MACHINE LEARNING

MINI - PROJECT REPORT

Topic: Sleep Stages Classification using EEG Waves

TE COMPS B 2019 - 2020

Palash Rathod

60004170076

Parag Vaid

60004170077

Siddharth Sanghavi

60004170108

Project Guide: Prof. Kriti Srivastava

Table of Contents

1. Problem Statement

Brain research has been highlighted as an area of interest in recent years. Study in this domain has the potential to impact many important areas, from the detection, treatment and increased understanding of diseases such as Alzheimer's and Epilepsy, to the neural control of devices to aid the handicapped, to a greater understanding of how the human brain functions on a basic level. The classification of brain signals recorded by imaging devices using machine learning approaches is a very powerful tool in many of these areas of research. For example, machine learning techniques show promise in the early detection of Alzheimer’s or giving warning before an epileptic seizure.

Deep learning is seldom used in the classification of electroencephalography (EEG) signals, despite achieving state of the art classification accuracies in other spatial and time series data. Instead, most research has continued to use manual feature extraction followed by a traditional classifier, such as SVMs or logistic regression. This is largely due to the low number of samples per experiment, high-dimensional nature of the data, and the difficulty in finding appropriate deep learning architectures for classification of EEG signals.

Thus, our study aims at making use of Deep Learning architectures for the classification of brain signals or EEG. To prove that classification of brain waves can also be done using deep learning architecture, we have chosen a dataset consisting of sleep patterns of a certain number of subjects and tried to classify the sleep stages depending on the EEG signals of the subjects.

2. Introduction

Sleep is the primary function of the brain that has an essential role in every individual's performance, learning capabilities and physical movement. Humans spend around one third of their life sleeping. Human sleep is a regular state of rest for the body in which the eyes are not only usually closed, but also have several nervous centers being inactive; hence, rendering the person either partially or completely unconscious and making the brain a less complicated network. Sleep diseases such as insomnia and obstructive sleep apnea have a negative impact on the quality of human life. Therefore, the need to distinguish human sleep stages is very important for diagnosis and treatment of sleep disorders such as sleep apnea, insomnia and narcolepsy.

Studies of sleep assist doctors to diagnose sleep disorders and provide the baseline for appropriate follow up. Clinical sleep study design is based on polysomnography (PSG) in which several biological signals are acquired while the patient is asleep, including electroencephalography (EEG) for monitoring brain activity, electrooculogram (EOG) for eye movements and electromyogram (EMG) to measure muscle tone. The measurements are used to classify sleep score, i.e., classify the sleep stages the patient goes through during the study and assess the existence of any dysfunction. Manually scoring sleep stages is a tedious task requiring sleep experts to visually inspect PSG data recorded during the whole sleep study. With the increasing accessibility of EEG signals there is a growing interest in sleep quantification based on EEG alone. The EEG signal often presents significant bursts of rhythmic components.

Thus, there has been considerable effort over the past years to develop machine learning methods for automatic sleep scoring and the development of efficient sleep stage identification techniques using single channel EEG signal would be very beneficial to analysis, diagnosis and treatment of sleep disorders.

The American Academy of Sleep Medicine (AASM) recommends segmentation of sleep in five stages:

- **W (wakefulness):** alpha (8–12 Hz) rhythm is present, high-amplitude muscle contractions and movement artefacts on EMG and eye blinking on EOG, which can also be appreciated in low frequency EEG (0.5-2 Hz).
- **Nl (Non-REM 1):** alpha (8–12 Hz) rhythm is attenuated and replaced by mixed frequency theta signal (4–7 Hz), decrease in muscle tone and slow eye movements.
- **N2 (Non-REM 2):** presents K-complexes (negative peak followed by a positive complex and a final negative voltage) in the < 1.5 Hz range and sleep spindles (burst of oscillatory waves) at sigma (12–15 Hz) band.
- **N3 (Non-REM 3):** slow wave activity exists (0.5-3 Hz), eye movements are unusual and EMG tone is low.
- **R (REM):** existence of rapid eye movements clearly visible in EOG, relatively low-amplitude and mixed-frequency activity in EEG, and presents the lowest muscle tone on EMG.

2.1 Dataset Description

The dataset used in this project is the Physionet Sleep-EDF dataset which is a publicly available dataset from the PhysioNet database.

The EEG Sleep-EDF database is a collection of 61 males and females. But here we are taking the data for 20 subjects consisting of 10 males and 10 females. Its associated hypnogram files contain sleep patterns corresponding to each subject. All hypnograms were manually scored by well-trained technicians according to Rechtschaffen and Kales manual. The hypnograms represent an event for segments (epochs) of 30 seconds.

The Sleep-EDF dataset includes whole-night polysomnograms (PSGs) sleep recordings at the sampling rate of 100 Hz. Each record contains EEG (from Fpz-Cz and Pz-Oz electrode locations), EOG, chin electromyography (EMG), and event markers. Few records often also contain oro-nasal respiration and rectal body temperature.

Each stage was considered to belong to a different class (stage). The classes include wake (W), rapid eye movement (REM), N1, N2, N3, N4 and M (movement time). According to American Academy of Sleep Medicine (AASM) standard, we integrated the stages of N3 and N4 in one class named N3 and excluded the M (movement time) stage to have five sleep stages: W, N1, N2, N3, REM corresponding to classes 0, 1, 2, 3 and 4 respectively. Stages 1 and 2 - 3 are the light sleep time in which the stage N1 is the lightest stage and has a short period time. The stage N2 - N3 takes longer than the stage N1, including approximately 40-60% of total sleep time. The stage N3 is called deep sleep and the REM (Stage 5) is known as the dreaming stage taking 90 - 120 minutes per night. Considering different stage time periods results in having imbalanced stage numbers in the sleep datasets.

Wake (W)	Stage 1 (N1)	Stage 2 (N2)	Stage 3 (N3)	REM (R)
37410	1240	9200	2981	3756

Table. 1. Sleep stages class label count

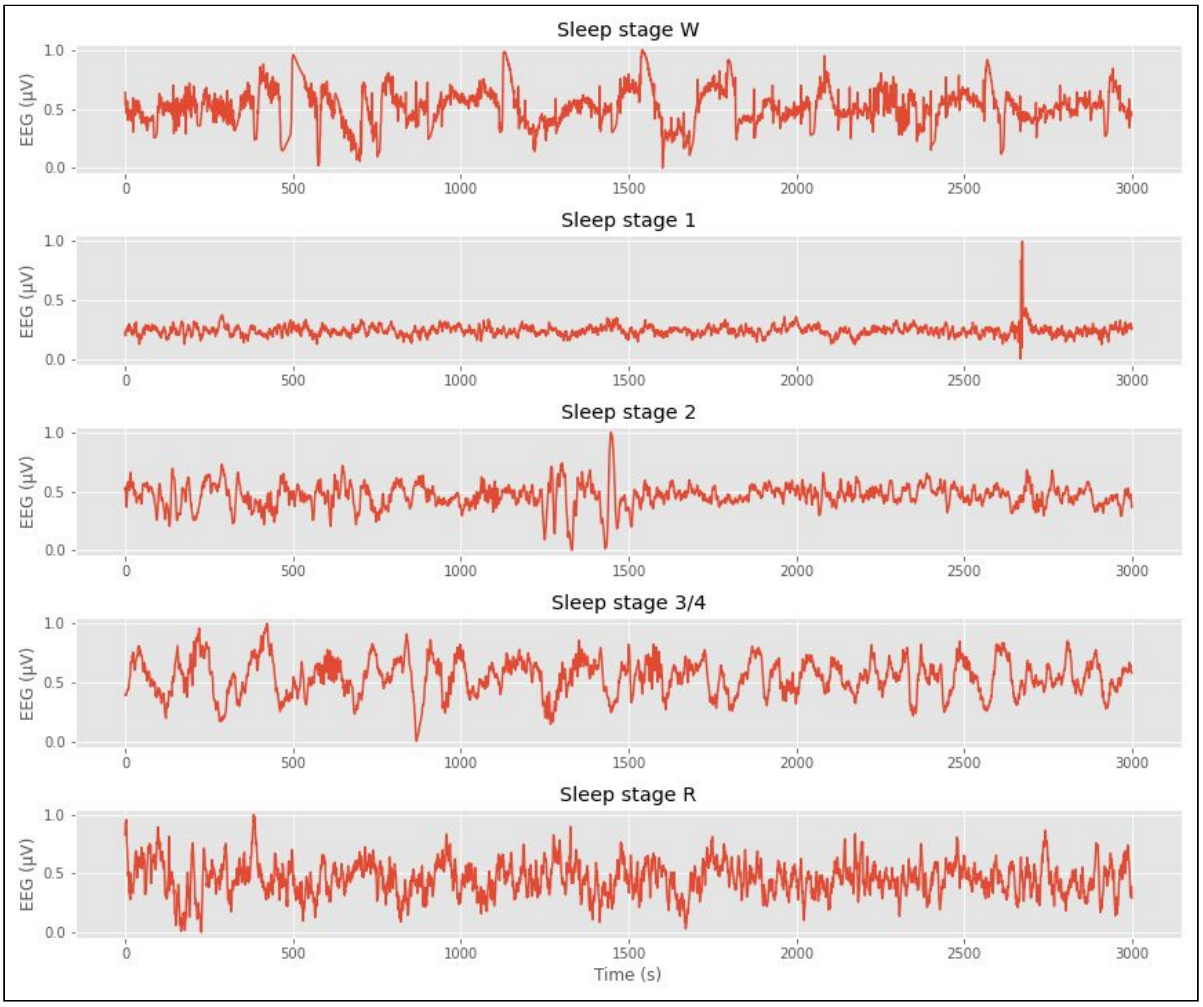

Fig. 1. EEG vs. Time for each sleep stages class

2.2 EEG Monitoring Hardware

Electroencephalography (EEG) is a technique with over a hundred years of history, and while it was originally used more strictly in the fields of psychology, medicine, and neuroscience, it is widely used today in gaming, human-computer-interaction, neuromarketing, simulations, and beyond.

Due to this increased use and demand of high-quality EEG devices, there are now numerous companies that are able to cater to the specific needs of EEG users. Each offers something unique for the consumer – whether it's the number of channels, a stationary or portable device, the predefined metrics offered, or of course the price. Here we mention economically viable and research oriented EEG headsets as follows:

2.2.1 NeuroSky MindWave Mobile 2

Fig. 2. NeuroSky MindWave Mobile 2

Specifications	● Portable and Lightweight EEG brainwave headset ● Automatic Wireless computer pairing ● Single AAA battery with 6 - 8 hours run time ● Direct connect to dry electrode ● 1 EEG channel + Reference + Ground ● 512Hz sampling rate at 12 bits ● 3Hz - 100Hz frequency range ● Extremely low-level signal detection ● Advanced filter with high noise immunity ● Supports Windows, Mac OS ● ESD Protection: 4kV Contact Discharge; 8kV Air ● Max Power Consumption: 15mA @ 3.3V ● Operating voltage 2.97V - 3.63V ● 8 bit UART (Serial): 1200, 9600, 57600 baud ● Price: $177
Data Outputs	● RAW EEG Signal ● Attention ● Meditation ● Delta, Theta, Low Alpha, High Alpha, Low Beta, High Beta and Gamma waves

2.2.2 OpenBCI Ganglion Board (with the EEG Headband Kit)

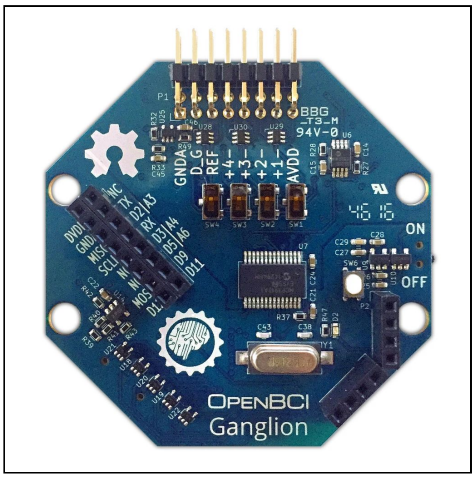

Fig. 3. OpenBCI Ganglion Board

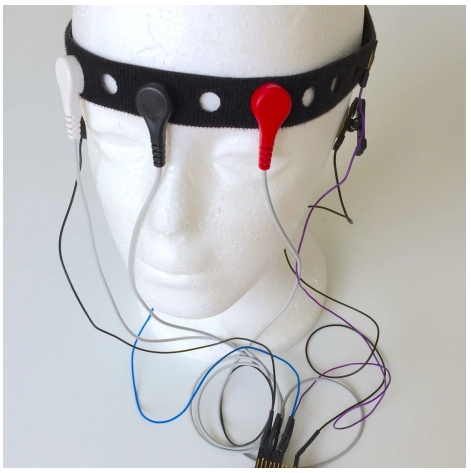

Fig. 4. EEG Headband Kit

Specifications	• High-quality, affordable bio-sensing device • 6V AA battery • Wireless Connectivity with Bluetooth v4.0 • 4 channel high-impedance differential inputs, a driven ground (DRL), a positive voltage supply (Vdd), and a negative voltage supply (Vss) • 200Hz sampling rate on each of the 4 channels • Compatible with OpenBCI's free open-source software • Power with 3.3V - 6V DC battery • Has a bandpass filter which cuts out any frequency below 0.3 Hz • Current Draw: 14mA when idle, 15mA connected and streaming data • 3 lead wires terminating in snaps for flat EEG snap electrodes • 5 standard lead wires terminating in clips for dry comb electrodes • 2 ear clips with replacement electrodes • 1 headband, one-size-fits-all • Price: $250
Data Outputs	• RAW EEG Signal • The kit allows three frontal cortex measurements (F7, AF7, Fp1, Fpz, Fp2, AF8, F8) via the three included lead wires with flat EEG snap electrodes. • The kit facilitates temporal, parietal, and occipital measurements via five standard leads. Measurement can be taken at including but not limited to FT7/FT8, T7/T8, TP7/TP8, P7/P8, PO7/PO8, O1/O2, and Oz nodes, depending where you position the dry comb electrodes. • Delta, Theta, Low Alpha, High Alpha, Low Beta, High Beta and Gamma waves

2.2.3 EMOTIV Insight 5 Channel Mobile Brainwear

Fig. 5. EMOTIV Insight Mobile

Specifications	● Designed for self-quantification, brain-computer interface and field research ● Elegant, lightweight and user-friendly design ● Wireless Connectivity with Bluetooth Low Energy ● Internal Lithium Polymer battery 480mAh giving usage up to 4 - 8 hours ● 5 channels: AF3, AF4, T7, T8, Pz ● 2 references: CMS/DRL references on left mastoid process ● 128Hz sampling rate channel ● Resolution of 14 bits with 1 LSB = 0.51μV ● Frequency response of 0.5-43Hz, digital notch filters at 50Hz and 60Hz ● Filtering: Built in digital 5th order Sinc filter ● Dynamic range (input referred): 8400 μV(pp) ● Sensor material: Hydrophilic semi-dry polymer ● Supports Windows, Mac OS, iOS, Android ● Price: $299
Data Outputs	● RAW EEG Signal ● Delta, Theta, Low Alpha, High Alpha, Low Beta, High Beta and Gamma waves

3. Review of Literature

3.1 Introduction to the identification of brain waves based on their frequency

At the root of all our thoughts, emotions and behaviors is the communication between neurons within our brains. It is a handy analogy to think of brainwaves as musical notes - the low frequency waves are like a deeply penetrating drum beat, while the higher frequency brain waves are more like a subtle high-pitched flute. Like a symphony, the higher and lower frequencies link and cohere with each other through harmonics. Our brainwaves change according to what we are doing and feeling. When slower brain waves are dominant, we can feel tired, slow, sluggish, or dreamy. The higher frequencies are dominant when we feel wired, or hyper-alert. The descriptions that follow are only broad descriptions - in practice things are far more complex, and brain waves reflect different aspects when they occur in different locations in the brain. This paper helped us to gain a holistic view of the concept of brain waves and the Brain Computer Interface (BCI).

To make use of brain waves, the first task is to collect the brain waves or ‘Signal Acquisition’. As stated earlier, brain cells or the neurons communicate with each other. This communication is in the form of electrical impulses. An electroencephalogram (EEG) is a test used to evaluate the electrical activity in the brain. EEG is the most prevalent method of signal acquisition. They are acquired using electrodes that follow the International 10/20 system which is a placement strategy which ensures ample coverage over all parts of the head. Brain signals are acquired by electrodes on the surface of the head. These signals are then digitized and processed to clean and denoise the data to enhance relevant information embedded in the signals.

The EEG signals are evaluated based on the frequency of the waves. Each brain wave has a different frequency. The readings of each type of frequency have different meanings. The distribution of brain waves by frequency of band waves is as follows:

Name of the band wave	Frequency of the band wave (Hz)
Alpha α	8-13
Beta β	13 - 30
Delta δ	0.5 - 4
Gamma γ	> 30
Theta θ	4 - 8

Table. 2. Different types of EEG waves

Alpha α waves connect the gap between our conscious thinking and subconscious mind. It helps us to calm down or promote the feeling of relaxation. Beta β are active in waking state. Delta δ waves occur during meditation in a state of deep sleep or coma. Gamma γ waves are important for learning, memory, and information processing. Theta θ waves are involved in sleep or daydreaming.

Thus, after analyzing this paper, we came to the conclusion of using EEG for our problem statement due to the following reasons:

- Firstly, EEG captures cognitive dynamics in the time frame in which cognition occurs. Cognitive, perceptual, linguistic, emotional, and motor processes are fast. EEG is well suited to capture these fast, dynamic, and temporally sequenced cognitive events.

- Secondly, EEG directly measures neural activity. The voltage fluctuations that are measured by EEG are direct reflections of biophysical phenomena at the level of the population of neurons.

- Thirdly, EEG signals are multidimensional. Although, it might be conceptualized as two dimensional, in fact EEG data comprise at least four dimensions: time, space, frequency, and power (the strength of frequency-band-specific activity) and phase (the timing of the activity).

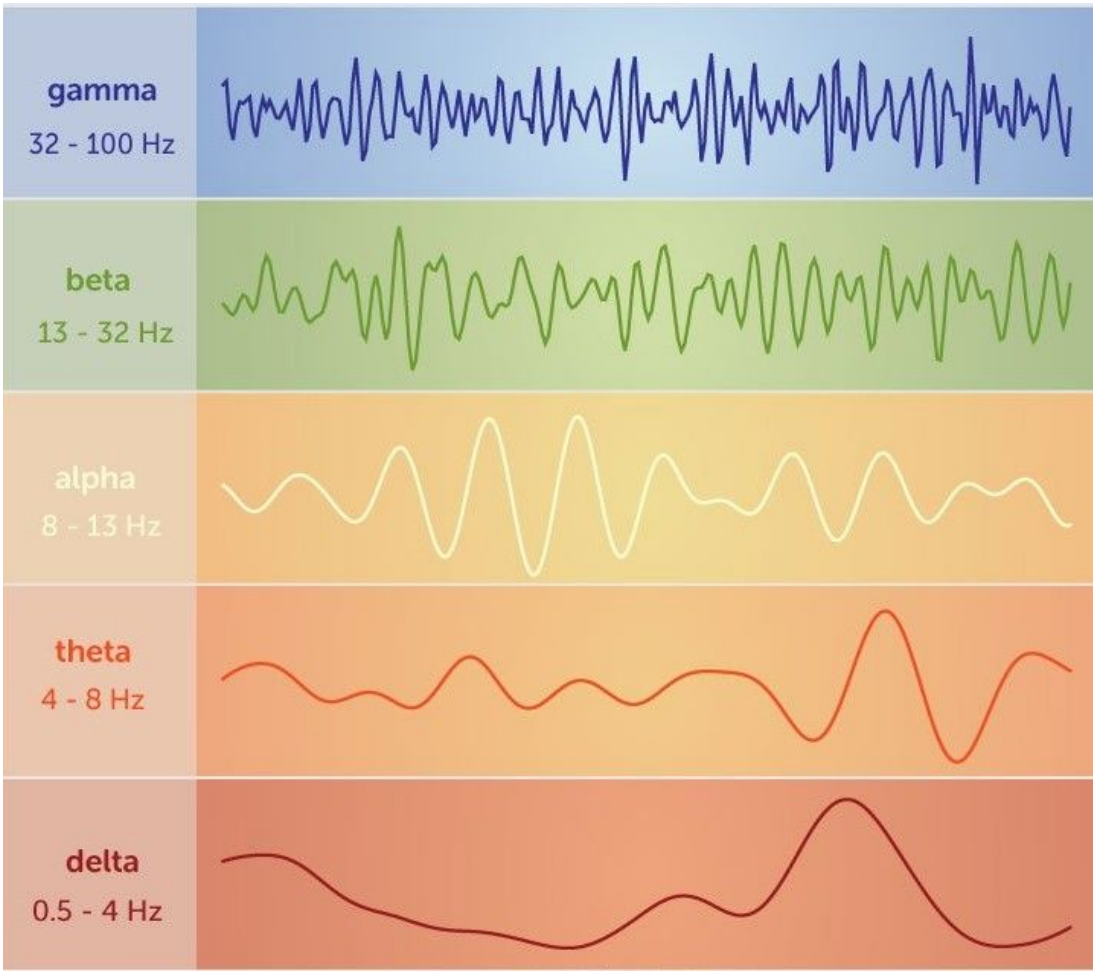

Fig. 6. Five major EEG waves distinguished by their frequency range

3.2 Deep learning for Electroencephalogram (EEG) Classification tasks: a review

There are many tools which can be used to analyze the EEG signals for applications in neuroscience, neural engineering, and even commercial applications. Recent trends have shown an enormous rise in use of Machine Learning to gain relevant information from the signals. In addition to that, the availability of extensive EEG datasets has made it possible to apply various Machine Learning algorithms and Deep Learning architectures for improvised classifications and predictions. The aim of this paper was to compare various Deep Learning architectures that can be used for analysis of EEG. The dataset used consisted of five different tasks and for each type of task specific input formulation, major characteristics and end classifier recommendations were described.

Typical EEG classification pipeline involves artifact removal, feature extraction and classification. On the most basic level, an EEG dataset consists of a 2D (time and channel) matrix of real values that represent brain-generated potentials recorded on the scalp associated with specific task conditions. This highly structured form makes EEG data suitable for machine learning and deep learning architectures.

This paper gives an in-depth comparison of various deep learning architectures that can be used to analyze EEG signals. It also shows comparison between various input formulations for each of the five tasks present in the dataset. These input formulations include Signal Values, Calculated Features, and Images. The general deep learning strategies included Convolutional Neural Network (CNN), Hybrid-CNN (H-CNN), Deep Belief Network (DBN), Recurrent Neural Network (RNN), Stacked Auto Encoders (SAE), Multi-Layer Perceptron (MLP) and Hybrid-MLP (H-MLP). Overall use of various architectures helped us to select the appropriate architecture along with the type of input formulation for our problem statement.

Thus, the study mentioned in this paper was successfully conducted. After a deep-rooted analysis of various deep learning architectures on their dataset, the paper provides a recommendation diagram which can be useful in selecting the optimum architecture depending on the use case and the problem statement.

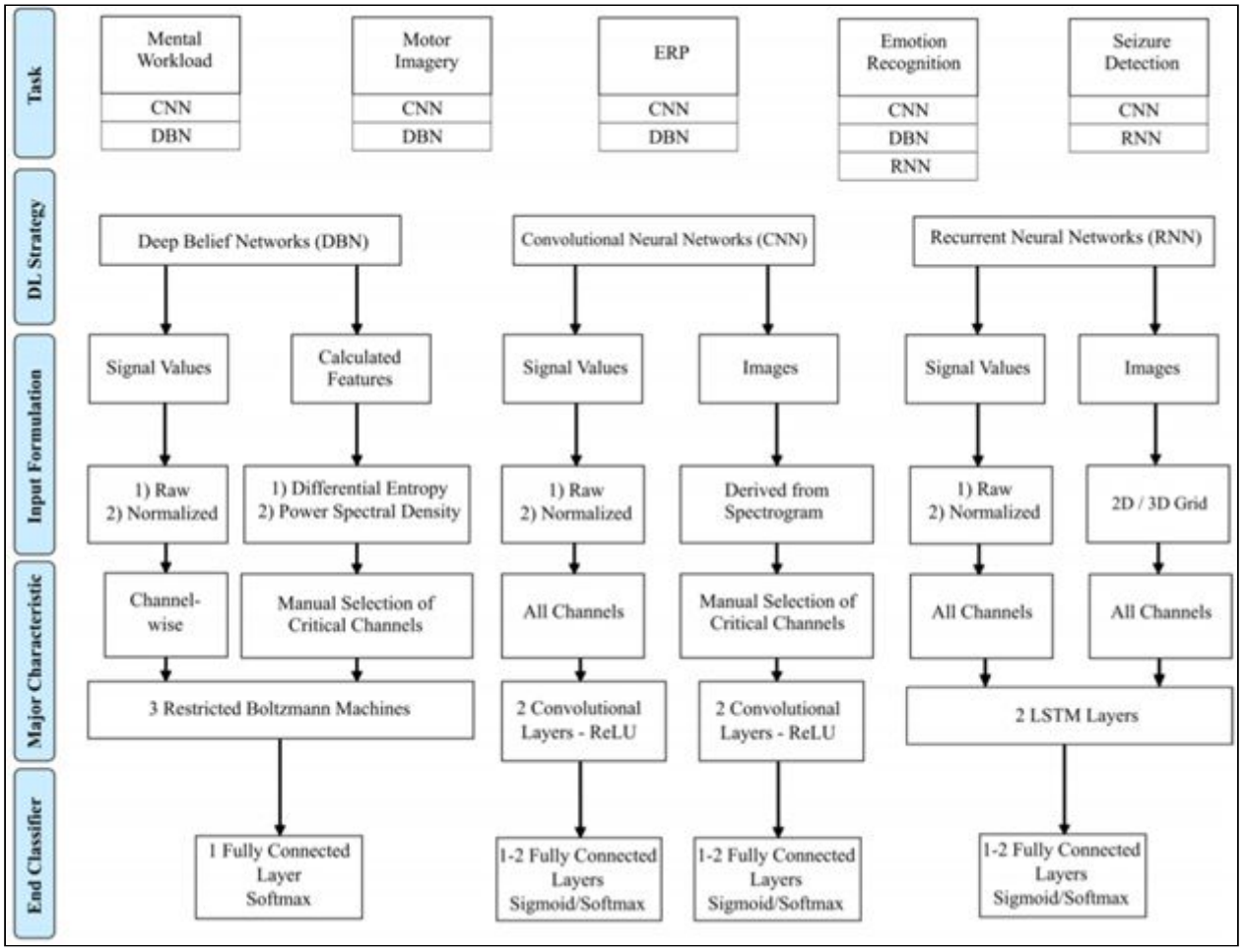

Fig. 7. Task-specific Deep Learning Recommendation diagram

On referring to this diagram, we decided to use two input formulations namely images and signal values. Images were used in the form of a spectrogram while the signal values were used in the form of time series data. 2D CNN architecture was applied for spectrograms whereas 1D CNN was applied for time series data for our problem statement.

3.3 Spectrum Analysis of EEG signals using CNN to model patient's Consciousness Level based on Anaesthesiologists' experience

Scientifically, the brainwave signal can be measured using Electroencephalography (EEG). The EEG signals are characterized by amplitude (voltage) and frequency. The frequency varies in each band, Delta range within 0.5 to 4 Hz, Theta range from 4 to 8 Hz, Alpha range from 8 to 13 Hz and

Beta range from 13 to 32 Hz. However, the raw EEG signal needs to be analysed in order to extract useful information for specific research.

Analyzing EEGs is mathematically complex, as it is a signal with multiple frequencies changing over time, often with a lot of embedded noise. Usually, EEG signal is extracted via three methods, namely time, frequency or time-frequency based. Generally, EEG raw signals are in time-based format. To analyse in frequency-based, typically the signals need to be transformed into Fourier Transform (FT).

The most often technique used to analyse signals in time-frequency based is Short Time Fourier Transform (STFT). The STFT is to perform a FT on the signal, then mapping the signal into a two-dimensional function of frequency and time. Most of the EEG signal analysis have been carried out using time-frequency based, and mainly in signal processing areas , thus, very few in image processing areas.In this paper, EEG signals were analysed based on time-frequency image processing technique or called spectrogram.

We're accustomed to seeing a waveform that displays changes in a signal's amplitude over time. A spectrogram, however, displays changes in the frequencies in a signal over time. Amplitude is then represented on a third dimension with variable brightness or color.

A spectrogram is a visual way of representing the signal strength, or "loudness", of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is more or less energy at, for example, 2 Hz vs 10 Hz, but one can also see how energy levels vary over time. A common format of spectrogram is a graph with two geometric dimensions: one axis represents time, and the other axis represents frequency; a third dimension indicating the amplitude of a particular frequency at a particular time is represented by the intensity or color of each point in the image. An example of an eeg spectrogram is shown below.

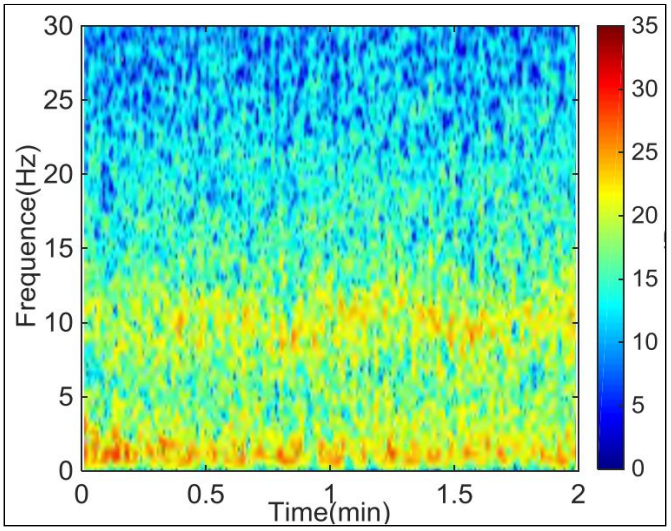

Fig. 8. EEG Spectrogram

In neural network literature, many techniques are proposed to extract features from the temporal and spatial domain of the physiological signal. However, only some work has been done to extract features simultaneously from time and frequency domains.

Spectrogram of the signal allows us to extract more precise low-frequency information for the long signal duration or high-frequency information from short duration signal for better classification.
Time-frequency based feature extraction has been successfully applied with promising results in many important fields such as biomedical, radar, speech, audio, video and image processing for various classification applications.

A novel approach using spectrogram images produced from EEG signals as the input dataset of Convolution Neural Network (CNN) is proposed in this paper.

Consider s(t) is an input EEG signal. Now the EEG signal is segmented using windowing function (ω), which produces a matrix where [x[0], x[1],...,x[k − 1]]^T is the first column, [x[1], x[2],...,x[k]]^T is the second column, and so forth. The S is a k × (n − k + 1) matrix which is the highly redundant representation of s(t). The spectrogram of s(t) with window size m can be calculated using the STFT approach. The STFT is achieved by sliding the window to the nonstationary EEG signal using the equation given below.

$$S(n,\ \omega) = \sum_{m=-\infty}^{\infty} s(m)\omega(n-m)\ e^{-jwm}$$

where s is the input signal, ω is the window function, m is the window interval centered around zero. Also n and k are the time and frequency domain indices. It is noteworthy that the rows of the spectrogram are indexed by frequency while the columns are indexed by time and, hence, each location on the spectrogram corresponds to a point in frequency and time. Further, the obtained spectrogram is a matrix which can be seen as an image. In the proposed method this generated image is used as input in the input image layer of CNN.

In the proposed method, the image input layer receives a spectrogram of the EEG signal from training images, and transforms this image into specific dimensions so that it can be delivered to the convolution layer correctly.

The best trained CNN model achieved an accuracy of 93.50%, the proposed method achieved good accuracy and provided acceptable classification rate with reasonable speed. The main observation needs to be taken into account is that classification accuracy and speed both are important for real-time applications.The main limitation of the proposed method is it takes much training time with lots of training data.

3.4 1D Convolutional Neural Network Models for Sleep Arousal Detection

During the last decade, Convolutional Neural Networks (CNNs) have become the de facto standard for various Computer Vision and Machine Learning operations. Deep 2D CNNs with many hidden layers and millions of parameters have the ability to learn complex objects and patterns providing that they can be trained on a massive size visual database with ground-truth labels. With proper training, this unique ability makes them the primary tool for various engineering applications for 2D signals such as images and video frames. Yet, this may not be a viable option in numerous applications over 1D signals like EEG, ECG, EMG, etc. especially when the training data is scarce or application-specific. To address this issue, 1D CNNs have recently been proposed and immediately achieved state-of-the-art performance levels in several applications such as personalized biomedical data classification and early diagnosis, structural health monitoring, anomaly detection and identification in power electronics and motor-fault detection.

Convolution in the time domain is an extension of the dot product, in which the dot product is computed repeatedly over time. Convolution can also be performed over space, but for EEG analyses, convolution over time is used.

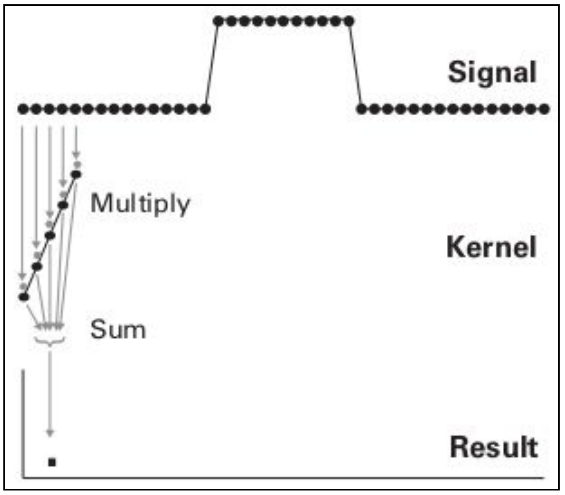

Fig. 8. Start of 1D Convolution

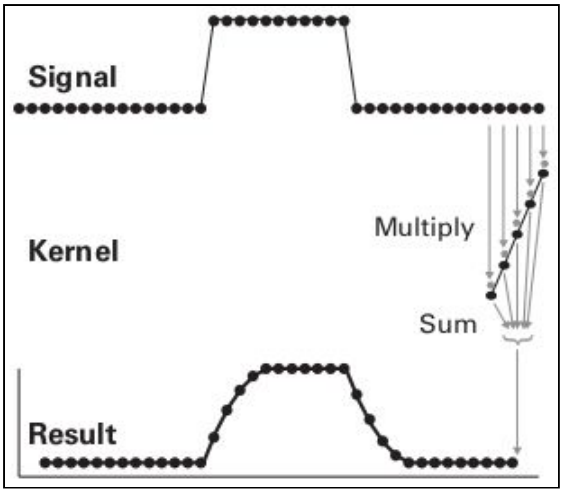

Fig. 9. End of 1D Convolution

Let us assume a square wave to be the signal and a five-point linear decay function will be the kernel as shown in Fig. 8. Imagine taking the kernel and then dragging it forward in time so it passes over the signal. As the kernel passes over the signal, it drags on the signal, and the result of the drag is something that is neither the signal nor the kernel, but looks a bit like each of them. Thus, conceptually, convolution entails sliding the kernel along the data and obtaining a result that shows what features the signal and the kernel have in common. This “commonality” at each time point is computed as the mapping between the kernel and the signal, that is, the dot product.

First line up the kernel with the data as shown in Fig. 8. Then, ````````````the dot product by multiplying each point in the kernel by each corresponding point in the data and summing over all of the multiplications. The dot product between the kernel and the corresponding part of the signal is then placed in a new vector in the position corresponding to the center of the kernel. It is convenient but not necessary to use kernels that have an odd number of data points, so that there is a center point. This is the first step of convolution. Now the kernel is shifted to the right by one time step, but the signal does not move. Repeat the dot product procedure and store this result, again in the position corresponding to the center of the kernel, which is now one time step to the right. Notice that the dot product is computed with the same kernel but a slightly different part of the signal. This process of computing the dot product and then moving the kernel one step to the right continues until the kernel has reached the end of the data or completeness. The mathematical formula for convolution is given as follows:

$$(a * b)_k = \sum_{i=1}^{n} a_i b_{k-i}$$

In this equation, a and b are the two vectors for which the dot product is computed, k indicates that this equation refers to the convolution at time point k , and i corresponds to the elements in the equal-length parts of vectors a and b . Vector b is the kernel because it is flipped backward.

In EEG data analyses, convolution is used to isolate channel specific activity and to localize that specific activity in time. This is done by convolving filters or kernels with EEG data in the 1D CNN. As the convolution kernel is dragged along the EEG data (the convolution signal), it reveals when and to what extent the EEG data contain features that look like that filter/kernel. When convolution is repeated on the same EEG data using 1D kernels of different sizes, a different amplitude-time representation can be formed.

Studies have shown that for certain applications 1D CNNs are advantageous and thus preferable to their 2D counterparts in dealing with 1D signals due to the following reasons:

- Rather than matrix operations, Forward and Backward Propagation in 1D CNNs require simple array operations. This means that the computational complexity of 1D CNNs is significantly lower than 2D CNNs.

- Recent studies show that 1D CNNs with relatively shallow architectures (i.e. small number of hidden layers and neurons) are able to learn challenging tasks involving 1D signals. On the other hand, 2D CNNs usually require deeper architectures to handle such tasks. Obviously, networks with shallow architectures are much easier to train and implement.

- Usually, training deep 2D CNNs requires special hardware setup (e.g. Cloud computing or GPU farms). On the other hand, any CPU implementation over a standard computer is feasible and relatively fast for training compact 1D CNNs with few hidden layers (e.g. 2 or less) and neurons (e.g. < 50).

- Due to their low computational requirements, compact 1D CNNs are well-suited for real-time and low-cost applications especially on mobile or hand-held devices.

4. Scope

Successful conduction of this study would imply that deep learning techniques can also be used for classification of EEG signals apart from traditional machine learning algorithms. The results of this study would serve as a proof of concept and pave the way for classification of EEG signals in many areas, such as the detection of disease state or brain computer interfaces.

As far as this study is concerned, better results can be achieved by making use of various other deep learning techniques. Such techniques can include use of Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM) or even an integrated architecture of LSTM and CNN.

The accuracy can be further improved by selecting a more extensive dataset. The data points in the dataset chosen for this study were not distributed evenly amongst the various class labels. A more uniform distribution of the data points would ensure a much better accuracy. Modifications can also be done on the input formulation of the data. Images and raw data were used in this study as input. Apart from these, calculated features can also be used as input which include Singular Value Decomposition (SVD), Wavelet, etc.

A more in-depth research of the combinations, particularly the number and arrangement of different layers, recurrent layers, convolutional layers and fully connected layers, can be done. Outside of network design, research to compare how deep networks interpret raw versus denoised EEG signals can also be encouraged, as this has not yet been specifically assessed. Such deep learning classification models can be applied to the applications in neuroscience, neural engineering, and even commercial applications.

5. Implementation

5.1 Dataset Cleaning

Artifacts are parts of the recorded signal that arise from sources other than the source of interest (i.e., neuronal activity in the brain).EEG data is inherently noisy because EEG electrodes also pick up unwanted electrical physiological signals, such as the electromyogram (EMG) from eye blinks and muscles on the neck. There are also concerns about the motion artifacts occurring from cable movement and electrode displacement when the subject moves. As such, artifacts are a form of interference or noise relative to the signal of interest. There are many possible causes of such interference which are explained as follows:

5.1.1 Causes of Interference

- **Environmental artifacts**
    - Persistent oscillations centered around the AC power line frequency (typically 50 or 60 Hz)
    - Brief signal jumps due to building vibration (such as a door slamming)
    - Electromagnetic field noise from nearby elevators, cell phones, the geomagnetic field, etc.

- **Instrumentation artifacts**
    - Electromagnetic interference from stimulus presentation (such as EEG sensors picking up the field generated by unshielded headphones)
    - Continuous oscillations at specific frequencies used by head position indicator (HPI) coils
    - Random high-amplitude fluctuations (or alternatively, constant zero signal) in a single channel due to sensor malfunction (e.g., in surface electrodes, poor scalp contact)

- **Biological artifacts**
    - Periodic QRS-like signal patterns (especially in magnetometer channels) due to electrical activity of the heart
    - Short step-like deflections (especially in frontal EEG channels) due to eye movements
    - Large transient deflections (especially in frontal EEG channels) due to blinking
    - Brief bursts of high frequency fluctuations across several channels due to the muscular activity during swallowing

There are also some cases where signals from within the brain can be considered artifactual. For example, if a researcher is primarily interested in the sensory response to a stimulus, but the experimental paradigm involves a behavioral response (such as button press), the neural activity associated with the planning and executing the button press could be considered an artifact relative to signal of interest (i.e., the evoked sensory response).

There are 3 basic options when faced with artifacts in your recordings:

- Ignore the artifact and carry on with analysis
- Exclude the corrupted portion of the data and analyze the remaining data
- Repair the artifact by suppressing artifactual part of the recording while leaving the signal of interest intact

It is indeed easy to visually identify abrupt outliers, for example when signals are lost or when intense EMG artifacts are present. However, it is difficult to identify persistent noisy channels or

noise that is sparsely presented in multi-channel recordings. More importantly, manual data processing is highly subjective, rendering it difficult for other researchers to reproduce the procedures. The most frequent artifact removal algorithms used are independent component analysis (ICA) and frequency domain filters to limit the bandwidth of the EEG to be analyzed which is useful when there is a certain frequency range of interest so that the rest can be safely discarded.

5.1.2 Methods for Artifact Removal

- **Low Pass Filter**

  The term cutoff frequency conjures up the image of an all-or-nothing effect at the frequency named. For example, a class may have a particular cutoff grade for passing or failing one point below the cutoff grade, and the student does not go on. An amusement park may have a particular height cutoff to go on certain Rides all individuals below that height are excluded from the ride. The behavior of low-frequency filters in terms of their cutoff frequencies is not at all so absolute as the behavior of classroom teachers or amusement park officials. In fact, it may be surprising to learn how little a filter affects activity at its cutoff frequency. When a low-frequency filter encounters a sine wave that happens to be exactly at its cutoff frequency, it cuts down the amplitude of that wave by approximately 30%. Sine waves at frequencies somewhat below that frequency are reduced by somewhat more than 30% the farther the wave's frequency is below the filter's nominal frequency, the more it is attenuated. Perhaps more surprising, sine waves at frequencies somewhat above the cutoff frequency are also reduced in size by the filter, although by somewhat less than 30%. Again, the more the sine wave's frequency exceeds the low-frequency filter's nominal frequency, the less it is affected by the filter.

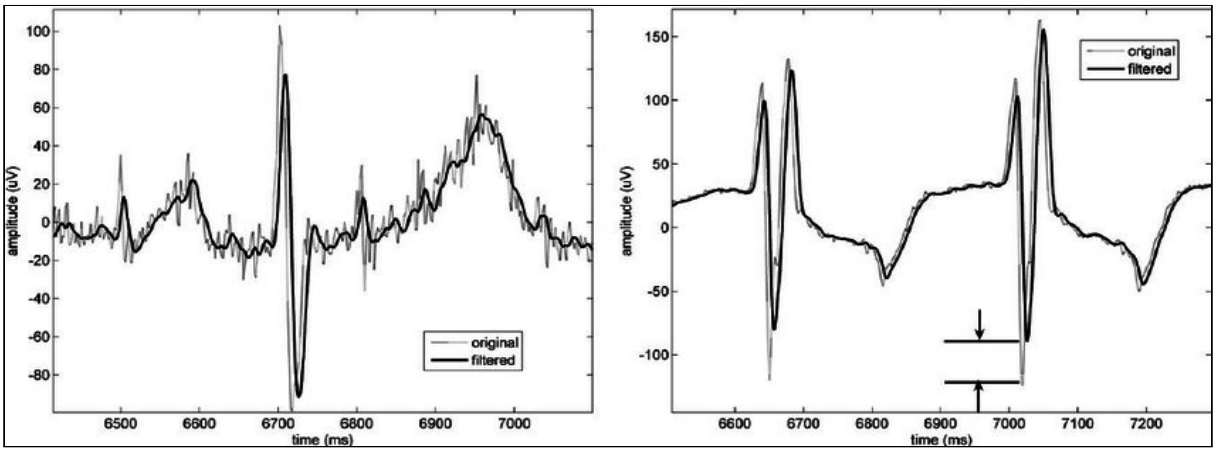

Fig. 10. Before and after applying Low Pass Filter

Low-pass filter example. The benefit of the low-pass filter is attenuation of high frequency noise. The filtered signal (solid line) is smoothed compared to the original signal (dotted line). Right panel: The cost of low-pass filtering can be seen in the reduction of the high frequency component of the signal - significant reduction in S wave amplitude in this case.

- **Independent Components Analysis (ICA)**

  Independent Components Analysis (ICA) is a technique for estimating independent source signals from a set of recordings in which the source signals were mixed together in unknown ratios. A common example of this is the problem of blind source separation: with 3 musical instruments playing in the same room, and 3 microphones recording the performance (each picking up all 3 instruments, but at varying levels), can you somehow

"unmix" the signals recorded by the 3 microphones so that you end up with a separate "recording" isolating the sound of each instrument? It is not hard to see how this analogy applies to EEG/MEG analysis: there are many "microphones" (sensor channels) simultaneously recording many "instruments" (blinks, heartbeats, activity in different areas of the brain, muscular activity from jaw clenching or swallowing, etc). As long as these various source signals are statistically independent and non-gaussian, it is usually possible to separate the sources using ICA, and then re-construct the sensor signals after excluding the sources that are unwanted.

ICA finds directions in the feature space corresponding to projections with high non-Gaussianity. Not necessarily orthogonal in the original feature space, but orthogonal in the whitened feature space. In contrast, Principal Components Analysis (PCA) finds orthogonal directions in the raw feature space that correspond to directions accounting for maximum variance. or differently, if data only reflect Gaussian processes ICA and PCA are equivalent. As seen in the diagram given below, PCA fails at recovering our "instruments" since the related signals reflect non-Gaussian processes.

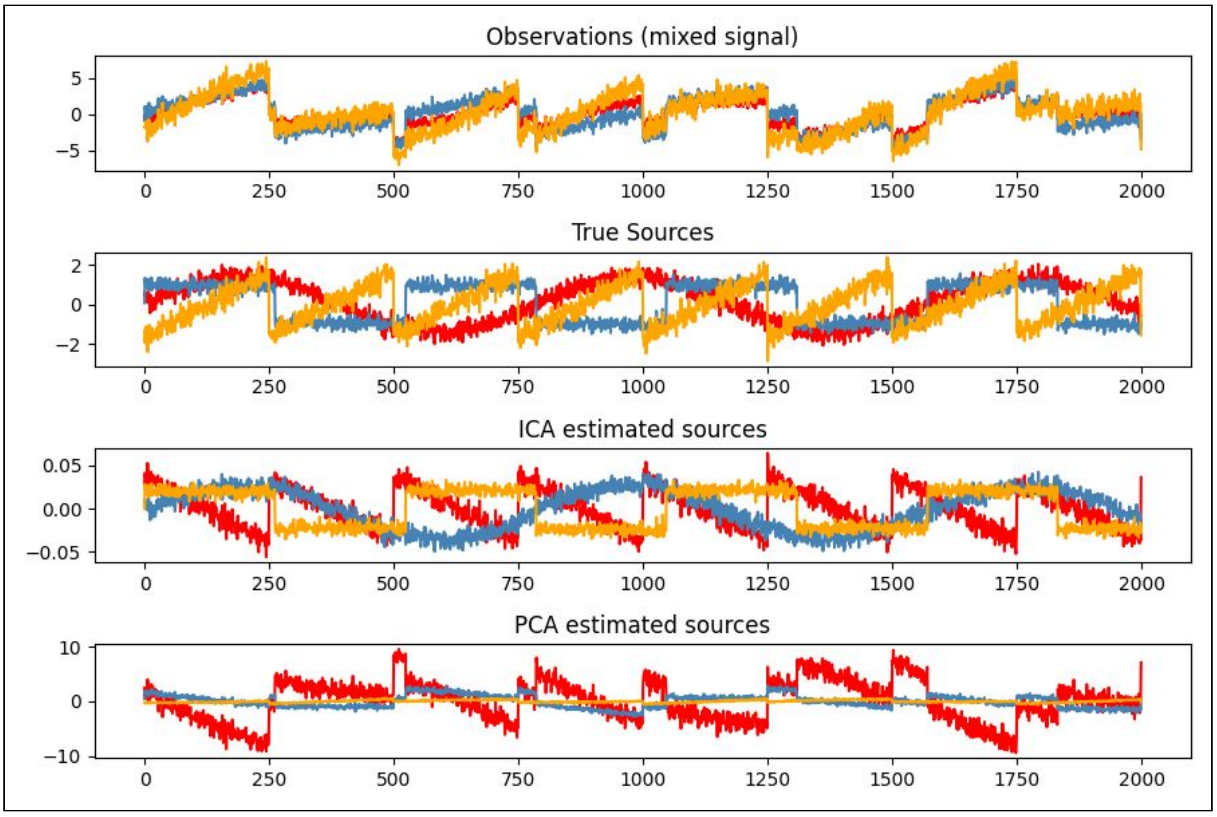

Fig. 11. ICA and PCA comparison

5.2 Dataset Preprocessing

Digital image processing is the use of computer algorithms to perform image processing on digital images.The acquired image data is usually messy and comes from different sources. To feed them to the ML model (or neural network), it needs to be standardized and cleaned up. More often than not, preprocessing is used to conduct steps that reduce the complexity and increase the accuracy of the applied algorithm. We can't write a unique algorithm for each of the conditions in which an image is taken, thus, when we acquire an image, we tend to convert it into a form that allows a general algorithm to solve it. The dataset preprocessing pipeline on the Physionet Sleep dataset involves the following steps:

5.2.1 Loading the Physionet Sleep Dataset

1. We first download the dataset for 20 subjects, each file for a subject is in edf format and is then converted into a numpy array object and combined for all 20 subjects.

2. The data for each subject consists of a time series data collected from 7 channels out of which there are 2 eeg channels from Fpz-Cz and Pz-Oz electrode locations, for single channel eeg data we select Fpz-Cz channel.

3. The recording on an average for each subject is for 22 hours and the sampling frequency of the data is 100Hz meaning at every second 100 readings are taken.

4. For each subject the total number of data points becomes around 7920000 data points (22 x 60 x 60 x 100).

5. The window for classification that we've selected is of 30 seconds so we divide the data for each subjects into samples of 30 seconds so the average number of sample for each subject becomes 2640 and for 20 subjects we have 54587 sample where each sample consist of data points for 30 seconds i.e. 3000 data points (30 x 100).

6. Thus the final dimension of the input dataset is 54587 x 3000.

5.2.2 Preprocessing for 2D CNN

1. From the input received in the previous stage our aim is to generate an image of the spectrogram and feed it into our network.

2. We take each sample which is nothing but a time series data of 30 seconds sampled at 100 Hz and use that to create a spectrogram for that sample.

3. Digitally sampled data, in the time domain, is broken up into chunks, which usually overlap, and Fourier transforms to calculate the magnitude of the frequency spectrum for each chunk. Each chunk then corresponds to a vertical line in the image; a measurement of magnitude versus frequency for a specific moment in time (the midpoint of the chunk). These spectrums or time plots are then "laid side by side" to form the image or a three-dimensional surface. An image of a spectrogram generated from one of the samples is shown below.

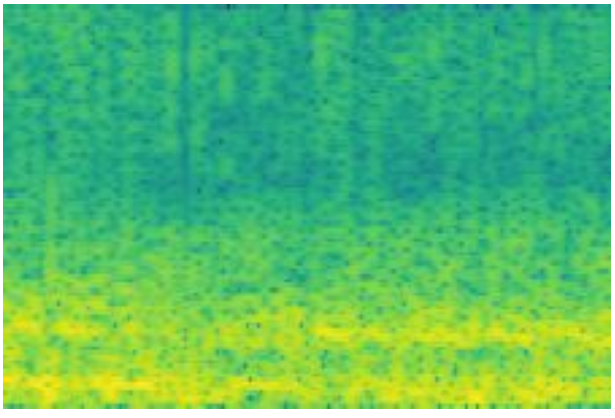

Fig. 12. Sample Spectrogram

4. The dimensions of the spectrogram generated is 36 x 54 x 3 where 36 is the height of the image and 54 is its width, 3 indicates the number of channels in the rgb image generated.

5. Image normalization is an important step which ensures that each input parameter (i.e, pixel in this case) has a similar data distribution. This makes convergence faster while training the neural network.

6. So we first convert each image from uint8 to float32 format. And then we normalize the pixel intensities of each image by dividing it by 255.0 so that it lies between the range [0.0, 1.0].

7. We One Hot Encode the Y labels to form a tensor of shape (batch_size, 5).

This preprocessed data is then fed as input into the 2D CNN in mini-batches of 16.

5.2.3 Preprocessing for 1D CNN

1. A 1D CNN is very effective when you expect to derive interesting features from shorter (fixed-length) segments of the overall data set. This applies well to the analysis of time sequences. It also applies to the analysis of any kind of signal data over a fixed-length period (such as EEG signals).

2. The data obtained from the loading dataset step is a time series data of 30 seconds for multiple samples which can be directly fed into our 1D CNN.

3. However, it is necessary to perform normalization for 1D CNN as well because it improves the numerical stability of the model and often reduces training time.

4. The normalization for each data in a sample is performed in the following way:

$$X^i = (X - X_{min}) / (X_{max} - X_{min})$$

where X^i is the normalised value of the data point, X is the actual value of that data point, X_{min} is the minimum value of a datapoint and X_{max} is the maximum value of a datapoint in the selected time series.

5. Since a neural network takes input data in batches, we add a superfluous channel dimension to the training, validation and testing data points. Hence the new shape of the instances now correspond to (no_of_images, time_steps, features). Here, the features dimension has a value of 1 as the instances in the PhysioNet Sleep dataset have been taken from a single EEG channel.

6. We One Hot Encode the Y labels to form a tensor of shape (batch_size, 5).

This preprocessed data is then fed as input into the 1D CNN in mini-batches of 16.

5.3 Dataset Splitting

1. This step involves randomly dividing the available set of observations into three parts, a training set, a validation set and a testing set.

2. Training dataset is the sample of data used to fit the model. The actual dataset that we use to train the model (weights and biases in the case of Neural Network). The model sees and learns from this data.

3. Validation dataset is the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

4. Test dataset is the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets).

5. In our case 70% of the dataset is allocated to the training set, 20% to the validation set and the remaining 10% is allocated to the testing dataset. This split is done for a dataset with 54,587 samples; the final distribution is given below.

Training Set	40967
Validation Set	10890
Testing Set	2730

Table. 3. Train, Validation and Test Split

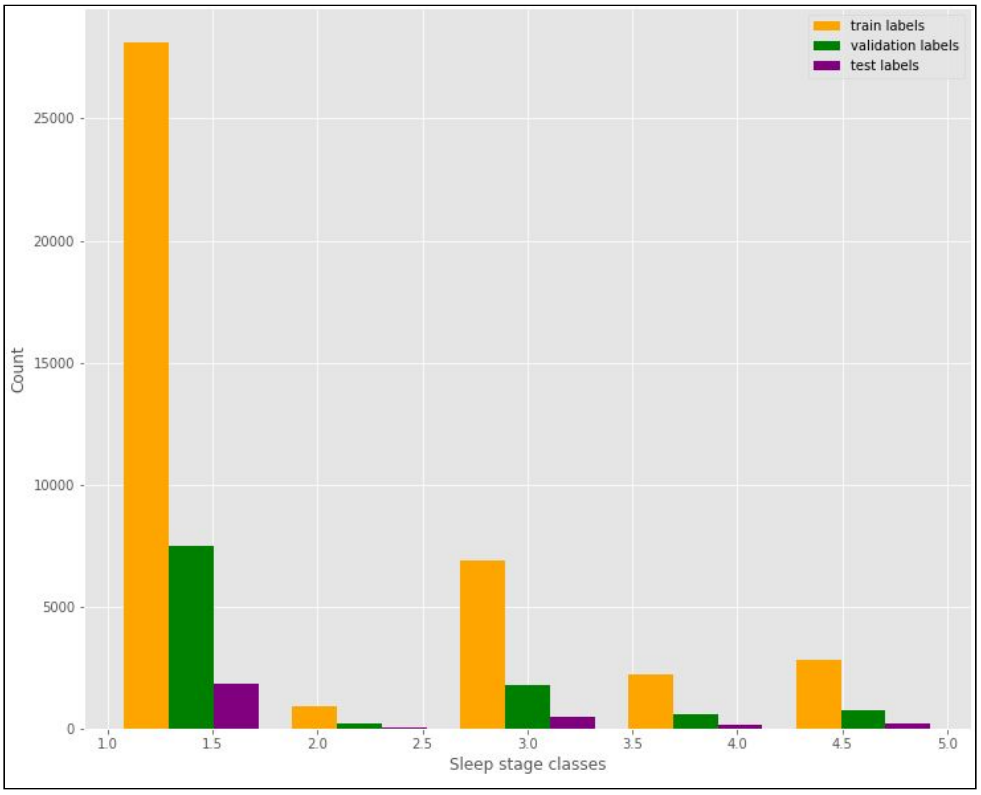

Fig. 13. Histogram Visualization for Train, Validation and Test Split

5.4 Dataset Visualization

For the dataset visualization after splitting, we use Principal Component Analysis (PCA) algorithm. PCA is an unsupervised linear dimensionality reduction technique that uses orthogonal transformations to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

Here we use the Scikit-learn PCA implementation to visualize training, validation and testing instances. The visualization is constructed using Matplotlib's Scatter Plot API. The algorithm manages to project the (54587, 3000) dimensional PhysioNet Sleep dataset onto a 2-dimensional

space using 2 Principal Components. There is a lot of overlap among classes as very few classes can be separated but most of them are mixed as shown below in Fig. 14, 15, 16.

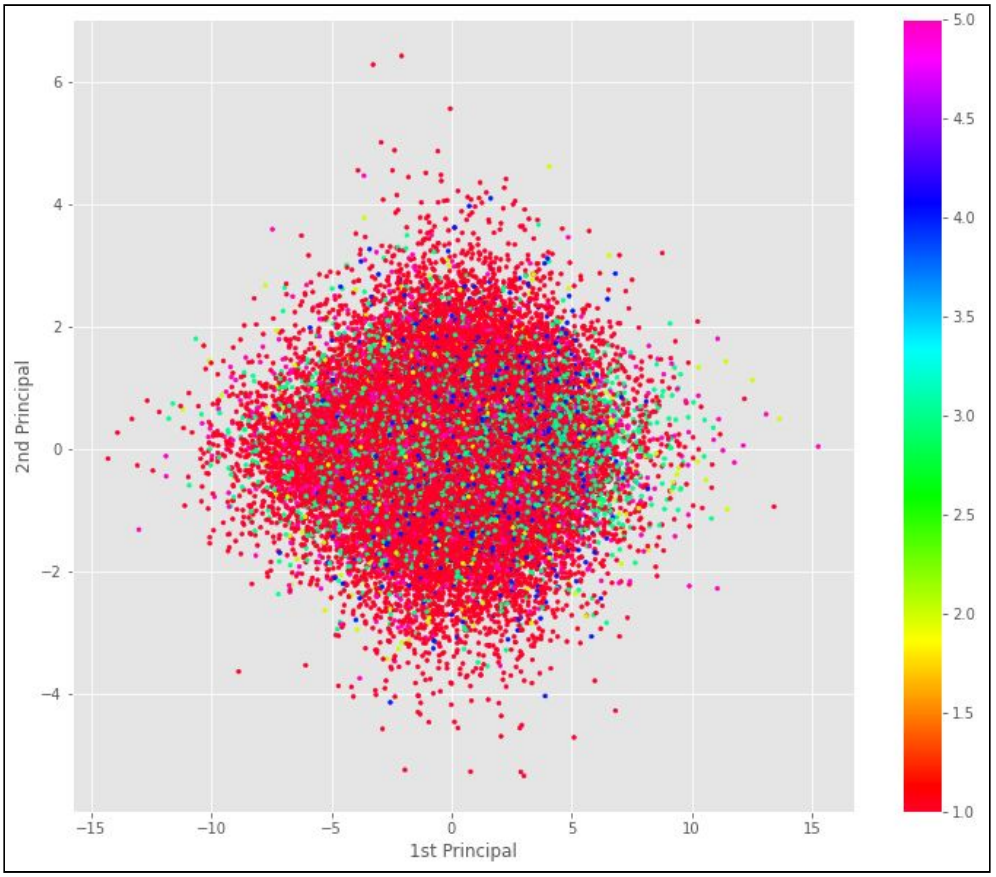

Fig. 14. PCA Visualization for Training data

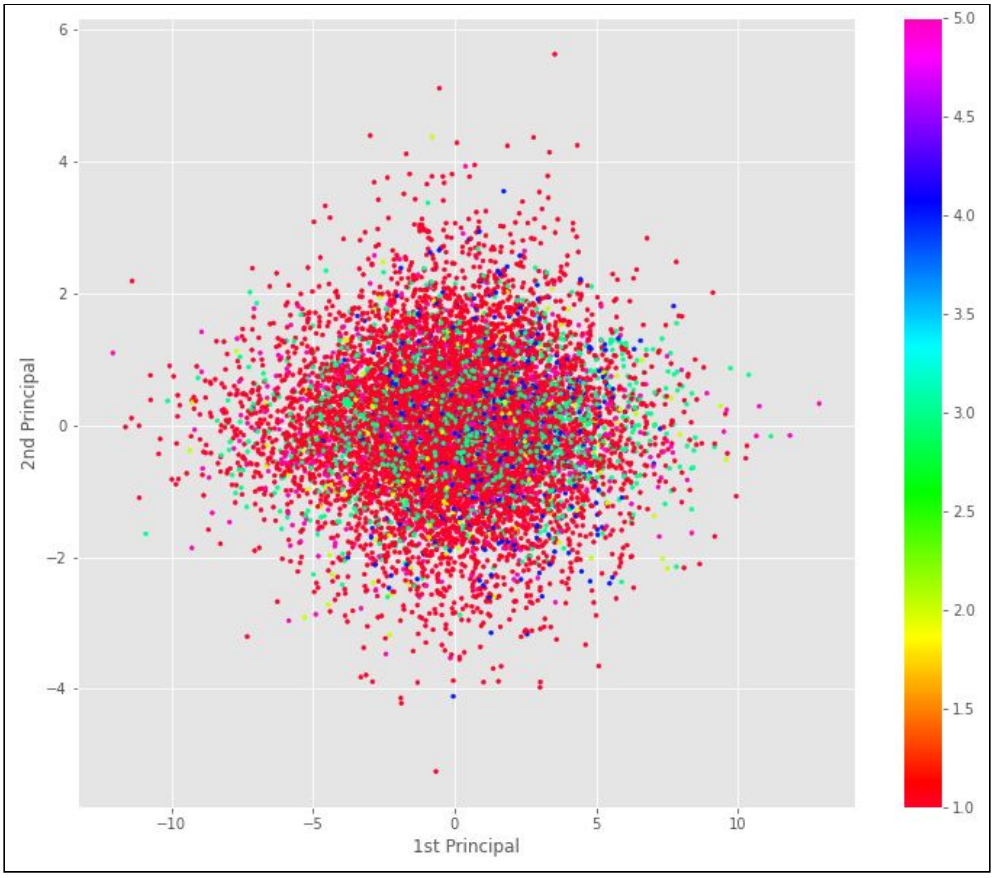

Fig. 15. PCA Visualization for Validation data

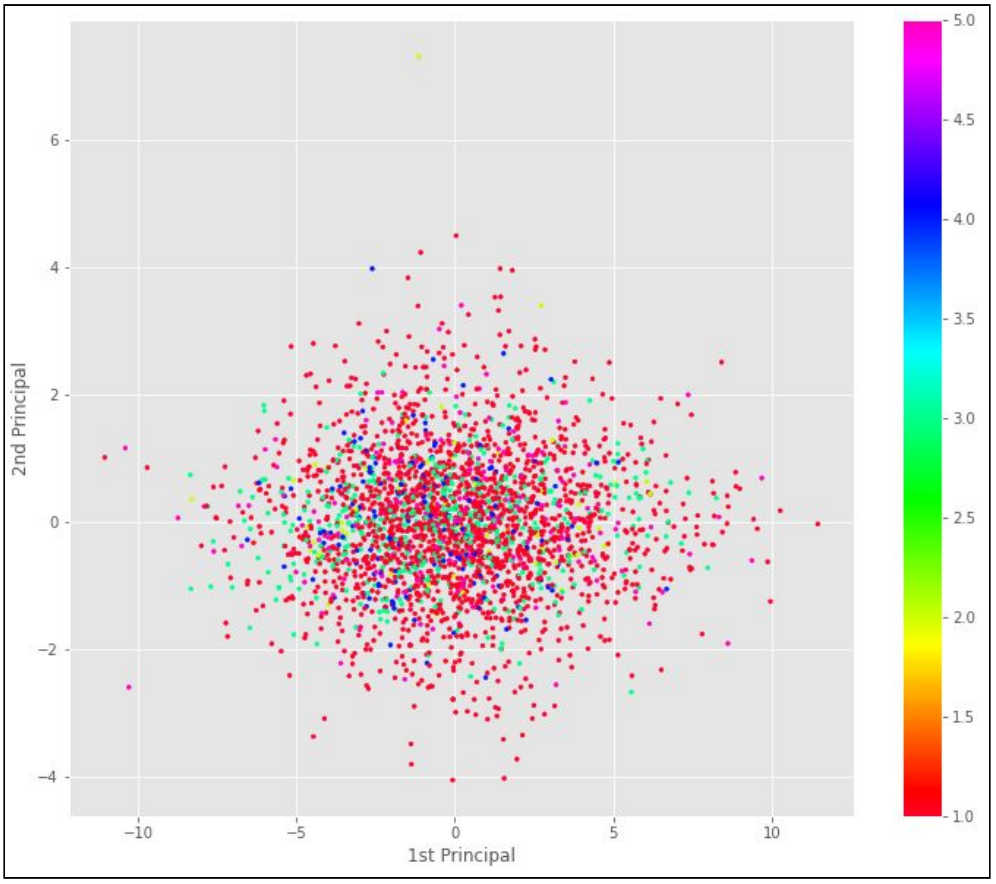

Fig. 16. PCA Visualization for Testing data

5.5 Algorithm

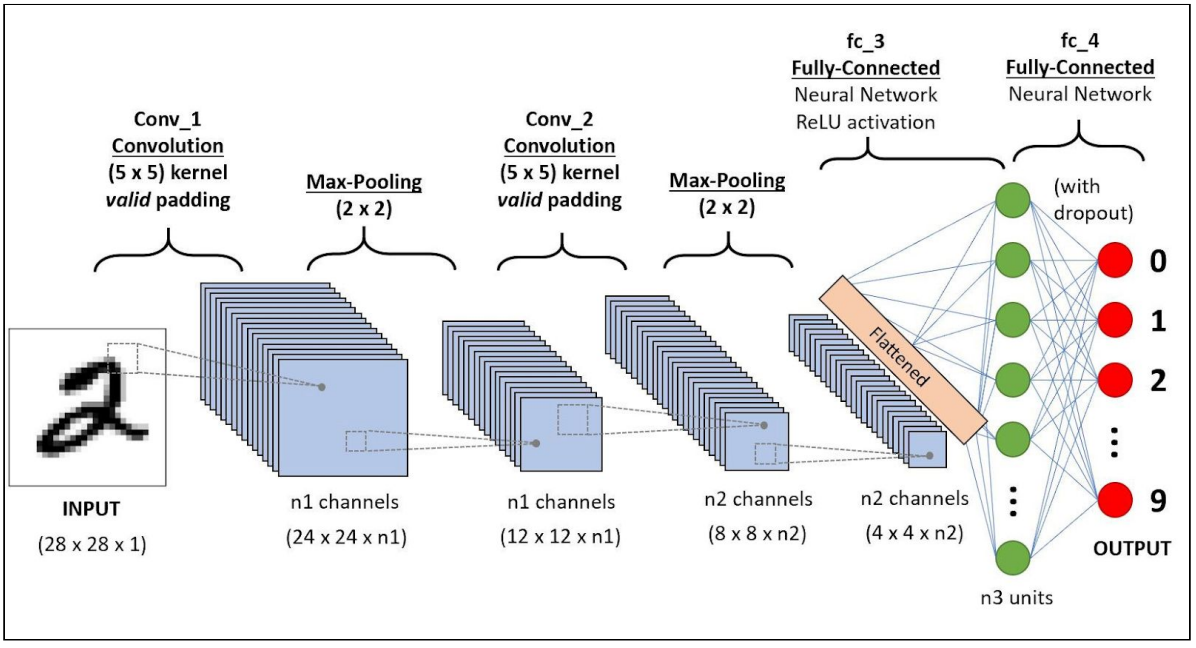

Fig. 17. Architecture of a 2D CNN

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics.

CNNs take biological inspiration from the visual cortex in the Human Brain. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. This idea was expanded upon by a fascinating experiment by Hubel and Wiesel in 1962 where they showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation. For example, some neurons fired when exposed to vertical edges and some when shown horizontal or diagonal edges. Hubel and Wiesel found out that all of these neurons were organized in a columnar architecture and that together, they were able to produce visual perception. This idea of specialized components inside of a system having specific tasks (the neuronal cells in the visual cortex looking for specific characteristics) is one that machines use as well, and is the basis behind CNNs.

5.5.1 Justification for using CNNs

The main motivation behind the emergence of Convolutional Neural Networks (CNNs) in Deep Learning scenarios has been to address many of the limitations that traditional neural networks faced when applied to those problems. Some of which are explained below:

- In the past, Traditional Multilayer Perceptron (MLP) models have been used for image recognition. However, due to the full connectivity between nodes, they suffered from the curse of dimensionality, and did not scale well with higher resolution images. A 1000×1000 pixel image with RGB color channels has 3 million weights, which is too high to feasibly process efficiently at scale with full connectivity.

- Also, such network architecture does not take into account the spatial structure of data, treating input pixels which are far apart in the same way as pixels that are close together. This ignores locality of reference in image data, both computationally and semantically. Thus, full connectivity of neurons is wasteful for purposes such as image recognition that are dominated by spatially local input patterns.

CNNs mitigate the challenges posed by the MLP architecture by exploiting the strong spatially local correlation present in natural images. As opposed to MLPs, CNNs have the following distinguishing features:

- **3D volumes of neurons:** The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth where each neuron inside a convolutional layer is connected to only a small region of the layer before it, called a receptive field. Distinct types of layers, both locally and completely connected, are stacked to form a CNN architecture.

- **Local connectivity:** CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. The architecture thus ensures that the learned "filters" produce the strongest response to a spatially local input pattern. Stacking many such layers leads to non-linear filters that become increasingly global (i.e. responsive to a larger region of pixel space) so that the network first creates representations of small parts of the input, then from them assembles representations of larger areas.

- **Parameter Sharing:** In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. This means that all the neurons in a given convolutional layer respond to the same feature within their specific response field. Replicating units in this way allows for the resulting feature map to be equivariant under changes in the locations of input features in the visual field, i.e. they grant translational equivariance. Weight sharing dramatically

reduces the number of free parameters learned, thus lowering the memory requirements for running the network and allowing the training of larger, more powerful networks.

- **Pooling:** In a CNN's pooling layers, feature maps are divided into rectangular sub-regions, and the features in each rectangle are independently down-sampled to a single value, commonly by taking their average or maximum value. In addition to reducing the sizes of feature maps, the pooling operation grants a degree of translational invariance to the features contained therein, allowing the CNN to be more robust to variations in their positions.

- **Sparse Representations:** A CNN is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The “filters” in CNNs tend to be drastically smaller than the input which simplifies the number of computations required to train the model or to make predictions. This results in reducing the computational operations by a huge factor and leads to efficient training.

5.5.2 CNN Architecture

Technically, Deep Learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between [0, 1].

- **Convolution Layer**
  The main building block of the encoder is the Convolutional Layer. The primary purpose of Convolution is to extract features from the input image. It preserves the spatial relationship between pixels by learning image features using small squares of input data. The type of convolutions we are interested in are 1-dimensional and 2-dimensional discrete convolutions, which act like a weighted sliding sum over an area of pixels. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

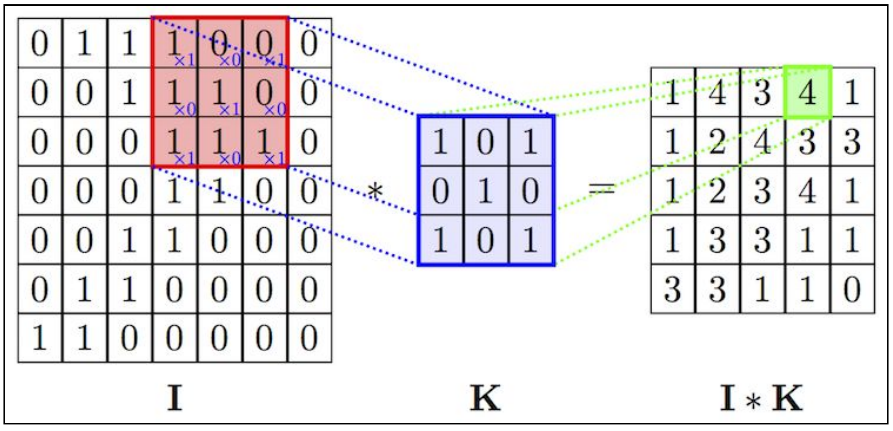

Fig. 18. 2D Convolution operation

For instance, a 3x3 matrix called a kernel/filter (K in Fig. 18.) slides across the pixels in an image. At each point, it calculates the weighted sum of the kernels’ values and every pixel in the 3x3 chunk of the image. The sum is then put in the first value of the output image called the feature map (I*K in Fig. 18.). The kernel then slides over one pixel and repeats the process for every pixel in the image.The red area shown above in Fig. 18. where the convolution operation takes place is called the receptive field.

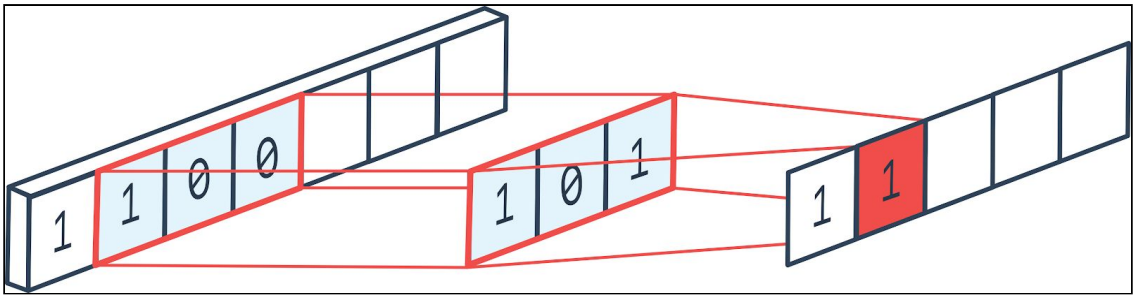

Fig. 19. 1D Convolution operation

1D Convolution operation is similar to 2D Convolution operation except that instead of using 2D images and 2D kernels, we use 1D EEG wave vectors and 1D kernels as shown above in Fig. 19.

- **Activation Layer with ReLU**

  One of the most popular activation functions in Deep Learning models, the Rectified Linear Unit (ReLU) function, promises to deliver state-of-the-art performance with stellar results. Compared to other AFs like the sigmoid and tanh functions, the ReLU function offers much better performance and generalization in deep learning. The function is a nearly linear function that retains the properties of linear models, which makes them easy to optimize with gradient-descent methods. ReLU's purpose is to introduce non-linearity in our CNN since the real world data would want our CNN to learn would be non-negative linear values. The ReLU function performs a threshold operation on each input element where all values less than zero are set to zero. The function and its derivative both are monotonic. ReLU is represented as:

$$f(x) = max(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases}$$

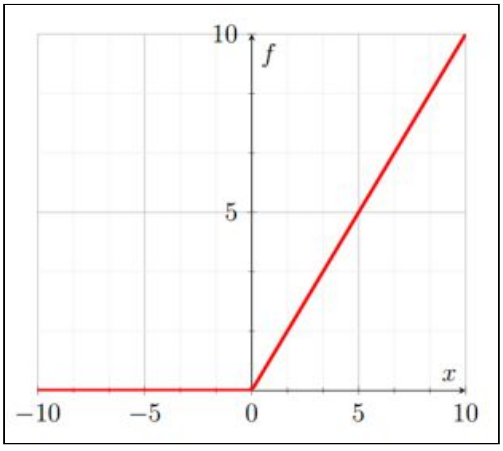

Fig. 20. ReLU activation function

- **Pooling Layer**

  Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:
  - Max Pooling
  - Average Pooling
  - Sum Pooling

  Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

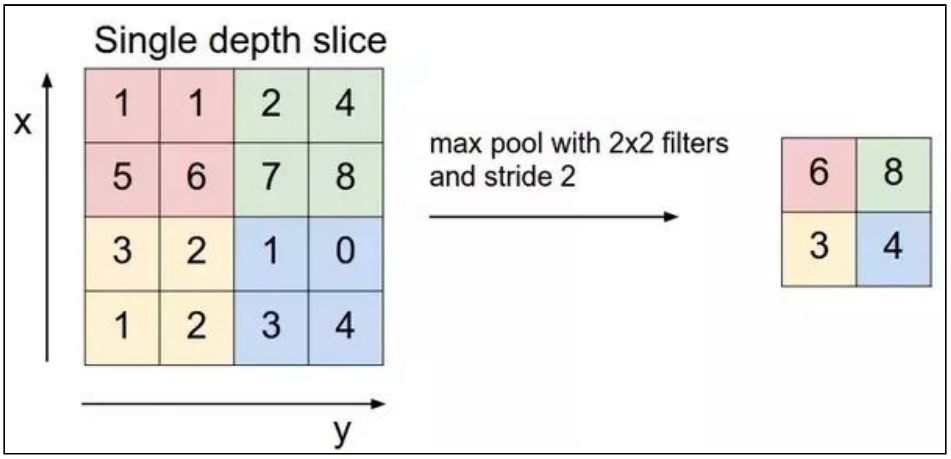

Fig. 21. 2D Max Pooling

1D Max Pooling operation is similar to 2D Max Pooling operation except that instead of using 2D images, we use 1D EEG wave vectors as shown above in Fig. 22.

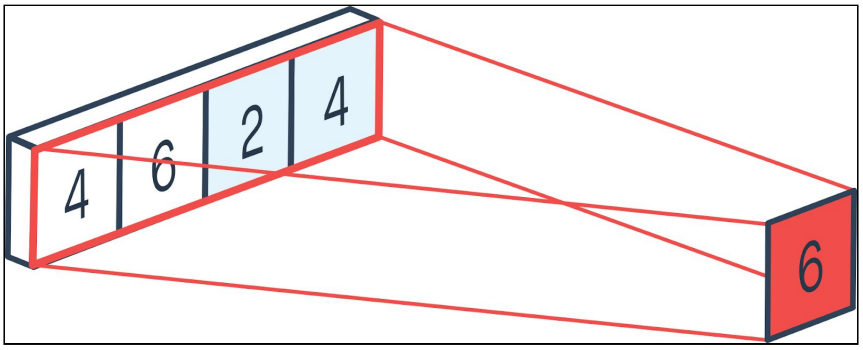

Fig. 22. 1D Max Pooling

- **Fully Connected Layer**
  The layer we call the FC layer, we flatten our matrix into vectors and feed it into a fully connected layer like a neural network.

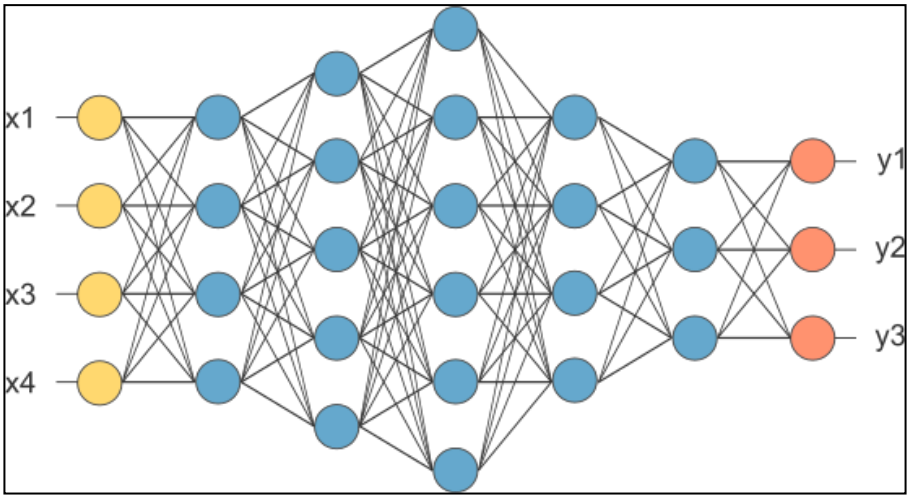

Fig. 23. Pooling Layer flattened as Fully Connected Layer

In the above diagram, the feature map matrix will be converted as a vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as W, N1, N2, N3 or R.

5.5.3 Proposed 2D CNN Architecture

The proposed 2D CNN is modelled after VGG16 architecture. It takes an input tensor of the shape (batch_size, height, width, no_of_channels) and outputs a tensor of shape (batch_size, 5). Here the input images are the RGB spectrograms of shape (36, 54, 3) of each datapoint in the dataset built during the preprocessing phase. The summary of the model is mentioned below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 34, 52, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 34, 52, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 17, 26, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 17, 26, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 17, 26, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 13, 128)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 13, 256)        295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 13, 256)        590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 13, 256)        590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 6, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 6, 512)         1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 6, 512)         2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 4, 6, 512)         2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 3072)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              12587008
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 20485
=================================================================
Total params: 37,004,741
Trainable params: 37,004,741
Non-trainable params: 0
_________________________________________________________________
```

5.5.4 Proposed 1D CNN Architecture

The proposed 1D CNN takes an input tensor of the shape (batch_size, time_steps, features) and outputs a tensor of shape (batch_size, 5). Here the input is the EEG wave vector of shape (batch_size, 3000, 1) of each datapoint in the dataset built during the preprocessing phase. The summary of the model is mentioned below.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 3000, 1)]	0
conv1d (Conv1D)	(None, 3000, 16)	64
leaky_re_lu (LeakyReLU)	(None, 3000, 16)	0
batch_normalization (BatchNo	(None, 3000, 16)	64
max_pooling1d (MaxPooling1D)	(None, 1500, 16)	0
conv1d_1 (Conv1D)	(None, 1500, 32)	1568
leaky_re_lu_1 (LeakyReLU)	(None, 1500, 32)	0
batch_normalization_1 (Batch	(None, 1500, 32)	128
max_pooling1d_1 (MaxPooling1	(None, 750, 32)	0
conv1d_2 (Conv1D)	(None, 750, 64)	6208
leaky_re_lu_2 (LeakyReLU)	(None, 750, 64)	0
batch_normalization_2 (Batch	(None, 750, 64)	256
max_pooling1d_2 (MaxPooling1	(None, 375, 64)	0
conv1d_3 (Conv1D)	(None, 375, 128)	24704
leaky_re_lu_3 (LeakyReLU)	(None, 375, 128)	0
batch_normalization_3 (Batch	(None, 375, 128)	512
max_pooling1d_3 (MaxPooling1	(None, 187, 128)	0
flatten (Flatten)	(None, 23936)	0
dropout (Dropout)	(None, 23936)	0
dense (Dense)	(None, 512)	12255744
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
batch_normalization_4 (Batch	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0

```
dense_1 (Dense)              (None, 5)                 2565
=================================================================
Total params: 12,293,861
Trainable params: 12,292,357
Non-trainable params: 1,504
_________________________________________________________________
```

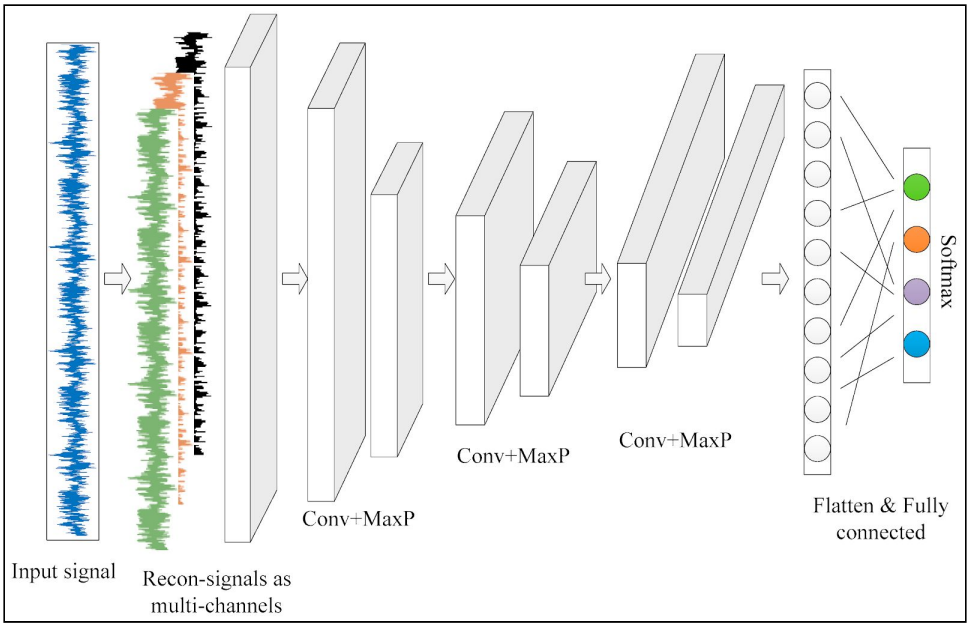

Fig. 24. Architecture of the proposed 1D CNN

5.5.5 Learning Methodology and Parameter Tuning for Model Training

- The aforementioned neural network models are trained using Mini-batch Gradient Descent Algorithm. It is a first-order optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our models. Parameters refer to weights **w** and biases **b** in our neural network.

- Optimization refers to the task of minimizing/maximizing an objective function **f(x)** parameterized by **x**. In machine learning terminology, it is the task of minimizing the cost/loss/objective function **J(w)** parameterized by the model's parameters **w, b ∈ R^d**.

- Consider the 3-dimensional graph below (Fig. 25.) in the context of a cost function. Our goal is to move from the mountain in the top right corner (high cost) to the dark blue sea in the bottom left (low cost). The arrows represent the direction of steepest descent (negative gradient) from any given point–the direction that decreases the cost function as quickly as possible.

- Starting at the top of the mountain, we take our first step downhill in the direction specified by the negative gradient. Next we recalculate the negative gradient (passing in the coordinates of our new point) and take another step in the direction it specifies. We continue this process iteratively until we get to the bottom of our graph, or to a point where we can no longer move downhill–a local minimum.

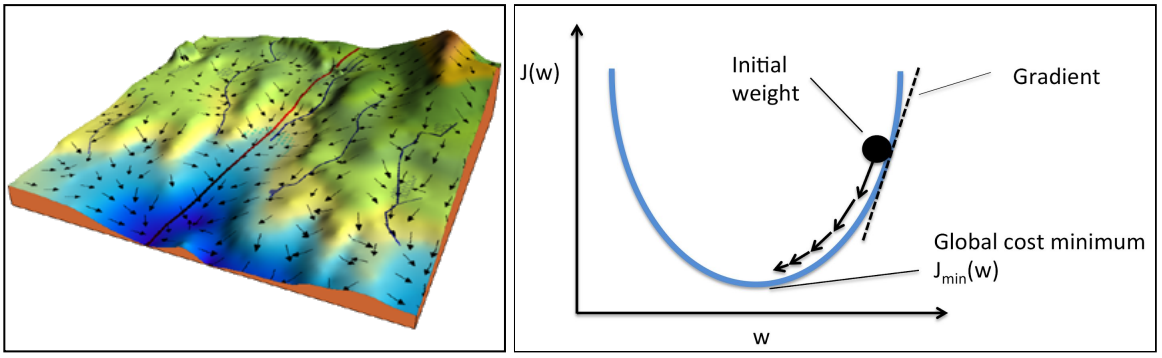

Fig. 25. Gradient Descent algorithm objective

- **Learning Rate:** The size of these steps is called the learning rate. With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.

- **Loss Function:** It tells us “how good” our model is at making predictions for a given set of parameters. The cost function has its own curve and its own gradients. The slope of this curve tells us how to update our parameters to make the model more accurate. Here we use Categorical Crossentropy Error as our loss function.

$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i} y_i \log(\hat{y}_i)$$

Fig. 26. Catgorical Crossentropy Error

- **Algorithm:**
  Repeat until convergence {

$$\begin{aligned}\frac{\partial}{\partial\Theta}J_\Theta &= \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i)-y]^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i)-y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y) \\ &= \frac{1}{m}(h_\Theta(x_i)-y)x_i\end{aligned}$$

Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i)-y)x_i]$$

  }

The weights and biases are updated using the **Backpropagation Algorithm.** The use of Mini-batch Gradient Descent in the neural network setting is motivated by the high cost of running back propagation over the full training set. It can overcome this cost and lead to fast convergence. Generally each parameter update in Mini-batch Gradient Descent is computed w.r.t a few training examples or a mini-batch as opposed to a single example. The reason for this is twofold: first this

reduces the variance in the parameter update and can lead to more stable convergence, second this allows the computation to take advantage of highly optimized matrix operations that should be used in a well vectorized computation of the cost and gradient. Hence we use a **mini-batch size of 16 (spectrogram images for 2D CNN and EEG vectors for 1D CNN) for 15 epochs.**

We use the **Adam Optimizer with an initial learning rate (alpha) of 0.0001** along with Mini-batch Gradient Descent. Adaptive Moment Estimation (Adam) is an adaptive learning rate method that computes adaptive learning rates for each parameter. Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with Momentum.

6. Results Analysis

6.1 Results for 2D CNN

Accuracy and Loss Metrics		
	Loss	**Accuracy (%)**
Training	0.2122	92.13
Validation	0.2292	92.10
Test	0.2477	91.94

Classification Report				
	Precision	**Recall**	**f1 - score**	**Support**
Class W	0.97	1.00	0.98	1819
Class 1	0.60	0.29	0.39	65
Class 2	0.88	0.91	0.89	490
Class 3/4	0.85	0.75	0.80	157
Class R	0.72	0.73	0.72	199
Micro avg	0.92	0.92	0.92	2730
Macro avg	0.80	0.74	0.76	2730
Weighted avg	0.91	0.92	0.91	2730
Samples avg	0.92	0.92	0.92	2730

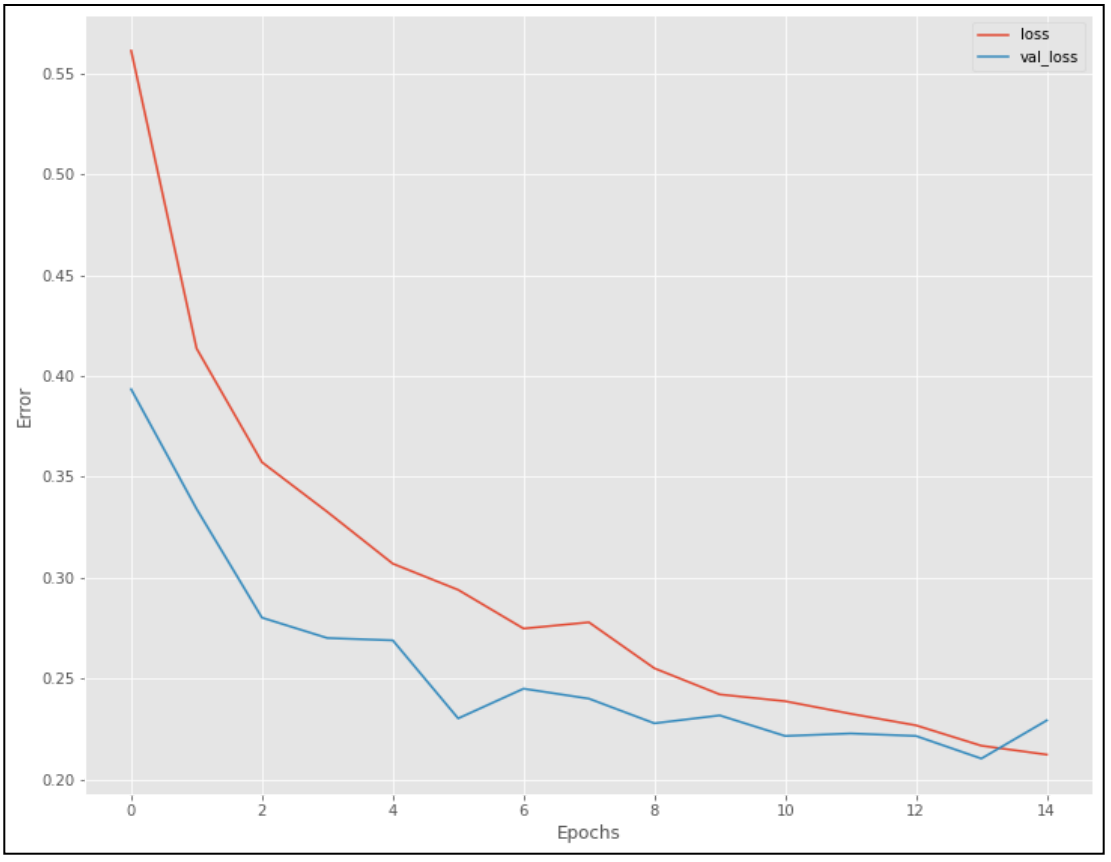

Fig. 27. Crossentropy Error vs. Epochs

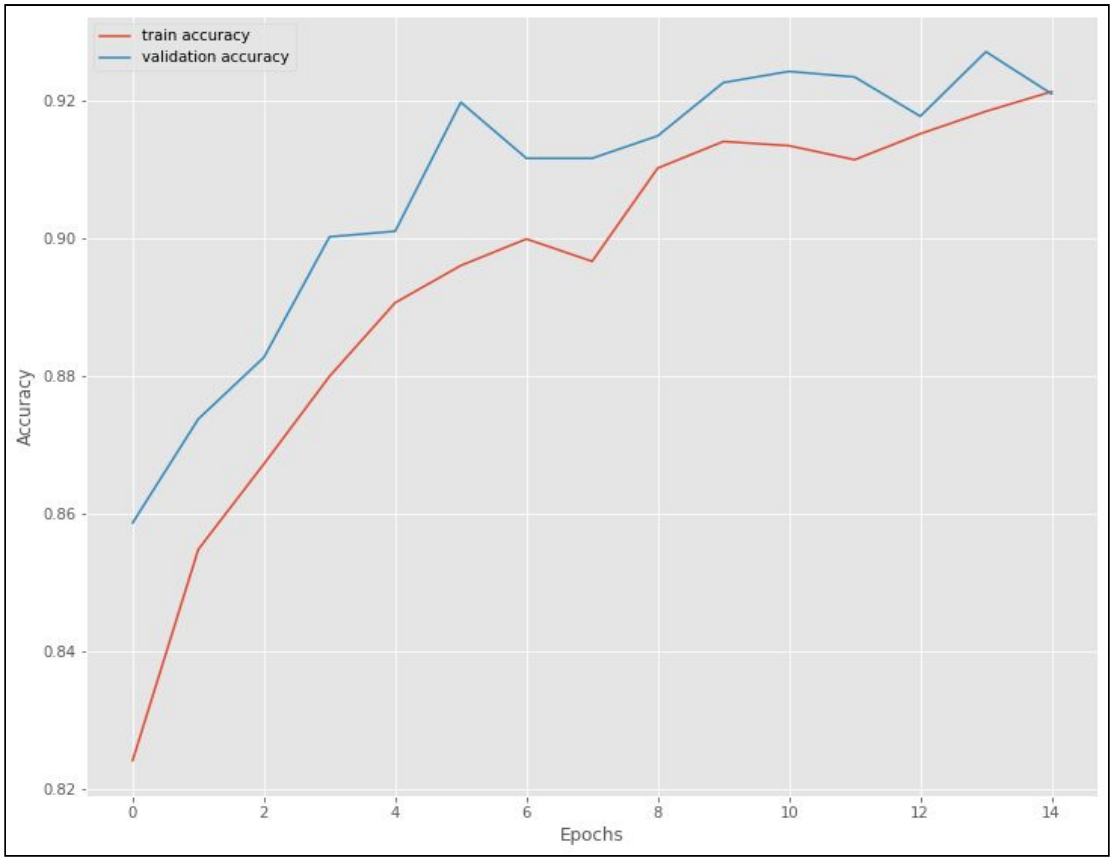

Fig. 28. Accuracy vs. Epochs

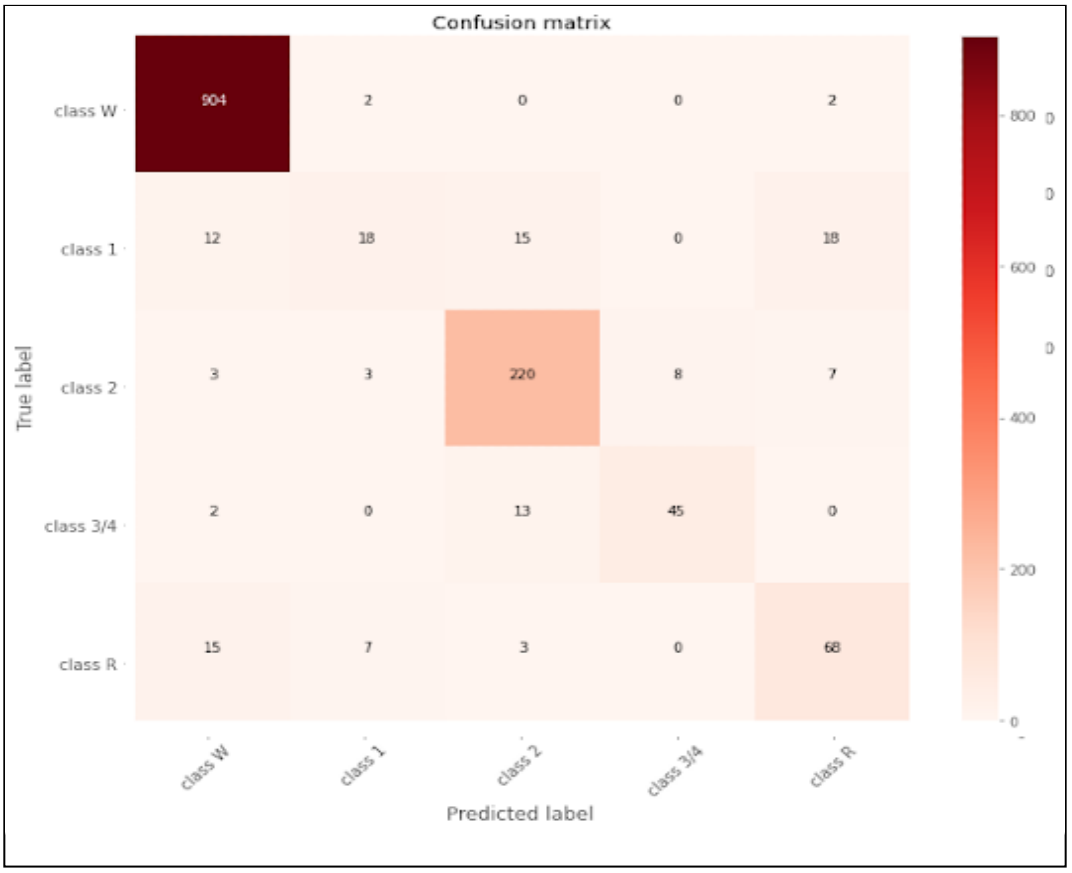

Fig. 29. Confusion Matrix

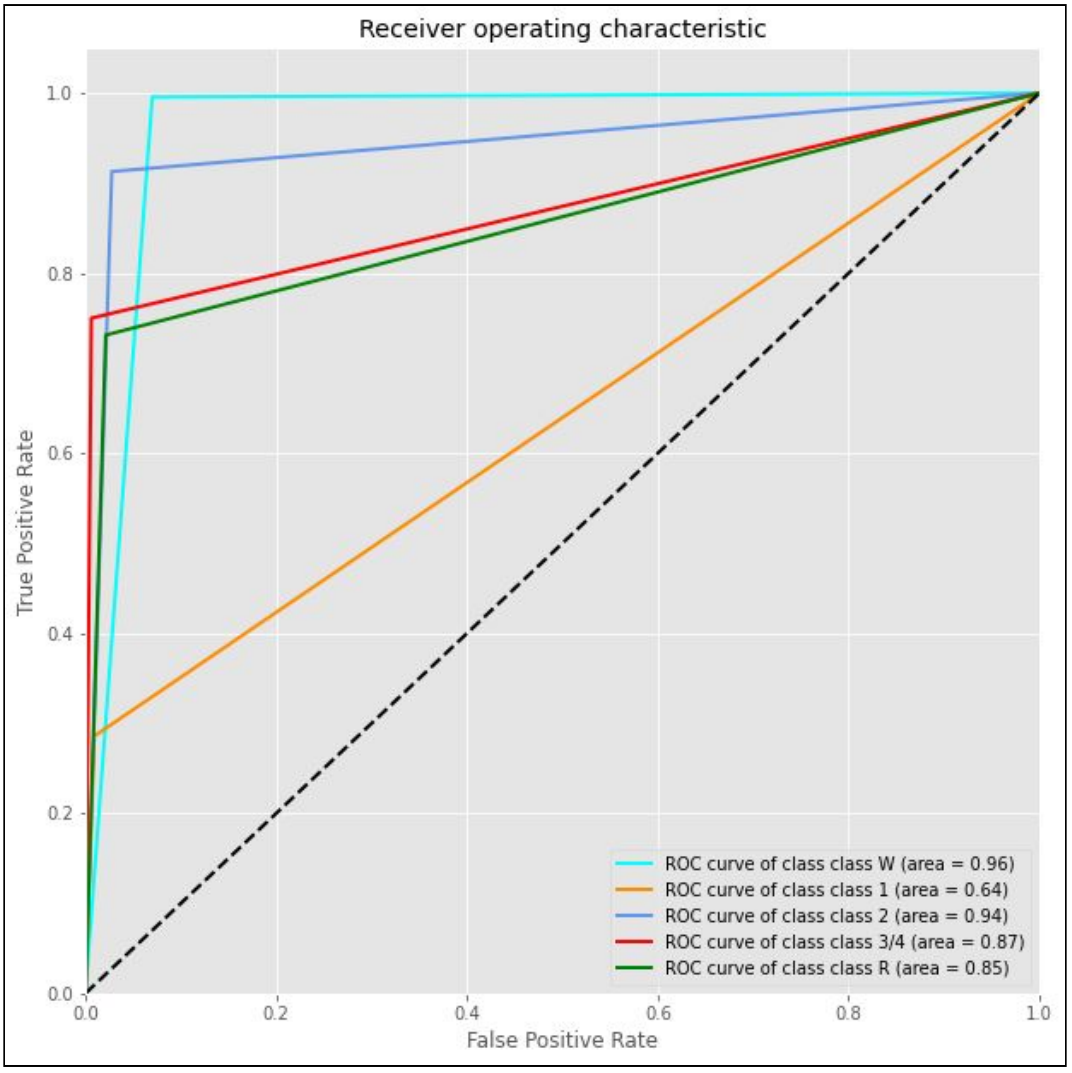

Fig. 30. ROC

6.2 Results for 1D CNN

Accuracy and Loss Metrics		
	Loss	**Accuracy (%)**
Training	0.7795	92.91
Validation	0.7567	92.94
Test	0.7563	92.38

Classification Report				
	Precision	**Recall**	**f1 - score**	**Support**
Class W	0.98	0.98	0.98	1819
Class 1	0.44	0.28	0.34	65
Class 2	0.85	0.86	0.86	490
Class 3/4	0.82	0.88	0.85	157
Class R	0.74	0.77	0.76	199
Micro avg	0.92	0.92	0.92	2730
Macro avg	0.77	0.76	0.76	2730
Weighted avg	0.92	0.92	0.92	2730

Samples avg	0.92	0.92	0.92	2730

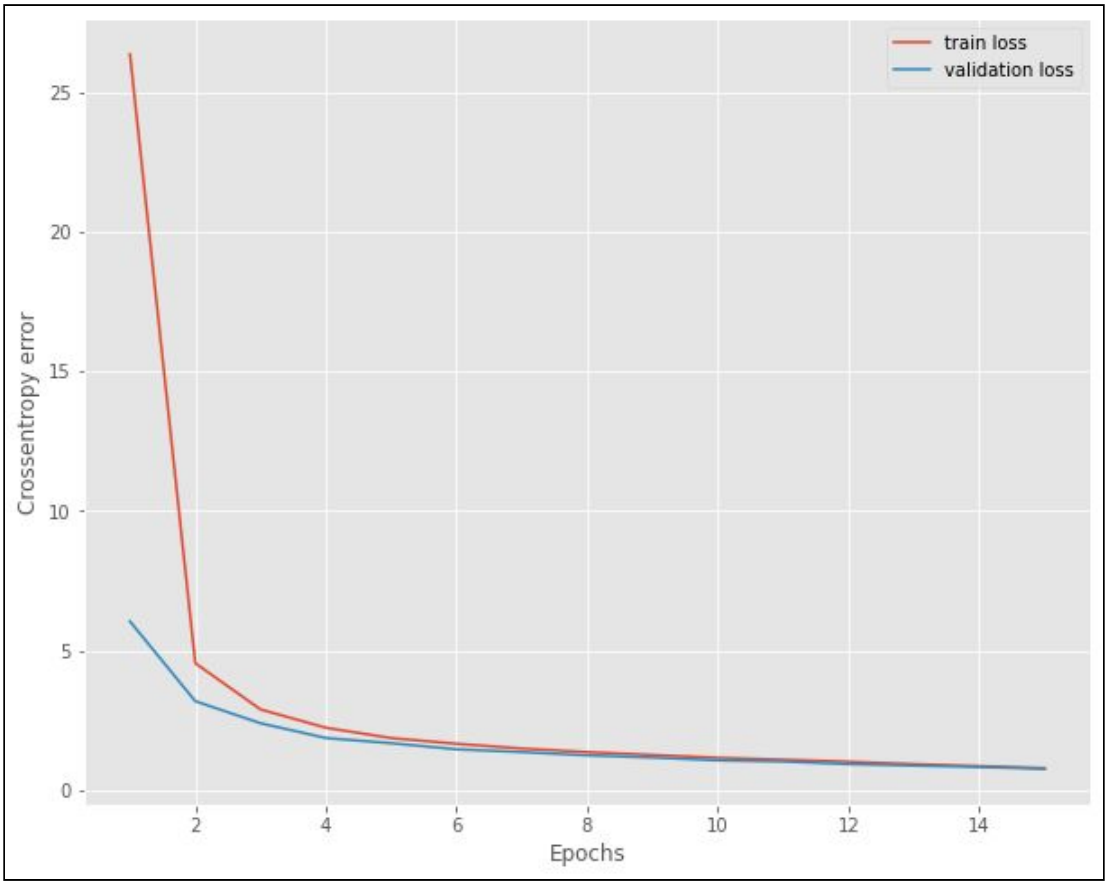

Fig. 31. Crossentropy Error vs. Epochs

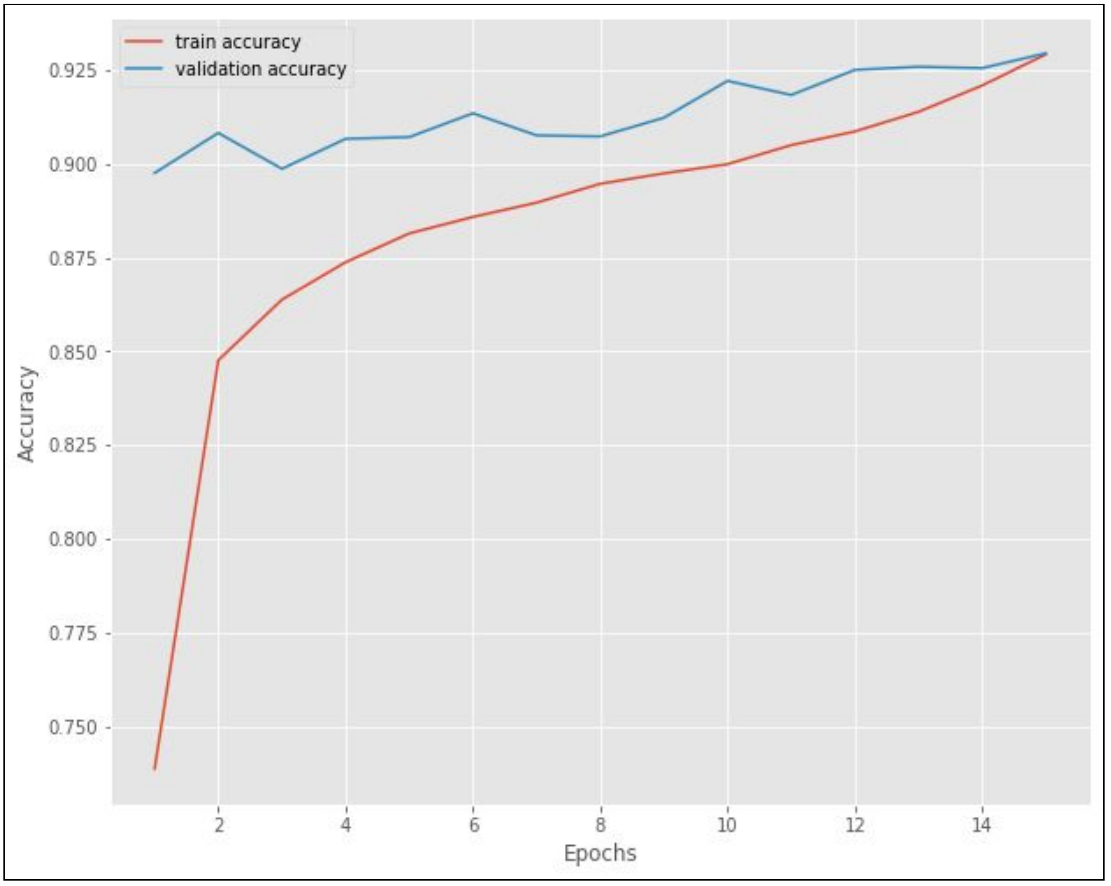

Fig. 32. Accuracy vs. Epochs

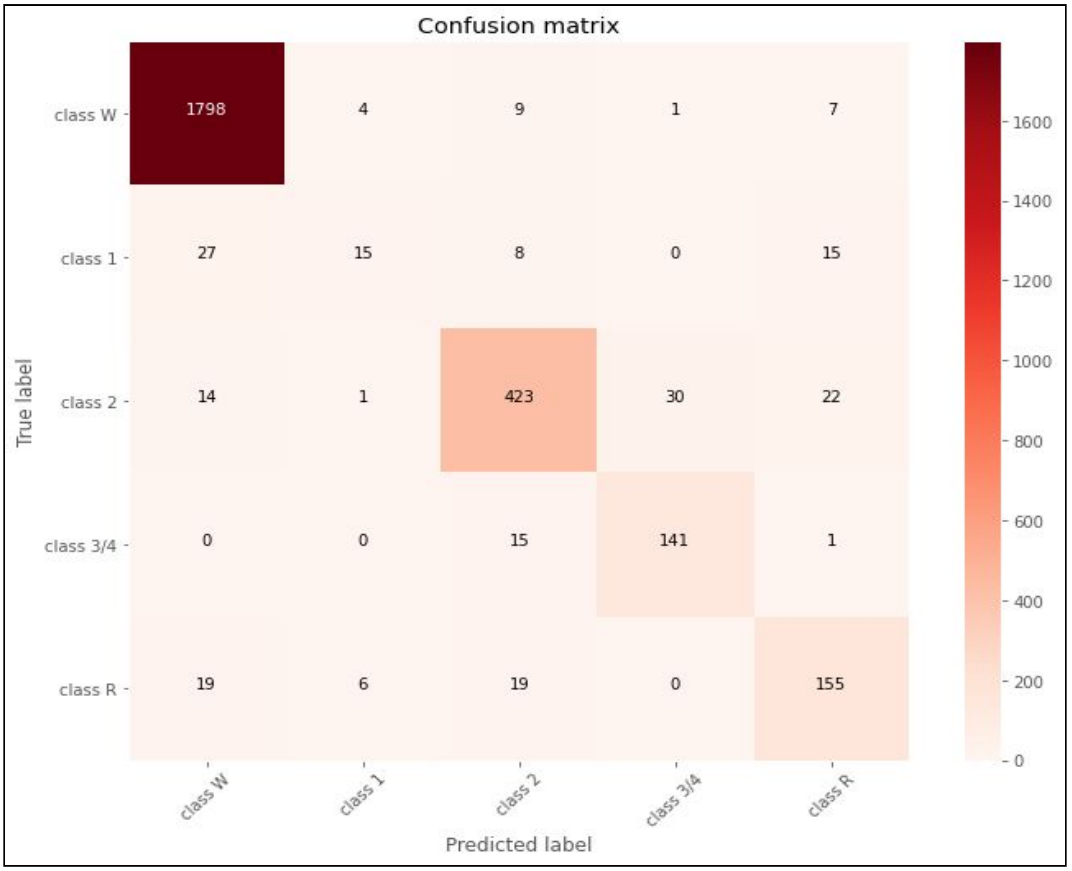

Fig. 33. Confusion Matrix

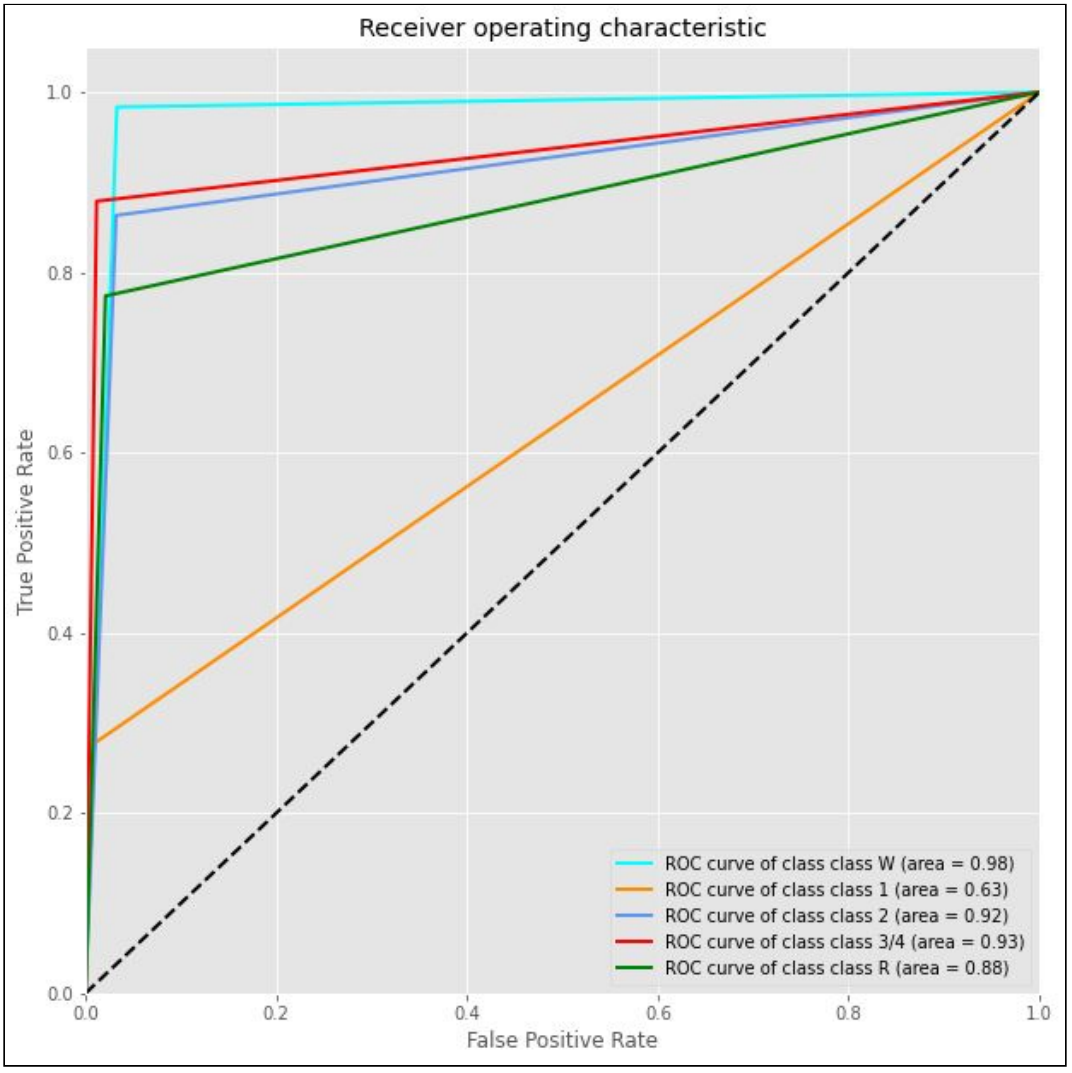

Fig. 34. ROC

6.3 Results Comparison

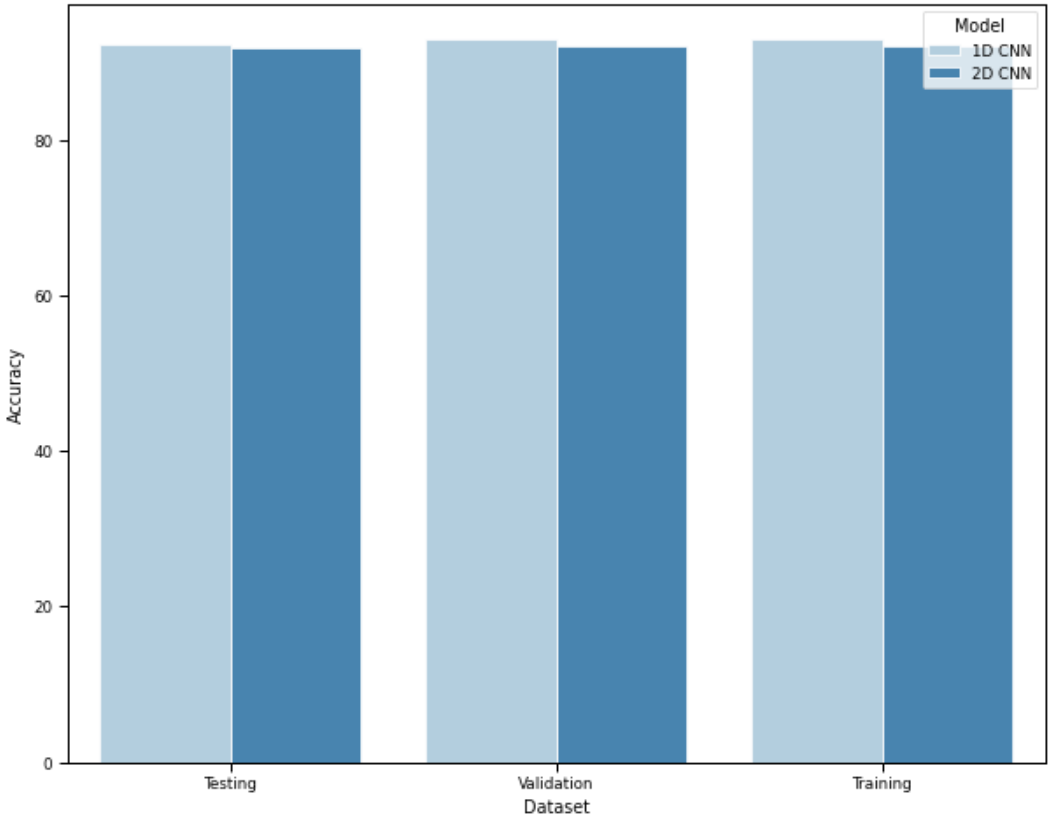

Fig. 35. Results comparison for 2D CNN and 1D CNN

As evident from Fig. 35., both the models are performing extremely well and have generalized well on the test dataset. However 1D CNN seems to be performing better than 2D CNN in all the test cases.

7. Conclusion and Future Scope

7.1 Conclusion

Our study proves that Deep Learning architectures can also be effectively used for the task of classification of brain waves. The two architectures that were used, namely, 2D CNN and 1D CNN, showed promising results with 1D CNN being a bit more accurate than 2D CNN. The reason behind this slight difference might be due to all the pre-processing involved in input formulation of 2D CNN. 1D CNN does not involve much pre-processing as it directly works on the raw data (time series data). But 2D CNN can be useful in some cases as the generation of spectrograms preserves all three features of the waves: time, frequency, and power. Apart from this, the overall accuracy could have been improved if the distribution of data points amongst the class labels would be even. Thus, many such deep learning architectures can be used for the task of classification of brain waves and use of more extensive and balanced datasets can lead to improved accuracy.

7.2 Future Scope

The concept of applying machine learning algorithms and deep learning techniques to brain waves for classification or prediction can be used in neural engineering and commercial applications as well. Two applications where this technique can prove to be of utmost help are:

- **Autism Detection**

  Autism Spectrum Disorder (ASD) is a syndrome that adversely affects a child where the behavioral symptoms start to appear during the first year of life. This early childhood onset includes symptoms such as lack in social interaction and very slow language skills development as stated by researchers. A continuous character and behavioral assessment are conducted by specialists to detect autistic presence in a child. A documented analysis done by pediatrics stated that an autistic child at approximately 24 months, is still unable to produce two meaningful words that do not involve imitating and repeating. Despite so much research being conducted, the exact factors to why this disorder occurs remain unanswered. As of why this atypical behavior is very difficult to detect is maybe due to the barely noticeable changes of the primary neural impairment itself.

  The relation of ASD with EEG signal is that there is a significant decline of EEG complexity perceived in an autistic child. The noteworthy differences were observed between brain regions in the right hemisphere and the central cortex represented by EEG signals. The current diagnostic method for ASD is time consuming. By applying deep learning architectures to evaluate the patterns of EEG signals of autistic and normal kids, a model can be created for the early detection of autism. Publicly available datasets of EEG signals of an autistic patient are limited. Perhaps, working towards building a more extensive dataset can help in the creation of a much accurate model for detecting autism.

- **Brain Password**

  Biometrics is the measuring and statistical analysis of people's physical and behavioral attributes. This technology can be used to define an individual's unique identity, often employed for security purposes. The traditional biometric traits are face recognition, retina or iris scanning, fingerprints, hand geometry, palm print signature, keystroke entry pattern, and voice recognition. However, many of these traits can be forged or stolen. Fingertips, for example, can be damaged by an injury or can be forged by a gummy finger. Disguises can

trick face recognition applications. Brain waves or Electroencephalogram (EEG) are the electrical activity of an individual's brain that is unique and cannot be tampered with. Hence, EEG is proposed as an alternative or an additional way of securing biometric applications.

EEG signals are gathered from electrodes that are placed in several locations on the scalp. Because everyone's brain is structured differently each EEG signal is unique for each person. EGG uniqueness makes the biometric un-forgeable or un-duplicable. The current EEG-based biometric procedures are based on eye blinks, visual stimuli, and rest states. Data can be collected via hardware devices and after preprocessing which can involve spectrogram computation, feature extraction, feature selection, etc., the pre-processed data can be trained on classifiers such as SVM or ANN for pattern recognition.

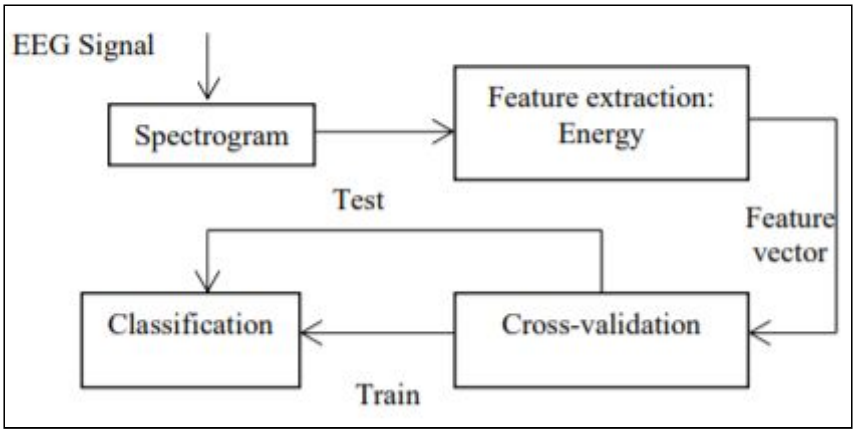

Fig. 36. Flowchart of the Methodology

8. References

Serkan Kiranyaz , Onur Avci , Osama Abdeljaber , Turker Ince , Moncef Gabbouj , Daniel J. Inman. 1D Convolutional Neural Networks and Applications – A Survey. arXiv:1905.03554 [eess.SP]

Morteza Zabihi , Ali Bahrami Rad , Serkan Kiranyaz , Simo Särkkä , Moncef Gabbouj. 1D Convolutional Neural Network Models for Sleep Arousal Detection.

Shaojie Bai, J. Zico Kolter, Vladlen Koltun.(2018, April). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271 [cs.LG]

He, Yangdong & Zhao, Jiabao. (2019). Temporal Convolutional Networks for Anomaly Detection in Time Series. Journal of Physics: Conference Series. 1213. 042050. 10.1088/1742-6596/1213/4/042050.

Dilated Convolutions and Kronecker Factored Convolutions, May 2016. [Online]. Available: https://www.inference.vc/dilated-convolutions-and-kronecker-factorisation/

EEG (Electroencephalography): The Complete Pocket Guide, August 2019. [Online]. Available: https://imotions.com/blog/eeg/

EEG Headset Prices – An Overview of 15+ EEG Devices, july 2019. [Online]. Available: https://imotions.com/blog/eeg-headset-prices/

Yuan L., Cao J. (2018) Patients' EEG Data Analysis via Spectrogram Image with a Convolution Neural Network. In: Czarnowski I., Howlett R., Jain L. (eds) Intelligent Decision Technologies 2017. IDT 2017. Smart Innovation, Systems and Technologies, vol 72. Springer, Cham

QUAN LIU , JIFA CAI , SHOU-ZEN FAN , MAYSAM F. ABBOD , JIANN-SHING SHIEH , YUCHEN KUNG , AND LONGSONG LIN. (May 2019). Spectrum Analysis of EEG Signals Using CNN to Model Patient's Consciousness Level Based on Anesthesiologists' Experience.

A Deep Learning Model for Automated Sleep Stages Classification Using PSG Signals
Ozal Yildirim, Ulas Baran Baloglu, U Rajendra Acharya
Int J Environ Res Public Health. 2019 Feb; 16(4): 599. Published online 2019 Feb 19. doi: 10.3390/ijerph16040599

B Kemp, AH Zwinderman, B Tuk, HAC Kamphuisen, JJL Oberyé. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave micro continuity of the EEG. IEEE-BME 47(9):1185-1194 (2000).

Deep learning for electroencephalogram (EEG) classification tasks: a review. Alexander Craik *et al* 2019 *J. Neural Eng.* 16 031001

Koudelková, Zuzana & Strmiska, Martin. (2018). Introduction to the identification of brain waves based on their frequency. MATEC Web of Conferences. 210. 05012. 10.1051/matecconf/201821005012.

Jacob M. Williams. (2017). Deep Learning and Transfer Learning in the Classification of EEG Signals.

Ali, Nur. (2020). Autism spectrum disorder classification on electroencephalogram signal using deep learning algorithm. IAES International Journal of Artificial Intelligence (IJ-AI). 9. 91. 10.11591/ijai.v9.i1.pp91-99.

O. Nieves and V. Manian, "Automatic person authentication using fewer channel EEG motor imagery," 2016 World Automation Congress (WAC), Rio Grande, 2016, pp. 1-6, doi: 10.1109/WAC.2016.7582945.