded.

2) **Per-Iteration Testing**

For each EEG data collection and model inference iteration:

- The application was run through its full cycle (painting display → EEG collection → inference → result display).
- Immediately after, we collected battery and system usage stats with command:

```
adb shell dumpsys batterystats [package_name]
```

After retrieving the data, we reset the statistics again to isolate the next iteration.

3) **Metrics Collected**

From the battery and network statistics, we extracted the following metrics, filtered specifically for our application process:

- *CPU Utilization Time*: total time the CPU was actively used by the app.
- *CPU Energy Consumption*: energy in mAh attributed to CPU usage during the iteration.
- *Total Energy Consumption*: energy in mAh from both CPU and Wi-Fi module.
- *Wi-Fi Energy Consumption*: estimated energy usage for Wi-Fi operations in the session.
- *Data Received Over Wi-Fi*: number of bytes received by the app via Wi-Fi during inference or communication with the server.

These metrics were used to evaluate the efficiency of different models and offloading strategies, particularly in comparing local vs. remote inference. Remote models generally exhibited higher Wi-Fi usage and energy consumption due to network transfers, while local models were more CPU-intensive but reduced data transmission. The experimental results, illustrated in Figures [8] through [12], highlight the trade-offs between local and edge inference strategies across multiple performance and energy dimensions. In Figure [9], we observe that the edge solution consistently uses less CPU time compared to the global average, indicating a reduced computational load on the client device. This offloading of processing tasks contributes directly to the energy savings shown in Figure [9], where the edge solution achieves up to 15% lower overall energy consumption than global average. As expected, Figure [10] shows that the edge solution incurs a modest increase in Wi-Fi data usage, with up to 6% more data received, a result of transferring EEG data to the remote server for inference. Correspondingly, Figure [11] reveals that this increased data transmission leads to higher power consumption by the Wi-Fi module, reaching up to 20% more than the global average. Nevertheless, when considering the total energy footprint (including both CPU and Wi-Fi components) Figure [12] demonstrates that the edge solution still outperforms the local strategy, achieving up to 19% lower energy consumption overall. All percentages presented are calculated based on the global average of all measurement iterations, ensuring that the comparisons reflect sustained performance differences rather than isolated anomalies.

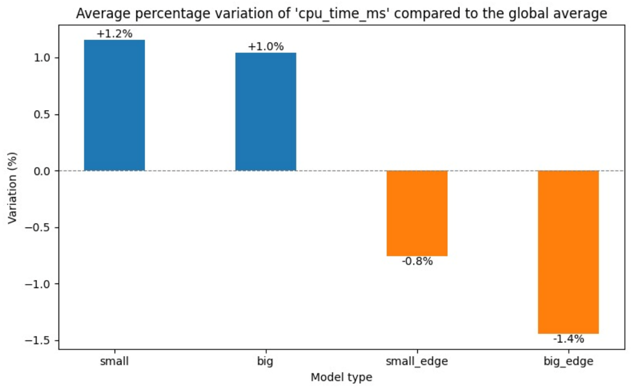

Figure 8: Average percentage variation of 'cpu_time_ms' compared to the global average

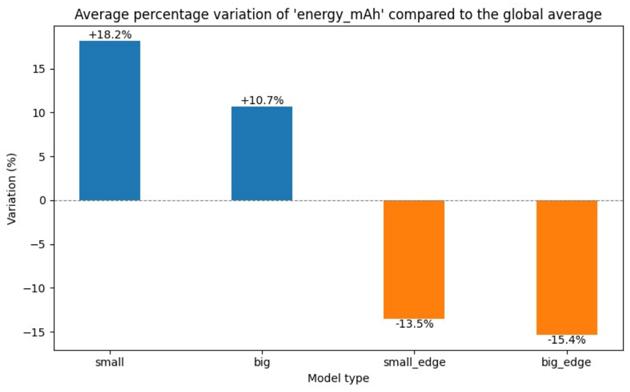

Figure 9: Average percentage variation of 'energy_mAh' compared to the global average

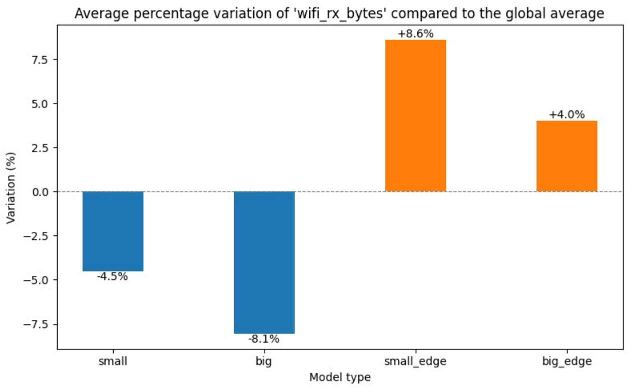

Figure 10: Average percentage variation of 'wifi_rx_bytes' compared to the global average

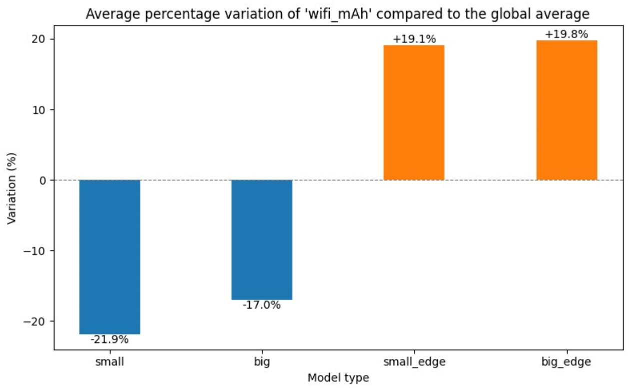

Figure 11: Average percentage variation of 'wifi_mAh' compared to the global average

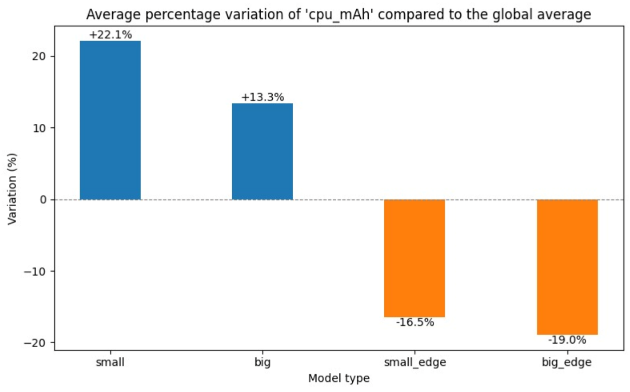

Figure 12: Average percentage variation of 'cpu_mAh' compared to the global average

9 Conclusion

In this work, we developed a mobile application that connects to a MindRove EEG headset to classify users' aesthetic perception of images using pre-trained neural network models. We implemented both mobile and edge computing architectures. While classification accuracy was not evaluated due to signal mapping and pre-trained models, we focused on comparing energy consumption between small and big models. This work demonstrated the feasibility and efficiency of using an edge-based solution for EEG data processing in a Kotlin-based mobile application. By offloading model inference to a remote server, the system achieved lower CPU usage and overall energy consumption compared to local processing, despite slightly increased Wi-Fi usage. These results highlight the edge approach as a practical and energy-efficient alternative for real-time EEG applications.

9.1 Future Directions

Several avenues for future improvement of the application have been identified to enhance its accuracy, usability, and overall functionality. First, adopting the same electrode configuration used during the original model training in [1] would make it possible to carry out a proper evaluation of prediction accuracy, which was not included in this work due to the mismatch between the electrode recording setup and the training conditions. This alignment would allow for a more reliable comparison between models and a better understanding of their performance in practical scenarios. Moreover, expanding the application scope beyond the current predefined gallery to allow users to upload and analyze any image would greatly increase its flexibility and appeal to a wider audience. Integrating the device camera directly into the application presents an additional opportunity to simplify the user workflow, enabling immediate image capture. Lastly, the addition of user account management features, coupled with the ability to view aggregated statistics from multiple users for each image, would not only enrich the user experience but also provide valuable data for deeper analysis of collective neuroaesthetic responses, fostering a more interactive and engaging platform.

REFERENCES

[1] Recognizing Special Artpieces through EEG: A Journey in Neuroaesthetics Classification (Maurizio Palmieri, Marco Avvenuti, Francesco Marcelloni and Alessio Vecchio) M. Palmieri, M. Avvenuti, F. Marcelloni and A. Vecchio, "Recognizing Special Art Pieces Through EEG: A Journey in Neuroaesthetics Classification," in IEEE Access, vol. 13, pp. 77696-77708, 2025, doi: 10.1109/ACCESS.2025.3562908.

[2] A. D. Bateson and A. U. R. Asghar, "Development and Evaluation of a Smartphone-Based Electroencephalography (EEG) System," in IEEE Access, vol. 9, pp. 75650-75667, 2021, doi: 10.1109/ACCESS.2021.3079992.

[3] Wenlu Hu, Ying Gao, Kiryong Ha, Junjue Wang, Brandon Amos, Zhuo Chen, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2016. Quantifying the Impact of Edge Computing on Mobile Applications. In Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '16). Association for Computing Machinery, New York, NY, USA, Article 5, 1–8. https://doi.org/10.1145/2967360.2967369.

Conference Location: Pisa, Italy ITA
Year: 2025
Date: May
Copyright Year: 2025
Copyright Statement: rights retained