Kushagra Kesarwani

Bachelor's thesis

Information and Communication Technology

2023

Exploring Machine Learning Models for Detection of Human Eye Blinks

– Brain Computer Interface and Machine Learning

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communication Technology

2023 | number of pages: 33

Kushagra Kesarwani

Exploring Machine Learning Models for Detection of Human Eye Blinks

- Brain Computer Interface and Machine Learning

Brain-computer interface (BCI) technology offers promise for enriching human-computer interactions across diverse domains. This thesis focuses on blink detection within BCI systems and their potential applications. The methodology encompasses data collection, EEG pre-processing, and model training. EEG data collected using the Ultracortex Mark IV headset was labeled for four categories. Various machine learning models were trained and evaluated, with the multi-layer CNN achieving a top performance of 74.70%. The results provide a foundation for advanced BCI projects, including assistive technology and hands-free systems, and suggest the need for hardware upgrades to expedite training. Future work entails data collection from multiple individuals, exploring real-world data collection settings, additional EEG nodes, and real-time applications.

In summary, this research advances BCI by addressing blink detection, enhancing human-computer interactions in healthcare, gaming, communication, and assistive technology.

Keywords:

BCI, CNN, EEG, Epoch, Machine Learning

Table of Contents

Figures

Equations

List of abbreviations

AI	Artificial Intelligence
BCI	Brain Computer Interface
CNN	Convolutional Neural Networks
EEG	Electroencephalogram
ML	Machine Learning
RNN	Recurrent Neural Networks

1 Introduction

Brain-Computer Interface (BCI) has rapidly emerged as a promising field with significant potential in numerous domains. BCI systems establish a direct communication pathway between the brain and external devices, allowing individuals to control and interact with technology using their brain signals. While BCI holds great promise, the accurate identification and interpretation of user commands remain essential for unlocking its full potential.

One crucial aspect that has garnered the attention of the author in BCI research is blink detection. Blinks are natural and involuntary eye movements that occur frequently in daily life. However, they also possess potential as a command signal in BCI systems. Accurately detecting and interpreting blinks will inherently allow users to execute commands, thereby improving the efficiency, usability, and naturalness of interactions within BCI systems.

The motivation behind this study stems from a prior experience of the author with BCI technology during an internship. This hands-on experience provided valuable insights into the challenges and opportunities in BCI research.

This study aims to address the gap by *checking the feasibility of developing a reliable method for identifying blinks using BCI* and *exploring its potential applications*. By building upon the prior internship experience and contributing to the field of BCI, this thesis seeks to advance the capabilities of BCI systems and enhance the user experience in various domains such as healthcare, gaming, communication, and assistive technology. By identifying blinks accurately, it becomes possible to use it as a control command for a more natural interaction using BCI devices.

2 Literature Review

2.1 The Human Brain

The human brain, comprising of *cerebrum, cerebellum, and brain stem*, assumes primary responsibility for controlling all bodily functions, interpreting external stimuli, and embodying the essence of the mind and soul. It governs crucial factors such as intelligence, emotions, creativity, and memory, serving as the epicenter of human cognition.

2.1.1 Cerebrum

The cerebrum is the largest part of the brain and consists of two hemispheres, namely the left and the right. Its primary role is to perform complex functions such as perceiving sensational, visual, and auditory information. It is also responsible for emotional and logical reasoning, speech, learning, and hand-eye coordination. The left hemisphere is responsible for speech and logical reasoning whereas the right hemisphere is responsible for creativity and spatial ability. (Mayfield Clinic, 2018)

As shown in Figure 1, the cerebrum can further be divided into 4 distinct lobes: frontal lobe, parietal lobe, temporal lobe, and occipital lobe.

→ ***The frontal lobe*** (blue) accounts for decision-making, problem-solving, planning, and organizing. It also controls the voluntary actions for fine motor skills. Additionally, it is involved in speech production, emotion regulation, and personality traits. Furthermore, it is responsible for attention, concentration, moral reasoning, self-awareness, and introspection.

→ ***The parietal lobe*** (yellow) is situated in the middle of the brain and is liable for interpreting languages, vision, hearing, and other sensory tasks. It also perceives the spatial and visual depth.

→ ***The temporal lobe***, (green) located on the side of the brain is essentially for audio processing, language comprehension, and memory formation. It is also responsible for processing visual stimuli and houses the Wernicke's area responsible for understanding languages.

→ ***The occipital lobe*** (red) is situated at the back of the brain and is dedicated to vision perception and procession. This allows humans to recognize shapes, colors, and movements.

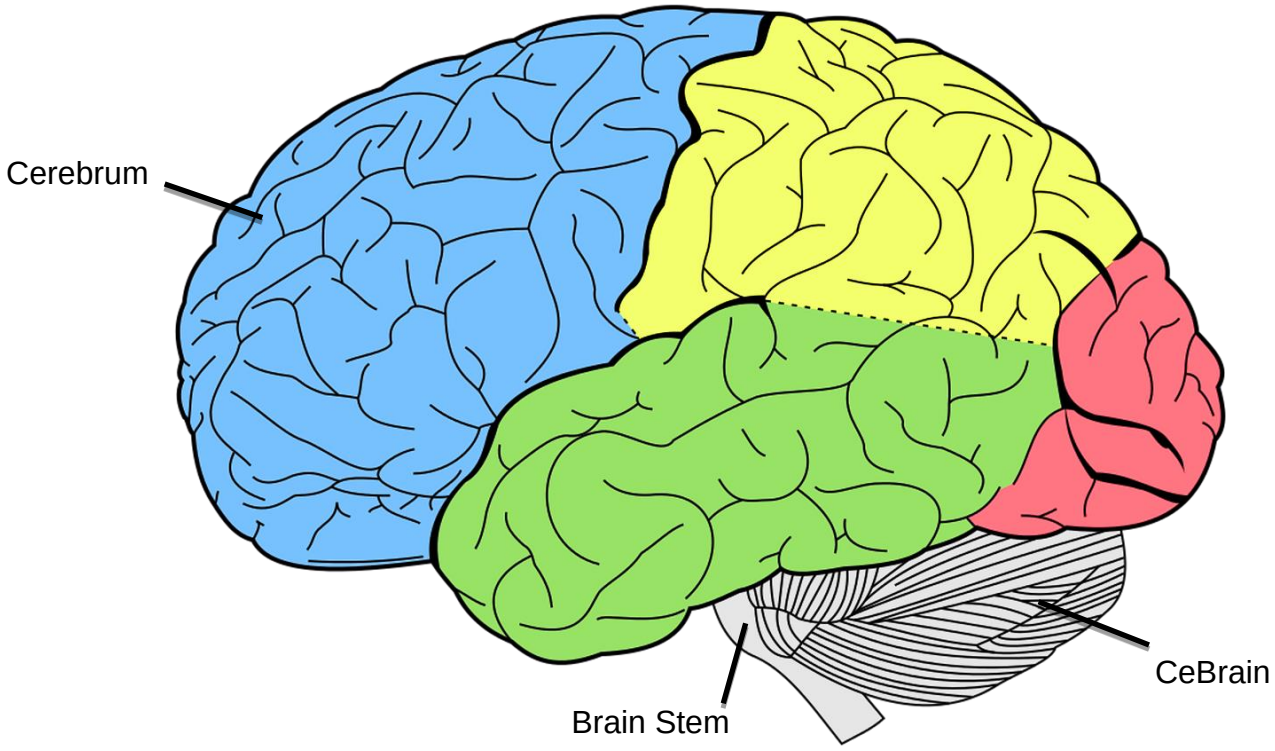


Figure 1. Lobes of the Brain (Pixabay, 2015).

2.1.2 Cerebellum

The cerebellum, situated just below the cerebrum is responsible for voluntary movement, maintaining the posture and the equilibrium of the body.

2.1.3 Brain Stem

The brain stem operates as the central relay system, facilitating communication between the cerebrum, cerebellum, and spinal cord (Mayfield Clinic, 2018). It carries out involuntary functions paramount to life sustenance including respiration, sleep cycles, heart rate, reflex actions, and digestion.

2.2 Brain-Computer Interface

Brain-Computer Interface (BCI), also known as Human-Computer Interface (HCI) is a device that can utilize different signals from the body to communicate with an external device. Three main kinds of signals are adopted as signal input: Electromyogram (EMG) for muscle activity; Electrooculogram (EOG) for eye movement and Electroencephalogram (EEG), for respiratory and brain movement (Kaya, 2021).

BCI is categorized into three different categories (Neurotech, 2012):

1. Invasive: electrodes planted directly into the cortex.
2. Semi-Invasive: electrodes are planted on the dura.
3. Non-Invasive: electrodes are placed on the scalp.

In Figure 2, we can see the different layers of a human skull for a better understanding of the anatomy of the different types of BCI.

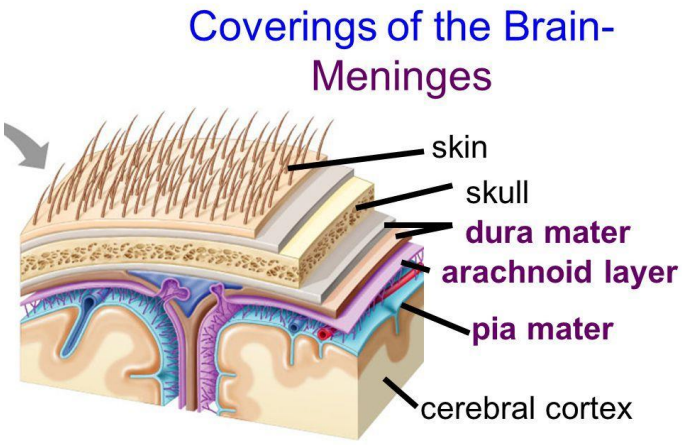


Figure 2. Dura mater, Cerebral cortex, Skin (Pinterest, n.d.).

2.3 Ultracortex Mark IV

The Ultracortex is a non-invasive 3D-printable headset that comes with electrodes, ear clips, cables, and a board (OpenBCI Documentation, 2016).

The electrodes are divided into three main types; spikey, flat, and comfort. The spikey electrodes are used to measure the EEG signals from the part that is covered with hair whereas the flat electrodes are used to measure the input from the parts that would be in direct contact with the skin (forehead). The comfort electrodes make sure there is an even weight distribution for a comfortable fit.

The ear clips are used as the reference and bias point (acts as a ground) to find the error estimate when collecting the data. The Ctyon board offers 8 channels and 16 channels with an extension board (Daisy). A USB dongle is used to connect the cyton board to the computer.

2.4 Machine Learning

2.4.1 Introduction to Machine Learning

”Machine learning is programming computers to optimize a performance criterion using example data or experience.” (Alpaydin E., 2014)

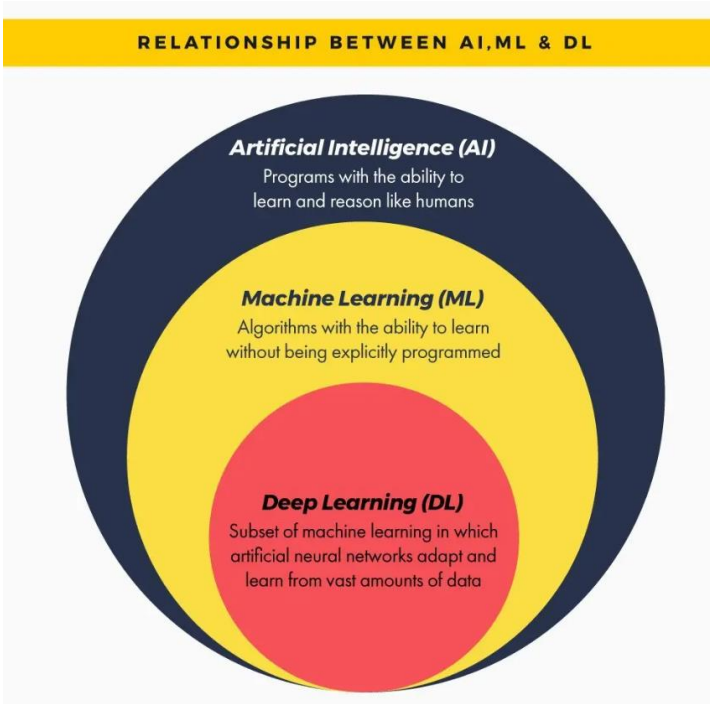


Figure 3. AI, ML, DL Venn Diagram (Corpnce, n.d.).

In ML, algorithms are designed to iteratively learn patterns and relationships from input data and improve their performance over time. This is achieved using various techniques, such as supervised learning, unsupervised learning, and reinforcement learning. *Supervised learning* involves training a model with labeled data, where the desired output is known, while *unsupervised learning* deals with discovering hidden patterns and structures in unlabeled data. *Reinforcement learning* focuses on training an agent to make sequential decisions through interactions with an environment, based on receiving rewards or penalties.

ML models can be divided into classification and regression models. *Classification models* are used for pattern recognition. Some of the commonly used classification algorithms are Binary Classification, Decision Trees, Random Forest, and Support Vector Machines(SVM). *Regression models* are used to make predictions based on the input variable. A few good regression models are Linear regression, Decision Tree, Random Forest, Support Vector Regression(SVR), and Neural Network.

2.4.2 Pre-processing

Feature selection and extraction are two extremely important processes of data pre-processing. Irrelevant features not only waste computing resources but also introduce unnecessary noise (Zhang, 2021). Feature selection refers to selecting the most relevant features that are related to the data and how they can be used to get the desired results.

The treatment of these features to get a higher correlation factor between the input and output is referred to as feature extraction. After selecting the features, the extracted features should have smaller dimensions and maximum correlation with the target.

2.5 Low Pass Filter

A low-pass filter allows low frequencies to pass and restricts the high frequencies. A 10Hz Butterworth low-pass filter (Equation 1) is implemented in the project. It allows frequencies up to 10Hz to pass and attenuates the other frequencies.

$$f_H = \frac{1}{2\pi\sqrt{R_2\ R_2 C_2 C_2}} = \frac{1}{2\pi\sqrt{R^2 C^2}} = \frac{1}{2\pi RC}$$

Equation 1. Butterworth Filter Formula (Circuit Digest, 2019).

3 Methodology

The project can be broken down into four main parts: Data Collection, Visualisation, Data Pre-processing, and Model Training and Evaluation (Figure 4).

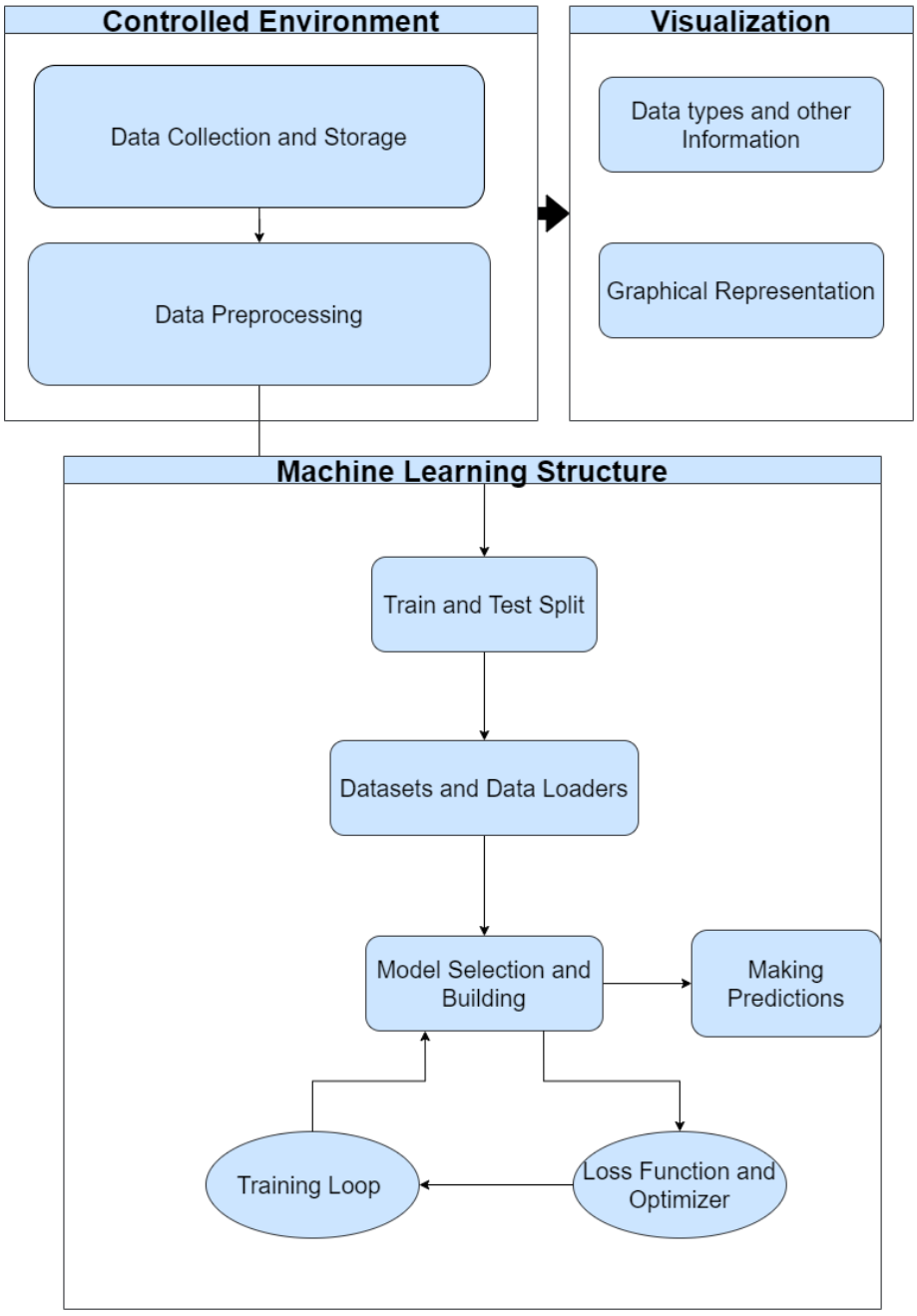


Figure 4. Research Design.

3.1 Data Collection

Data collection comprises of defining labels, the number of readings required per label, and determining the duration of each reading.

The labels are made based on the duration of a voluntary and involuntary eye blink. An involuntary eye blink takes less than or equal to one second whereas, when measuring a voluntary eye blink, we can control the duration. The readings are recorded for 4 labels: involuntary eye blinks, eyes closed, single voluntary eye blinks, and voluntary eye blinks at equal intervals.

Using a Python script, raw EEG data is collected from 8 channels over a duration of 5 minutes through the Ultracortex Mark IV headset. Optical activity takes part in two main parts of the brain, the frontal and the occipital lobe. Subsequently, channels 3 to 6 (Figure 5) are turned off to make the conditions ideal.

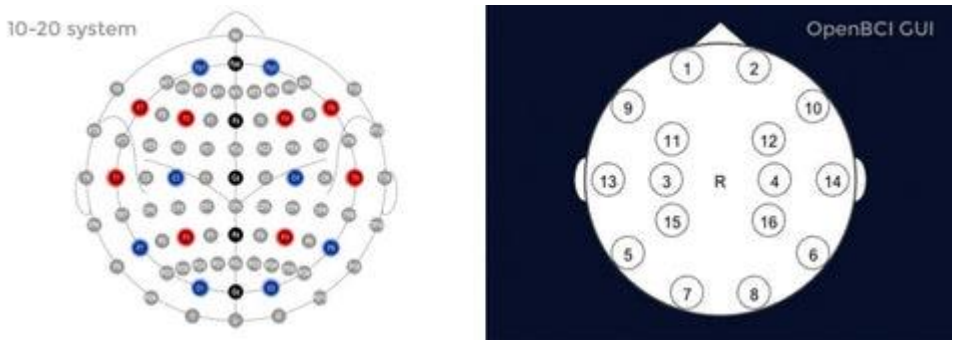


Figure 5. Head Plot (OpenBCI Documentation, 2016).

The output is stored in a CSV file under the appropriate label in the ’*raw_data*’ folder. This step is repeated six times for each label. Next, the noise and the channels that are turned off are removed from the CSV file and saved in the '*filtered_data*' folder. The difference of consecutive readings for each channel is calculated and four new channels are introduced. Lastly, the output is passed through a 10Hz low-pass filter and saved in the '*signal_filtered_data*’ folder (Figure 6).

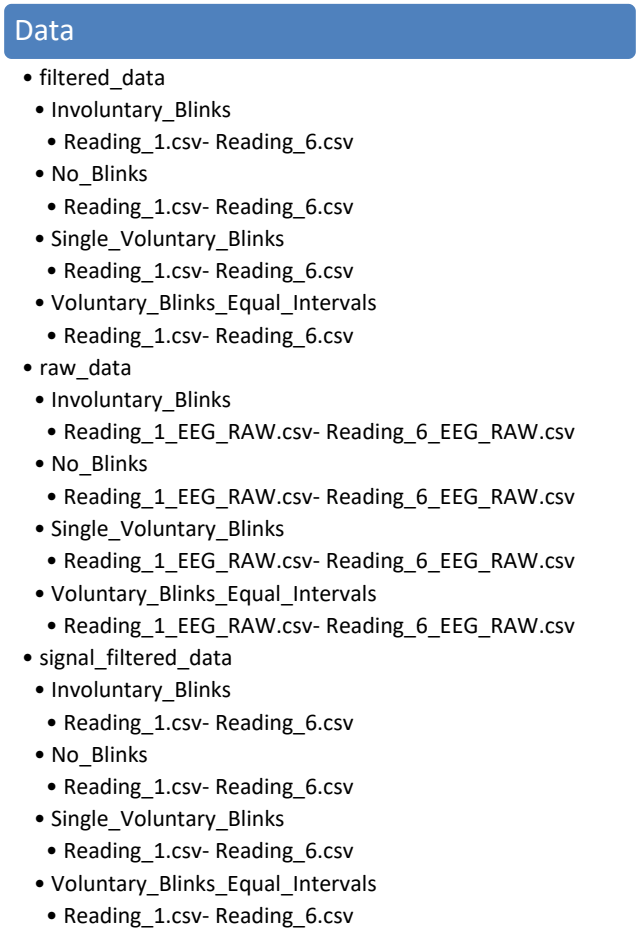
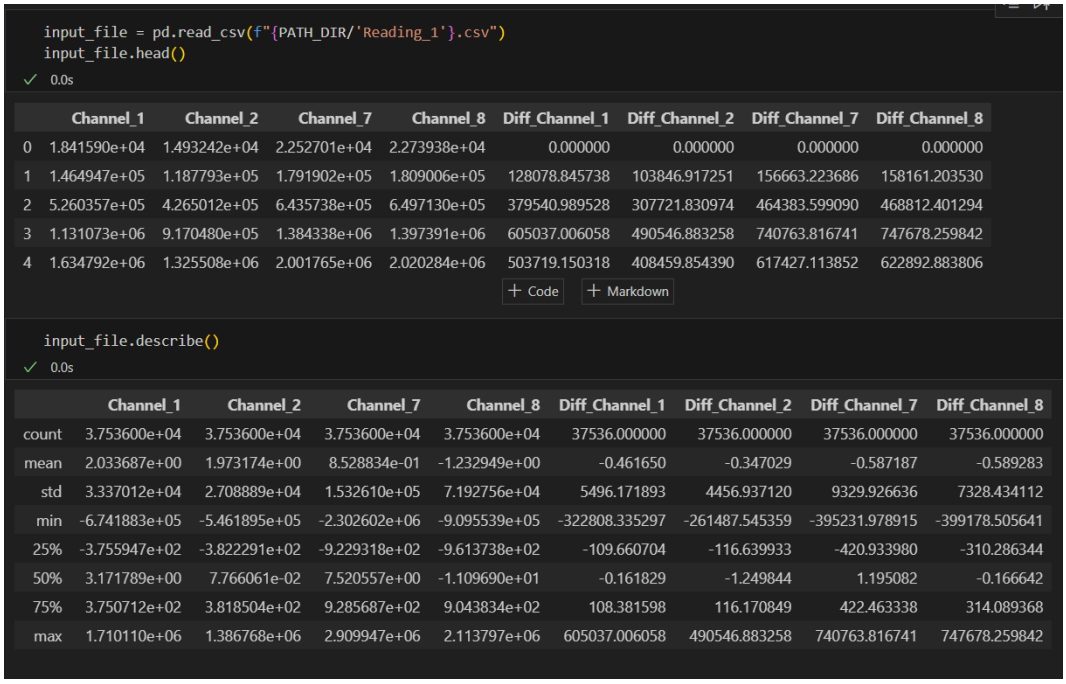


Figure 6. Data Storage Structure.

3.2 Visualization

The data collected from the headset, in the raw format does not give enough details and is not comprehensive. Therefore, to gain an insight from these readings, we remove the noise and pass it through a low-pass filter to extract information. The low-pass filter scales the readings, which makes it easier to plot and identify the patterns. A brief description of the first reading from each label is given below along with a graph after pre-processing is completed.



```
input_file = pd.read_csv(f"{PATH_DIR/'Reading_1'}.csv")
input_file.head()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
0	1.841590e+04	1.493242e+04	2.252701e+04	2.273938e+04	0.000000	0.000000	0.000000	0.000000
1	1.464947e+05	1.187793e+05	1.791902e+05	1.809006e+05	128078.845738	103846.917251	156663.223686	158161.203530
2	5.260357e+05	4.265012e+05	6.435738e+05	6.497130e+05	379540.989528	307721.830974	464383.599090	468812.401294
3	1.131073e+06	9.170480e+05	1.384338e+06	1.397391e+06	605037.006058	490546.883258	740763.816741	747678.259842
4	1.634792e+06	1.325508e+06	2.001765e+06	2.020284e+06	503719.150318	408459.854390	617427.113852	622892.883806

+ Code + Markdown

```
input_file.describe()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
count	3.753600e+04	3.753600e+04	3.753600e+04	3.753600e+04	37536.000000	37536.000000	37536.000000	37536.000000
mean	2.033687e+00	1.973174e+00	8.528834e-01	-1.232949e+00	-0.461650	-0.347029	-0.587187	-0.589283
std	3.337012e+04	2.708889e+04	1.532610e+05	7.192756e+04	5496.171893	4456.937120	9329.926636	7328.434112
min	-6.741883e+05	-5.461895e+05	-2.302602e+06	-9.095539e+05	-322808.335297	-261487.545359	-395231.978915	-399178.505641
25%	-3.755947e+02	-3.822291e+02	-9.229318e+02	-9.613738e+02	-109.660704	-116.639933	-420.933980	-310.286344
50%	3.171789e+00	7.766061e-02	7.520557e+00	-1.109690e+01	-0.161829	-1.249844	1.195082	-0.166642
75%	3.750712e+02	3.818504e+02	9.285687e+02	9.043834e+02	108.381598	116.170849	422.463338	314.089368
max	1.710110e+06	1.386768e+06	2.909947e+06	2.113797e+06	605037.006058	490546.883258	740763.816741	747678.259842

Figure 7. Closed Eyes Data Info.

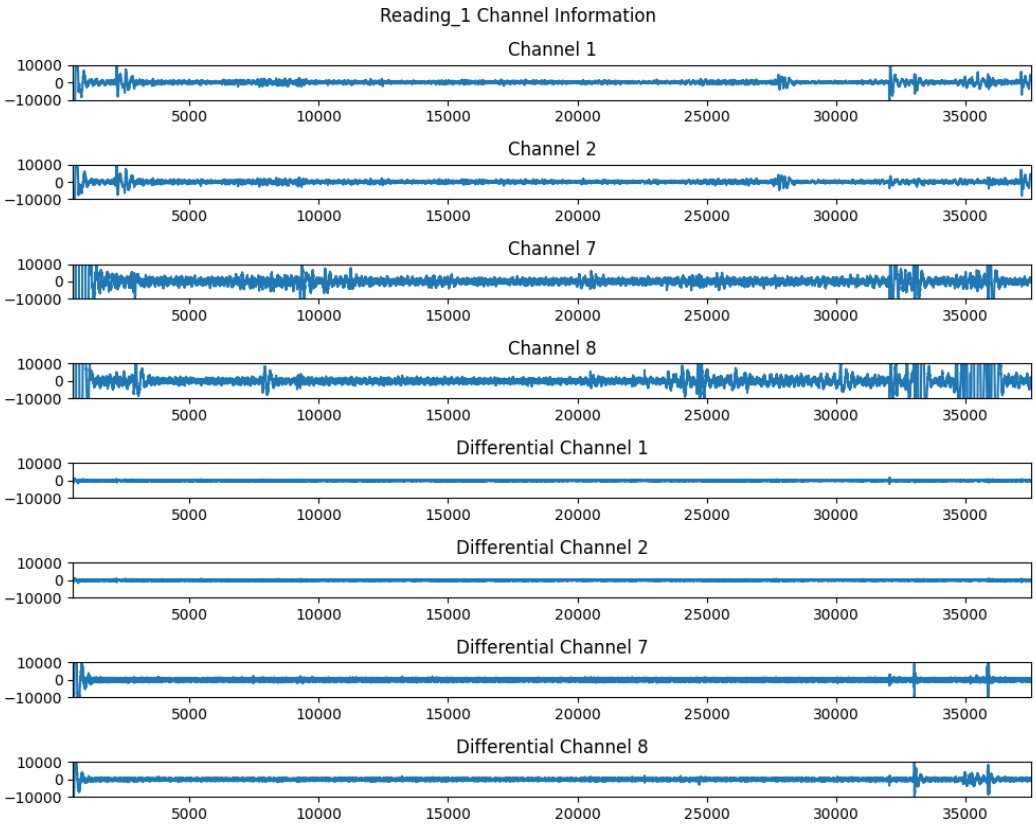


Figure 8. Closed Eyes Graph.

```
input_file = pd.read_csv(f"{PATH_DIR/'Reading_1'}.csv")
input_file.head()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
0	2.617018e+04	2.529541e+04	1.699028e+05	-7.041298e+04	0.000000	0.000000	0.000000e+00	0.000000e+00
1	1.820175e+05	1.759267e+05	1.181602e+06	-4.896112e+05	155847.282701	150631.268972	1.011699e+06	-4.191982e+05
2	5.655745e+05	5.466299e+05	3.671171e+06	-1.520926e+06	383557.022676	370703.169906	2.489570e+06	-1.031315e+06
3	1.041928e+06	1.006992e+06	6.762341e+06	-2.800780e+06	476353.134067	460361.964358	3.091170e+06	-1.279854e+06
4	1.281506e+06	1.238463e+06	8.315648e+06	-3.442168e+06	239578.065405	231470.973874	1.553307e+06	-6.413886e+05

```
input_file.describe()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
count	3.767800e+04	3.767800e+04	3.767800e+04	3.767800e+04	37678.000000	37678.000000	3.767800e+04	3.767800e+04
mean	-3.513073e-01	-5.196926e-01	-2.012420e-04	1.141660e-01	-0.679351	-0.658769	-4.509336e+00	1.880452e+00
std	2.378300e+04	2.307629e+04	1.547747e+05	6.547210e+04	4442.563237	4293.405680	2.881146e+04	1.194370e+04
min	-4.800341e+05	-4.638489e+05	-3.104629e+06	-3.442168e+06	-286891.346395	-277544.740244	-1.864382e+06	-1.279854e+06
25%	-5.728014e+02	-5.577014e+02	-9.843729e-04	-9.104948e+02	-114.606699	-122.462311	-1.461407e-05	-2.915287e+02
50%	-4.445847e+00	-2.026938e+01	-9.066075e-05	6.409635e+00	-1.501108	-1.498136	1.248263e-07	9.598355e-01
75%	5.645580e+02	5.318586e+02	7.900508e-04	8.979586e+02	110.384554	120.346738	1.507137e-05	2.898518e+02
max	1.281506e+06	1.238463e+06	8.315648e+06	1.275384e+06	476353.134067	460361.964358	3.091170e+06	7.752252e+05

Figure 9. Involuntary Eye Blink Data Info.

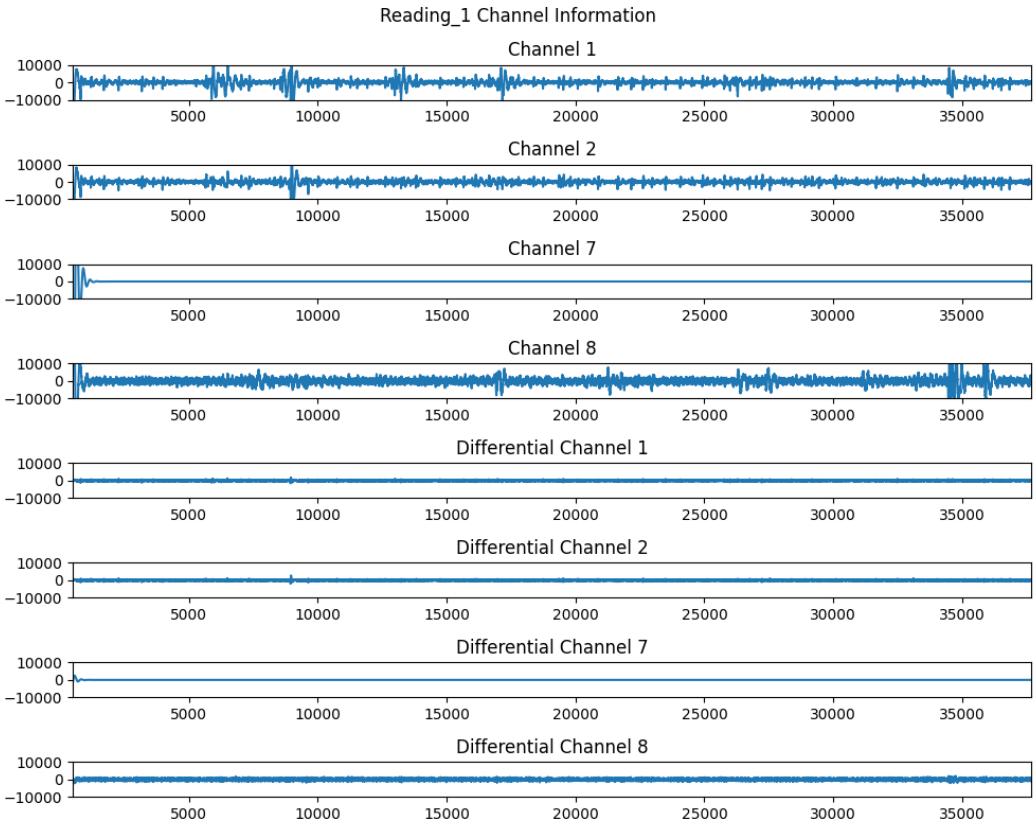


Figure 10. Involuntary Eye Blink Graph.

```
input_file = pd.read_csv(f"{PATH_DIR/'Reading_1'}.csv")
input_file.head()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
0	3.130426e+04	2.549692e+04	-2.824869e+04	-5.395898e+04	0.000000e+00	0.000000	0.000000	0.000000e+00
1	2.489950e+05	2.028049e+05	-2.230911e+05	-4.259243e+05	2.176907e+05	177307.983996	-194842.444438	-3.719653e+05
2	8.940291e+05	7.281873e+05	-7.956832e+05	-1.520176e+06	6.450341e+05	525382.440408	-572592.072406	-1.094252e+06
3	1.922193e+06	1.565639e+06	-1.701512e+06	-3.256258e+06	1.028164e+06	837451.763144	-905828.778655	-1.736082e+06
4	2.777950e+06	2.262669e+06	-2.450317e+06	-4.699842e+06	8.557568e+05	697029.746264	-748804.671187	-1.443584e+06

```
input_file.describe()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
count	3.751900e+04	3.751900e+04	3.751900e+04	3.751900e+04	3.751900e+04	37519.000000	37519.000000	3.751900e+04
mean	4.435455e-01	4.220945e-01	2.681878e+00	2.207124e+00	-8.643854e-01	-0.719702	0.297243	8.985161e-01
std	5.658259e+04	4.609370e+04	5.488942e+04	9.889174e+04	9.339400e+03	7608.431684	9373.753368	1.731023e+04
min	-1.143680e+06	-9.316442e+05	-2.558896e+06	-4.918959e+06	-5.492581e+05	-447337.402765	-905828.778655	-1.736082e+06
25%	-1.280956e+03	-1.131830e+03	-1.155705e+04	-1.232083e+04	-1.436522e+02	-145.049392	-2772.600230	-5.252513e+03
50%	-1.511140e+01	-4.140537e+01	1.887758e+01	-8.399077e+01	-2.756312e-01	-1.626768	-4.813193	-4.157208e+01
75%	1.135181e+03	1.017979e+03	1.149195e+04	1.210358e+04	1.414421e+02	141.470823	2801.416914	5.286731e+03
max	2.905328e+06	2.366420e+06	1.048851e+06	1.977493e+06	1.028164e+06	837451.763144	478059.016340	9.240304e+05

Figure 11. Single Voluntary Eye Blink Data Info.

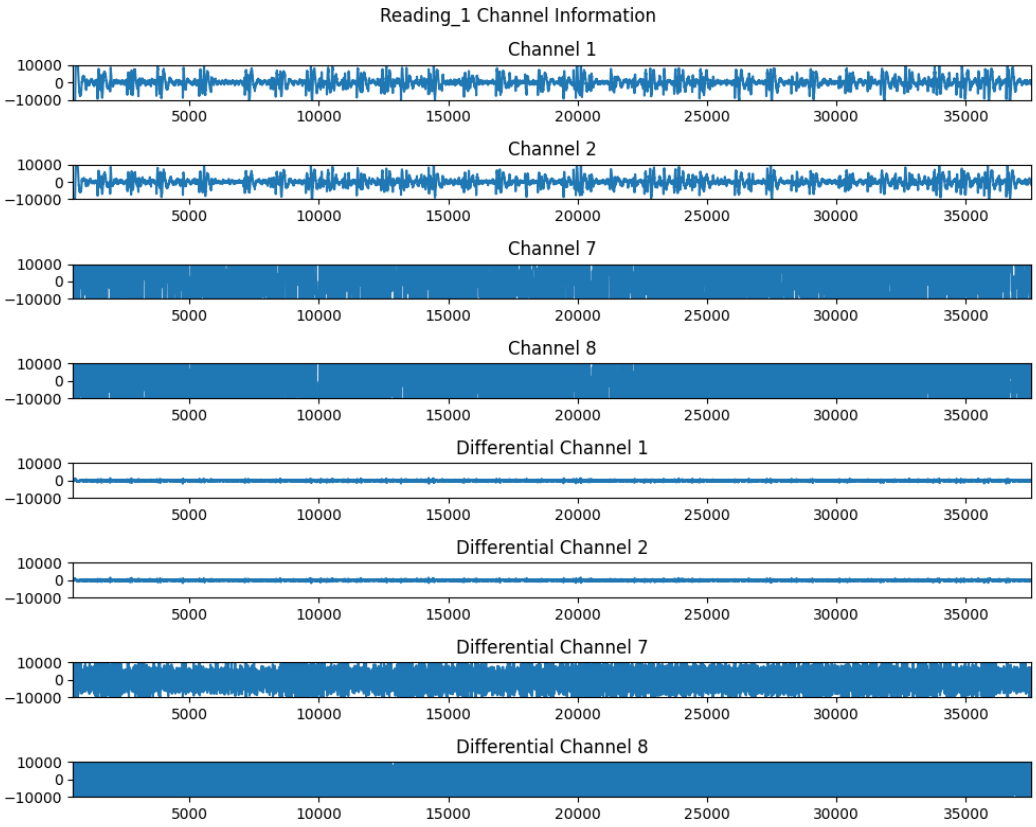


Figure 12. Single Voluntary Eye Blink Graph.

```
input_file = pd.read_csv(f"{PATH_DIR/'Reading_1'}.csv")
input_file.head()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
0	4.775840e+04	5.464494e+04	-6.413294e+04	-9.329446e+04	0.000000	0.000000	0.000000e+00	0.000000e+00
1	3.319466e+05	3.802364e+05	-4.462145e+05	-6.488807e+05	284188.184875	325591.467822	-3.820815e+05	-5.555862e+05
2	1.030714e+06	1.182199e+06	-1.387388e+06	-2.016581e+06	698767.065664	801963.041269	-9.411735e+05	-1.367700e+06
3	1.897464e+06	2.179390e+06	-2.559313e+06	-3.717343e+06	866750.358875	997190.781871	-1.171925e+06	-1.700762e+06
4	2.332066e+06	2.682236e+06	-3.156528e+06	-4.579263e+06	434602.262276	502846.168493	-5.972151e+05	-8.619199e+05

```
input_file.describe()
```

✓ 0.0s

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8
count	3.725000e+04	3.725000e+04	3.725000e+04	3.725000e+04	37250.000000	37250.000000	3.725000e+04	3.725000e+04
mean	1.026982e+00	2.592386e-01	-5.194813e+00	-4.859237e+00	-1.293539	-1.470247	1.903022e+00	2.694386e+00
std	4.370172e+04	5.022981e+04	7.121150e+04	9.478287e+04	8143.284110	9369.565748	1.196889e+04	1.664265e+04
min	-8.692005e+05	-1.000329e+06	-3.156528e+06	-4.579263e+06	-522770.757718	-601101.556241	-1.171925e+06	-1.700762e+06
25%	-1.418140e+03	-1.335685e+03	-1.010478e+04	-1.013590e+04	-315.650028	-237.983276	-1.796736e+03	-1.975101e+03
50%	-4.583291e+00	-1.960308e+01	-1.588726e+02	-1.162870e+02	-1.931549	-1.813302	2.622841e+01	3.387894e+01
75%	1.389303e+03	1.295061e+03	1.010566e+04	1.008757e+04	308.049101	234.519877	1.855945e+03	2.040381e+03
max	2.332066e+06	2.682236e+06	1.223014e+06	1.749918e+06	866750.358875	997190.781871	6.903512e+05	1.009883e+06

Figure 13. Single Voluntary Eye Blink (Equal Interval) Data Info.

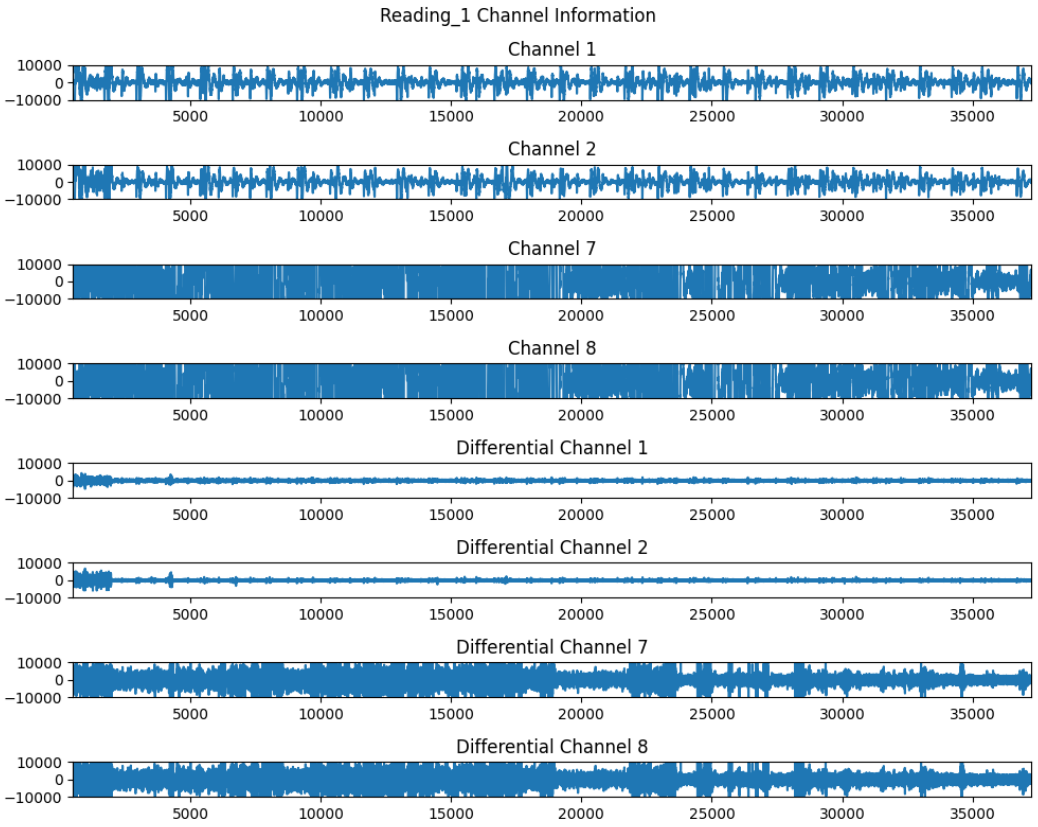


Figure 14. Single Voluntary Eye Blink (Equal Interval) Graph.

After visualizing the graphs, the change in the amplitude is evident whenever a blink occurs. This will be the basis for the model to differentiate the various labels and identify when an eye blink occurs.

3.3 Data pre-processing

The file structure (Figure 6) defined above cannot be passed directly through the model. As a result, all the readings from a single label are stacked together and a new column called '*label*' is introduced to the CSV files. After the numerical assignment for each label, we create a dataset that is an integration of all 24 readings. The four labels are represented in a numerical format, as mentioned below.

- 0 -> Eyes Closed
- 1 -> Involuntary Eye Blinks
- 2 -> Single Voluntary Eye Blinks
- 3 -> Single Voluntary Eye Blinks at Equal Intervals

Following the ML project composition, the dataset is split into a train and test dataset for ease of verification once the model is trained. The training data is shuffled to prevent the model from learning how the readings are stored.

Additionally, dataloaders are made to maintain the readability and modularity of the code. By using Equation 2, we define the 'batch size' hyperparameter as 1250 for the dataloader to pass the readings through the model at 10-second intervals.

Batch Size = Sampling Rate * Required Duration

Equation 2. Batch Size.

3.4 Model Training and Evaluation

Neural Network is a technique for building a computer program that learns from the data. It is based loosely on how the human brain works (Smilkov & Carter, n.d.).

Neural networks require a large dataset and are good at detecting small changes. Two CNN and RNN models are trained and validated on the processed dataset. The simple CNN model has one convolution and one classification layer. The multi-layer CNN model comprises three convolution layers and a classification layer. The RNN model has one RNN and one linear classification layer. The models use cross-entropy loss (Equation 3) as a loss function and Adam optimizer (Equation 4) with a learning rate of 0.001.

$$H(p,q) = -\sum_{i=0}^{n} p(x_i) log(q(x_i))$$

Equation 3. Cross Entropy Loss Formula (Stack Overflow, 2017).

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective)
λ (weight decay), $amsgrad$, $maximize$
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ **to** ... **do**
  if $maximize$:
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / \big(1 - \beta_1^t\big)$
  $\widehat{v_t} \leftarrow v_t / \big(1 - \beta_2^t\big)$
  if $amsgrad$
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)$

return θ_t

Equation 4. Adam Optimizer Formula (PyTorch Contributors, n.d.).

For a classification model, the data loss decreases and the model accuracy increases over the training period. By plotting the train-test loss and accuracy, we can identify the best and the worst-performing models (Figure 15).

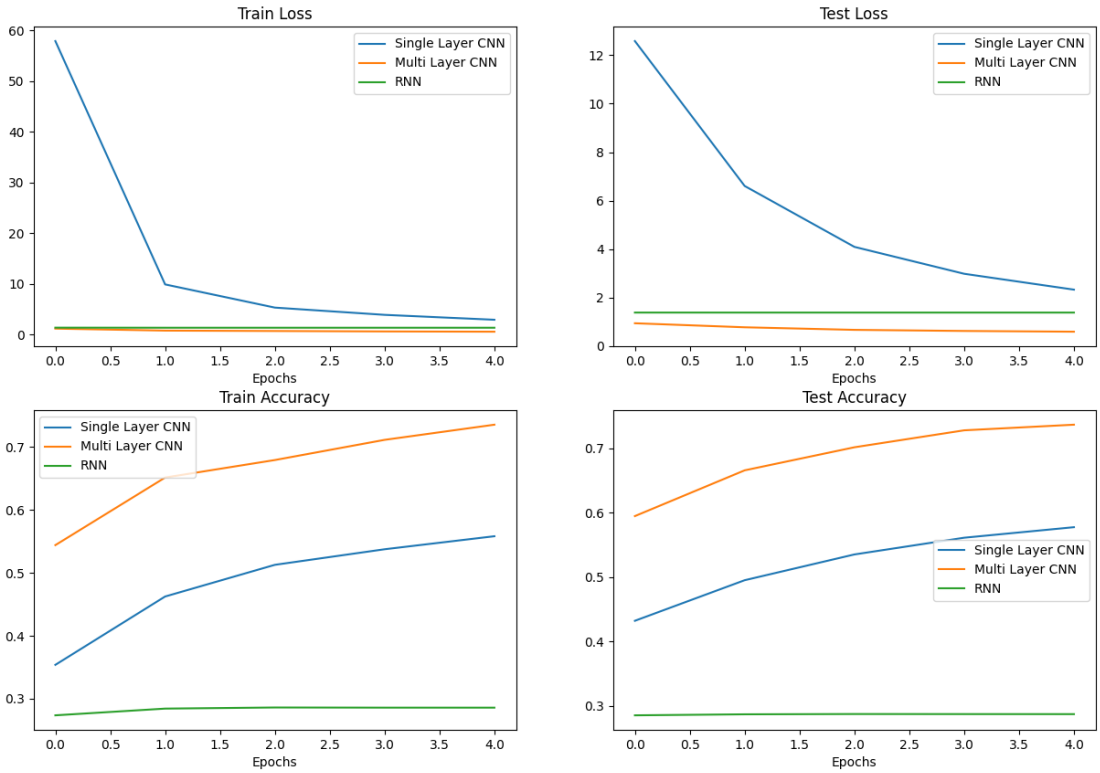


Figure 15. Validating the models.

4 Results

The combined dataset of the 24 readings has approximately 916,000 readings and 9 columns. The first four columns are the channel readings, the next four columns are the difference of consecutive readings of the channel readings and the last column has the labels for identifying and training the models.

	Channel_1	Channel_2	Channel_7	Channel_8	Diff_Channel_1	Diff_Channel_2	Diff_Channel_7	Diff_Channel_8	label
0	1.841590e+04	1.493242e+04	2.252701e+04	2.273938e+04	0.000000	0.000000	0.000000	0.000000	0
1	1.464947e+05	1.187793e+05	1.791902e+05	1.809006e+05	128078.845738	103846.917251	156663.223686	158161.203530	0
2	5.260357e+05	4.265012e+05	6.435738e+05	6.497130e+05	379540.989528	307721.830974	464383.599090	468812.401294	0
3	1.131073e+06	9.170480e+05	1.384338e+06	1.397391e+06	605037.006058	490546.883258	740763.816741	747678.259842	0
4	1.634792e+06	1.325508e+06	2.001765e+06	2.020284e+06	503719.150318	408459.854390	617427.113852	622892.883806	0
...	...	...	...	...	...	...	...	...	...
916220	5.972225e+03	5.076769e+03	2.984130e-04	2.984130e-04	68.675226	-119.798172	-0.000020	-0.000020	3
916221	6.064471e+03	5.024549e+03	2.782509e-04	2.782509e-04	92.246419	-52.220135	-0.000020	-0.000020	3
916222	6.342233e+03	5.282487e+03	2.578748e-04	2.578748e-04	277.762113	257.938625	-0.000020	-0.000020	3
916223	6.713630e+03	5.761552e+03	2.373021e-04	2.373021e-04	371.396124	479.064826	-0.000021	-0.000021	3
916224	6.994776e+03	6.124050e+03	2.165505e-04	2.165505e-04	281.146541	362.498277	-0.000021	-0.000021	3

Figure 16. Combined Dataset Overview.

Each model is trained for 5 epochs. The training and testing results are stored in a data frame for all the models (Figure 17).

```
single_layer_CNN_model:
    train_loss  train_acc  test_loss  test_acc
0   27.428698   0.421642   8.420415  0.490222
1    5.931011   0.507170   4.005494  0.529990
2    3.520999   0.539633   3.024728  0.543453
3    2.732707   0.555063   2.207703  0.549317
4    2.337790   0.563636   1.828976  0.569239
 Training time: 75.57

multi_layer_CNN_model:
    train_loss  train_acc  test_loss  test_acc
0    1.218681   0.522846   0.875742  0.620197
1    0.877056   0.631129   1.053505  0.576716
2    0.752121   0.670840   0.662469  0.705925
3    0.660838   0.704341   0.636034  0.712921
4    0.594411   0.733546   0.565731  0.747050
 Training time: 85.89

rnn_model:
    train_loss  train_acc  test_loss  test_acc
0    1.396789   0.277649   1.382721  0.286116
1    1.382830   0.285847   1.382432  0.287332
2    1.382747   0.285915   1.382513  0.287332
3    1.382731   0.285851   1.382460  0.287332
4    1.382720   0.286107   1.382606  0.287332
 Training time: 76.40
```

Figure 17. Model results.

The single-layer CNN model achieves the highest accuracy of 56.92% on the testing data. The ***best-performing model*** is the multi-layered CNN model which can reach the accuracy of 74.70% on the testing data. The RNN model reaches the best accuracy of 28.73% on the testing data. Even if the model randomly guesses the label it should be able to correctly identify the label 25% of the time (Equation 5), therefore it is safe to say the RNN model is not learning anything.

Random Accuracy = 100 / (Number of labels)

Equation 5. Randomly Calculated Accuracy.

5 Conclusion

The aim of the thesis was to assess the feasibility of constructing a dependable method for identifying eye blinks using BCI and exploring its potential applications. It justified the proof of concept and provided a foundation for enhancing various BCI applications in healthcare, gaming, communication, and assistive technology. It encompassed data collection, EEG pre-processing, and model training, utilizing EEG data from the Ultracortex Mark IV headset labeled for four categories. Three machine learning models were evaluated, with the multi-layer CNN achieving a top performance of 74.70%.

5.1 Interpretation of Results

The recorded values are for NVIDIA GeForce GTX 1650 Ti, Intel core i7-10750H CPU, and 16 GB RAM. Utilizing a faster GPU and CPU will expedite the model training time.

The multi-layered CNN model is the best-performing model. The accuracy of the model can be improved by making changes to the hyperparameters (such as increasing the hidden layers in the model). It has a training time of 85.89 seconds. The training time can be neglected at the current state of the model, however, if the model is implemented in real-time, it needs to be observed and kept to the lowest possible value to reduce delay.

5.2 Future Work

The recorded readings are for a single individual which may cause bias in real-world scenarios. It is critical to take readings from multiple individuals to improve the credibility of the model. The EEG data is recorded in a controlled environment in near to optimal conditions, taking readings with interference (such as walking and recording the EEG values) can be experimented with. As a part of feature

selection, only the four most relevant nodes have been utilized. Using the other nodes can be one of the test conditions. Lastly, implementing the code in real time will advance the project exponentially.

References

Alpaydin E., 2014. Introduction to Machine Learning. In: P. E. Central, ed. *Introduction to Machine Learning.* Cambridge: MIT Press.

Circuit Digest, 2019. *Butterworth Filter Formula (Equation 1.),* s.l.: s.n.

Corpnce, n.d. *Relationship between AI, ML, DL (Figure 3.),* s.l.: s.n.

Kaya, İ., 2021. A Brief Summary of EEG Artifact Handling. In: V. Asadpour, ed. *Brain-Computer Interface.* s.l.:Intech Open.

Mayfield Clinic, 2018. *Anatomy of Brain.* [Online]
Available at: https://mayfieldclinic.com/pe-anatbrain.htm
[Accessed 29 May 2023].

Neurotech, 2012. *NeurotechEDU.* [Online]
Available at: http://learn.neurotechedu.com/introtobci/
[Accessed 25 October 2023].

OpenBCI Documentation, 2016. *Ultracortex Mark IV* | *OpenBCI Documentation.* [Online]
Available at: https://docs.openbci.com/AddOns/Headwear/MarkIV/
[Accessed 27 May 2023].

Pinterest, n.d. *Dura mater, Cerebral cortex, Skin (Figure 2.),* s.l.: s.n.

Pixabay, 2015. *Lobes of the Brain (Figure 1),* s.l.: s.n.

PyTorch Contributors, n.d. *Adam Optimizer Formula (Equation 4.),* s.l.: s.n.

Smilkov, D. & Carter, S., n.d. *TensorFlow Neural Network Playground.* [Online]
Available at:
https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4&se

ed=0.24201&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xS
[Accessed 4 October 2023].

Stack Overflow, 2017. *Cross Entropy Loss Formula (Equation 3.),* s.l.: s.n.

Zhang, J., 2021. *Machine Learning: Feature Selection and Extraction with Examples.* [Online]
Available at: https://medium.com/nerd-for-tech/machine-learning-feature-selection-and-extraction-with-examples-80e3e2c2e1a1
[Accessed 20 September 2023].