ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

CLASSIFICATION OF THE MOTOR EEG SIGNALS BY USING DEEP NEURAL NETWORKS

M.Sc. THESIS

Leyla ABILZADE

Department of Applied Informatics
Applied Informatics Programme

DECEMBER 2019

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

CLASSIFICATION OF THE MOTOR EEG SIGNALS BY USING DEEP NEURAL NETWORKS

M.Sc. THESIS

Leyla ABILZADE
(708181011)

Department of Applied Informatics

Applied Informatics Programme

Thesis Advisor: Prof. Dr. Tamer ÖLMEZ

DECEMBER 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

DERİN SİNİR AĞLARI KULLANARAK MOTOR EEG SİNYALLERİNİN SINIFLANDIRILMASI

YÜKSEK LİSANS TEZİ

Leyla ABILZADE
(708181011)

Bilişim Uygulamaları Anabilim Dalı

Bilişim Uygulamaları Programı

Tez Danışmanı: Prof. Dr. Tamer ÖLMEZ

ARALIK 2019

Leyla Abilzade, a M.Sc. student of ITU Informatics Institute student ID 708181011, successfully defended the thesis entitled “CLASSIFICATION OF THE MOTOR EEG SIGNALS BY USING DEEP NEURAL NETWORKS”, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Tamer ÖLMEZ** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Ertuğrul KARAÇUHA** İstanbul Technical University	.....
	Doç. Dr. Gökhan BİLGİN Yıldız Technical University	.....

Date of Submission : 15 November 2019
Date of Defense : 9 December 2019

To my mother,

FOREWORD

I want to thanks to my advisor Prof.Dr. Tamer ÖLMEZ for all his efforts in encouraging and helping me in this study. Also, I am very grateful to Prof. Zumray DOKUR and Phd. Nuri KORHAN for assisting and supporting me in accomplishing my thesis.

December 2019 Leyla ABILZADE

TABLE OF CONTENTS

Page

ABBREVIATIONS

BCI : Brain Computer Interface
BMI : Brain Machine Interface
CNN : Convolutional Neural Network
CSP : Common Spatial Patterns
CSSP : Common Saptio Spectral Patterns
DL : Deep Learning
DNN : Deep Neural Network
EEG : Electroencephalogram
ERD : Event Related Desynchronization
ERP : Event Related Potentials
ERS : Event Related Synchronization
FCL : Fully Connected Layer
FCNN : Fully Connected Neural Network
fMRI : Functional magnetic resonance imaging
FW1 : Framework 1
FW2 : Framework 2
MEG : Magnetoencephalography
MI : Motor Imagery
ReLU : Rectified Linear Unit
SAE : Stacked Autoencoder
VAE : Variational Autoencoder

LIST OF THE SYMBOLS

$\boldsymbol{Y}$	**:** Spatially filtered amplitude values
$\boldsymbol{W}$	**:** Spatial filters
$\boldsymbol{X}$	**:** Matrix of amplitude values
$\boldsymbol{R_i}$	**:** Normalized covariance matrix
$\boldsymbol{\lambda}$	**:** Diagonal matrix
$\boldsymbol{S}$	**:** Whitening transformation

LIST OF TABLES

Page

LIST OF FIGURES

Page

CLASSIFICATION OF THE MOTOR EEG SIGNALS BY USING DEEP NEURAL NETWORKS

SUMMARY

The brain-computer interface (BCI), establishing relationship between brain and devices have become increasingly important recent times. Both being able to communicate with disabled people and playing video games without effort increases the value of the subject. However, the classification achievements of BCI that obtained by conventional methods have not yet reached the desired levels. In this respect, it forces to search for new methods that will improve the classification performance. It is observed that using deep learning techniques has achieved the desirable higher performances. In this thesis, the deep neural networks (DNN) were used to classify the EEG (electroencephalogram) signals to increase classification output.

In this study the proposed network models were applied on dataset of the BCI Competition 2008. The database contains inputs obtained from nine subjects. There are five EEG recordings for each subject collected at different times by three electrodes C3,C4 and Cz. During training feedback was used for three records. This informs the subject how accurately he/she performed the task, but in this study, the feedback records have not been used. To describe the experiment, each subject is asked to imagine moving his or her right or left hand in accordance with the direction of the arrow shown on the screen and EEG was recorded on each experiment. Therefore, the EEG features contain information only belonging to two classes.

In this study, the effects of channel number on classification performance were investigated. In this context, two methods will be compared: (i) EEG data from 3 channels are filtered by a filter with 5 different frequency bands. In this case, the number of channels increases to 15. EEG data from 15 channels are parsed using the Common Spatial Pattern (CSP) method and the signals at the CSP output are delivered to the deep neural network for classification. (ii) EEG data from 3 channels are directly transmitted to the deep neural network for classification. In general, it is desirable that the number of channels be large in order to classify the EEG data more accurately. However, this makes the system even more complicated. With the new channels that are increased in the study, both classification performance is increased and system complexity is reduced.

In this study, the effects of channel number on classification performance were investigated. In this context, two methods will be compared: (i) EEG data from 3 channels are directly employed by deep neural (DNN) for classification. (ii) EEG data from 3 channels are filtered by five bandpass filters. In this case, the number of channels increase to 15. EEG data from 15 channels are extracted to the features using the CSP method and the signals at the output of the CSP are then delivered to the deep neural network for classification. In general, it is desirable that the number of channels be large in order to classify the EEG data more accurately. However, this makes the

system even more complicated. But in this study the created extra channels can both reduce complexity and increase classification performance.

In this study, generalization of DNN is also examined. BCI systems are often designed depending on the subject. Therefore, there exist 9 different DNN for each subject. But doing so could make disadvantage for CSP to achieve its best performance, even though CSP is known best feature extractor method. This disadvantage overcame by using the DNN with CSP for the whole data together that collected from nine subjects.

In this study, EEG data is extracted in MATLAB environment by using prefiltering and CSP method. Then DNN has been used to train the extracted features on the GeForce 2080 Nvidia graphics card and coded by using the Tensorflow library in Python in the Linux environment. In total 80% of the EEG data was used for training set and other 20% is used for the test set. In this thesis, a new BCI method (filter + CSP + deep neural network) has been developed which gives high classification performance.

DERİN SİNİR AĞLARI KULLANARAK MOTOR EEG SİNYALLERİNİN SINIFLANDIRILMASI

ÖZET

Zaman geçtikçe, teknolojik yenilikler hayatımızın vazgeçilmezleri haline geliyor. Bu durumda, insanlar aralıksız yeni keşifler yapmayı veya diğerlerinin yaptıklarını geliştirerek diğerlerinden daha iyi sonuçlar elde etmeyi amaçlar. Bu teknolojik deneylerin türünde, eğitimde veya günlük yaşamımızda olsun, genel olarak insan yaşamını kolaylaştırması amaçlanmaktadır. BCI (Beyin bilgisayar arayüzü) da tam olarak bu sinifa dahildir.

BCI son dönemlerde araştırma konularında önemli ölçüde değer kazanmaya başlamıştır. Gerek engelli insanlarla iletişim kurulması, yaşam kalitesinin arttırılması; gerekse oyun sektöründe kullanılma potansiyelinin yüksek oluşu konunun insani ve ticari yönden değerini arttırmaktadır. Oyun ekipmanından yapay organlara kadar çok çeşitli uygulamalarda kullanılan BCI teknolojisinin başlıca amacı, insan beyni ile elektronik cihaz arasında hiçbir bir periferik sinir yolu olmayan ara iletişim kanalı oluşturmaktır.

Bağımsız ve pratik BCI türü olmasından dolayı, MI (hayali motor) çeşitli BCI en popüler metodlardan biridir. Hayali motor hareketler, gerçekte hiçbir fiziksel vücut haraketi yapmadan onun hayalini beyinde canlandırmaktır. Buna, sağ elin,sol elin veya ayaklarımızı hareket etdiriyor gibi düşünmek örnek ola bilir. MI hareketleri zamanı, beyinde sinyaller oluşmaya başlar ve bu motor sinyaller beyin bölgesinin üst-orta kısmında yerleşen motor korteksde üretiliyor ve genel olarak beş sınıfa ayrılıyor: $\alpha,\beta,\theta,\lambda,\gamma$. Her biri farklı frekans aralığını kapsıyor ve her biri farklı durumlarda yaranıyor, örneğin uyku zamanı veya ayık olduğumuz durumlar. Bu sinyalleri toplamak için çeşitli yöntemler kullanılıyor: Manyetoensefalografi (MEG), Elektroensefalogram (EEG), veya Fonksiyonel manyetik rezonans görüntüleme (fMRI) vb. Bilimsel araştırmalarda, örneğin, BCI'da en çok tercih edilen sinyal toplama yöntemi tam olarak EEG'dir. Non-invaziv olması EEG'yi tercih etmek için önemli sebeplerden biridir, çünkü bu teknik sinyal kaydı sırasında ameliyat gerektirmez. EEG başka bir deyişle- elektroansefalo, kafanın farklı bölgelerinde yerleştirilmiş elektrodlar, bu elektrodların bağlandığı amplifaktör ile insan beyninde üretilen sinyelleri kağıt üzerinde veya bilgisayarda basıyor. EEG bir çok alanda kullanıla bilinir: tıpta, deneysel araştırma labaratuarlarında ve saire. Epilepsi, beyin tümörü, hafıza bozuklukları, uyku problemleri, felç vb. birçok hastalıklar bile EEG kullanılarak tespit edilebilinir. Burada, BCIda sağlam veya hasta insanlar üzerinde farklı deneyler yapılarak tüm bedeni veya bedeninin bir kısmı felç olan insanlara bir tür yardımda bulunmak istenilir. Deneyler sırasında, EEG ile toplanılan beyinde üretilen sinyaller daha sonra sınıflandırılma adlandırılan prosedürde kullanılır. Mesela eğer deney sırasında sağ veya sol elimizi hareket etdirdiyimizi hayel etmeyi isterlerse, sınıflandırma yaparak toplanmış sinyallerin hangisinin sağ ve hangisinin sol el hayali hareket zamanı beyinde üretildiğini bula biliriz. Fakat bu o kadar da kolay bir işlem değildir. Sınıflandırma zamanı daha yüksek ve başarılı sonuçlar almak için herkes

farklı yöntemler kullanır. Ancak klasik yöntemlerle elde edilen sınıflama başarımları henüz istenilen düzeylere ulaşamamıştır. Bu bağlamda hızlı ve sınıflama başarımı yüksek yeni yöntemlerin araştırılması son derece önemlidir. Son yıllarda derin öğrenme konusunda yüksek sınıflama başarımları elde edildiği gözlenmektedir.

Bu tez çalışmasında EEG işaretleri, yüksek sınıflama başarımı olan derin sinir ağları kullanılarak sınıflandırılmıştır.

Çalışmada geliştirilen ağ modelleri, 2008 yarışmasında yayınlanan ve Graz Universitesi tarafından gerçekleştirilmiş veri tabanı üzerinde denenmiştir. 2008 yarışma veri tabanında 9 denek bulunmaktadır. Her denek için farklı zamanlarda alınmış 5 EEG kaydı bulunmaktadır. Kayıtlar alınırken 3 tanesi için geri besleme kullanılmıştır. Yani, kayıt alınırken ekranda gösterilen komutun ne kadar doğrulukla icra edildiği geri besleme ile deneğe bildirilir. Bu çalışmada geri besleme kullanılmadan alınan kayıtlar kullanılmiş ve elde edilmiş olan sonuçların bu kayıtlarda bile yüksek başarı kazana bildiği gösterilmiştir. Her deneğe hayali motor haraketleri icra etmesi istenilir, yani, 7 saniye kapsamında ekranda gösterilen okun yönüne uygun olarak sağ veya sol elini hareket ettirdigini düşünmesi istenlir ve bu sırada EEG kayıtları toplanır. Dolayısıyla EEG işaretlerinde sadece iki sınıfa ait bilgi bulunmaktadır. Kayıtlarda 3 kanaldan (C3, C4, Cz) alınan EEG verileri bulunmaktadır.

Çalışmada kanal sayısının sınıflama başarımı üzerindeki etkileri incelenmiştir. Bu bağlamda iki metot karşılaştırılacaktır:

(i) 3 kanaldan gelen EEG verileri, sınıflandırılması için direkt olarak derin sinir ağına (DNN) verilmektedir. Burda, öznitelik çıkarmak için başarılı sonuçları ile popüler olan Evrişimsel Sinir Ağı (CNN), sınıflandırma için ise Tamamen Bağlı Sinir Ağı (FCNN) kullanılmaktadır.

(ii) 3 kanaldan gelen EEG verileri, 5 farklı frekans bandına sahip süzgeçle süzülür. Bu durumda kanal sayısı, 15'e yükselir. 15 kanaldan gelen EEG verileri, Ortak Uzamsal Örüntü (CSP) metodu kullanılarak ayrıştırılır ve CSP çıkışındaki sinyalleri önce CNN daha sonra ise FCNN metodunu kullanarak sınıflmadırma yapılır.

Genelde EEG verilerinin daha doğru sınıflandırılabilmeleri için kanal sayısının büyük olması istenilmektedir. Ancak bu durum, sistemi daha da karmaşık hale getirebiliyor. Çalışmada sanal olarak arttırılan yeni kanalllar ile hem sınıflama başarımı arttırılır hem de sistem karmaşası azaltılır.

Çalışmada aynı zamanda derin sinir ağının genelleme özelliği de incelenmektedir. Beyin bilgisayar arayüzü sistemleri genellikle kişiye bağlı tasarlanmaktadır. Bu nedenle önce 9 deneğin her biri için ayrı bir derin sinir ağı bulunmuştur. Her ne kadar CSP basitlikle güçlü bir yöntem olsa da, bazı eksiklikleri de mevcutdur. CSP'nin beyin bilgisayar arayüzlerinde motor hareketindeki başarısı büyük ölçüde ERD (olaya dayalı senkronizasyon) ve ERS (olaya dayalı senkronizasyon) denilen fizyolojik olaylara bağlıdır ve bu yöntem çoğunlukla bireysel sistemlerde kullanılır. CSP ile yüksek sınıflama başarımı elde edilmesine karşın aslında CSP'nin kişiye/deneğe bağlı olması metodun bir dezavantajıdır. Bu dezavantaj, derin sinir ağı kullanılarak giderilmiştir. Bu bağlamda tüm 9 deneğe ait veri kullanılarak deneklerin hepsi için tek bir derin sinir ağı bulunmuştur.

Çalışmada EEG verileri, MATLAB ortamında ayrıştırılır. Filtreler ve CSP metodu kullanılarak veriler ayrı ayrı dizinlere yazılır. Daha sonra, dizinlere ayrıştırılan veriler, derin sinri ağını eğitmek için kullanılır. Derin sinir ağlarının eğitimi, GeForce 2080

Nvidia grafik kartına sahip bir iş istasyonunda yapılmıştır. Derin sinir ağı modelleri, Linuks ortamında Python dilinde Tensorflow kütüphanesi kullanılarak kodlanmıştır. Toplam EEG veri kaydının %80'si eğitim kümesi için; diğer %20'si ise test kümesi için kulllanılmaktadır. Bu tezde yüksek sınıflama başarımı veren, şahıstan bağımsız yeni bir beyin blgisayar arayüzü metodu (filter + CSP + derin sinir ağı) geliştirilmiştir.

1. INTRODUCTION

For decades, human brain has been studied for various purposes. These purposes span from extracting information from human brain by using brain imaging techniques[1] to transmitting the information into another environment in order to accomplish a given task[2].

BCI, or sometimes known as BMI (brain machine interface) allows computers to read signals created by human brain. Several methods can be challenged to learn brain activity within BCI system. Invasive and non-invasive methods are the most popular techniques that are broadly used. Invasive methods are developed to derive brain signals directly from human brain by surgery, while in non-invasive methods electrodes placed on the human scalp to measure brain activity. Because of its cheap and better resolution ability, the EEG is preferable type of non-invasive BCI methods (Figure 1.1).

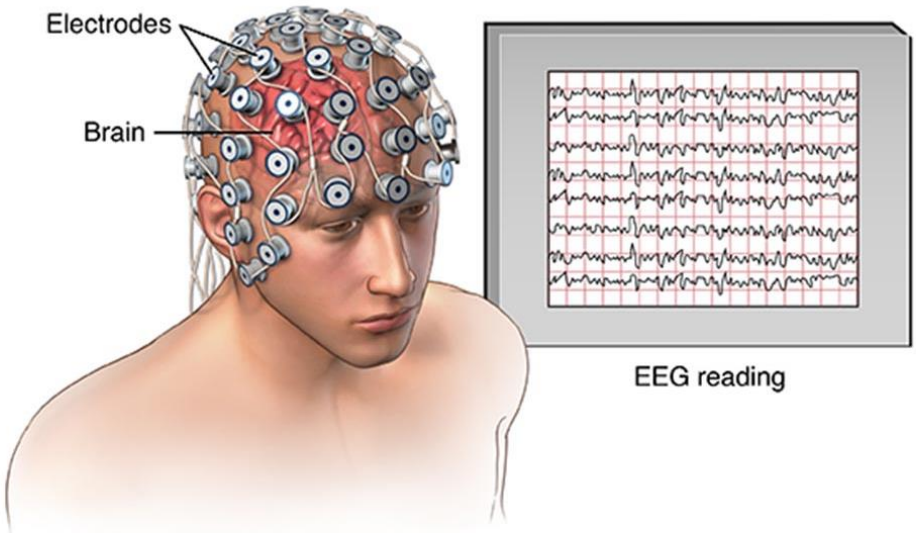


Figure 1.1: Electroencephalogram [13].

Motor imagery (MI), widely used mental task where a subject imagining himself or herself moving any body part, like hand, tongue, feet and etc. MI signals, together with Event related potentials(ERP) are the only signals that has been proven to work efficiently on BCI(Brain Computer Interface) tasks[3]. There exists motor cortex

region in the cerebral cortex which produces brain waves while executing MI movements (Figure 1.2).

BCI systems help to record EEG signals that is produces in human brain during performing motor actions. Signals are collected by electrodes attached to the specific scalp regions. Using collected signals people try to analyze brain diseases and offer cure. Figure 1.3 shows procedure of EEG signals collected in BCI system during implementing MI tasks.

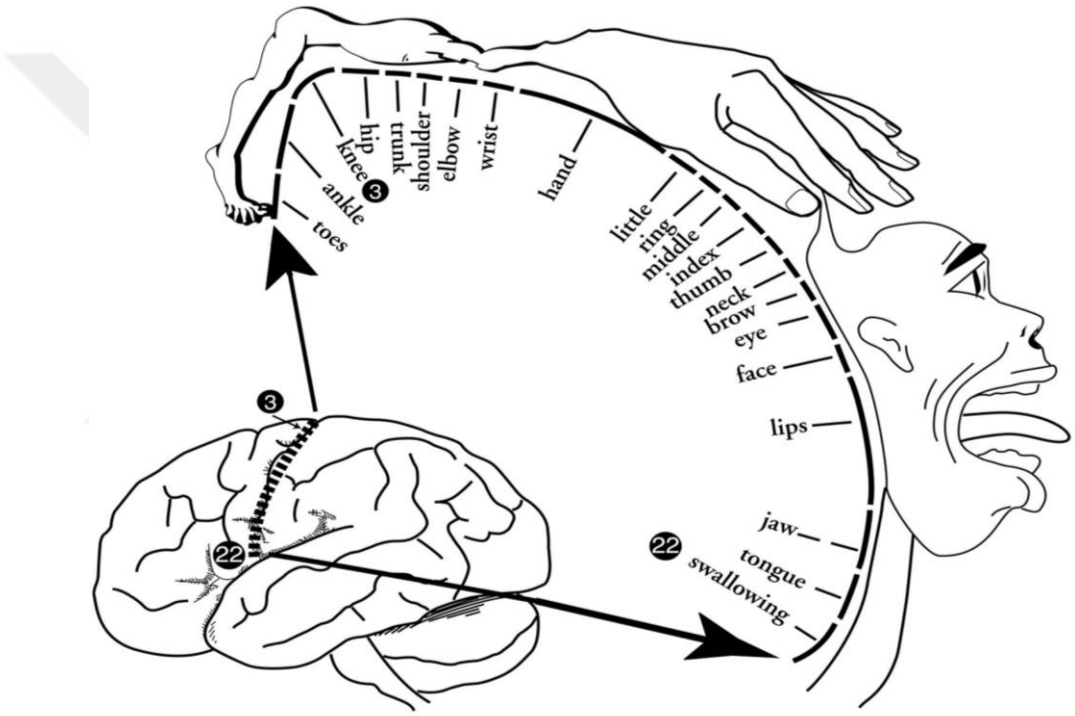


Figure 1.2: Topography of motor cortex labelled with red on the cerebral cortex [14].

Researchers have developed toolboxes and libraries in python such as Gumpy, MNE, Wyrm in order to make it easy to process the corresponding signals [4][5][6]. However, these tools are still not enough compact and easy to use. Therefore, MATLAB and Python are used together in most cases in order to employ BCI models.

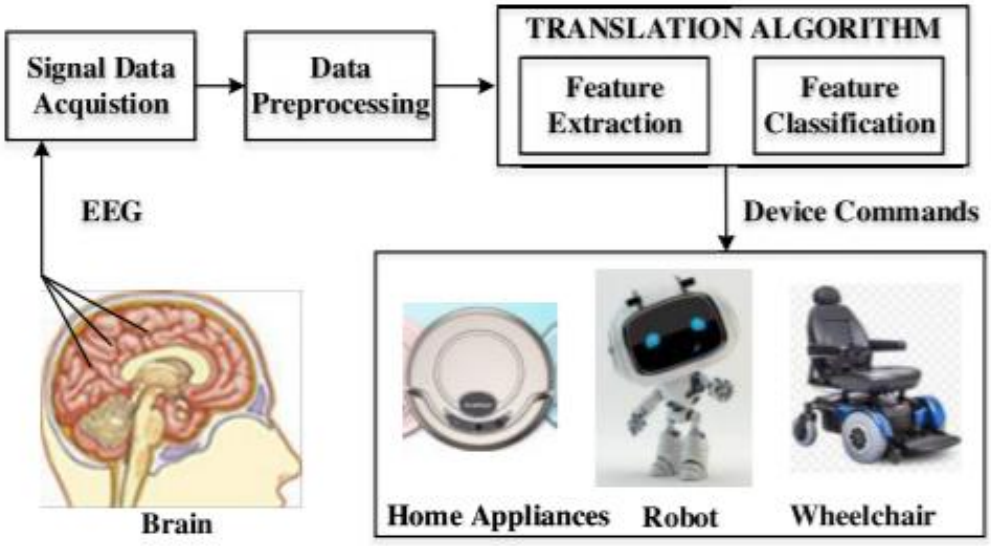


Figure 1.3: Procedure for MI based BCI system [15].

As seen from the Figure 1.3, MI BCI combines 4 main stages, namely: data collection, preprocessing, signal processing and translation to device commands. In the first stage, signals are collected, digitalized and stored with the help of EEG recorder and electrodes. In preprocessing stage collected signals involved to be filtered, cleaned and transformed and so on. Signal processing stage itself combine feature extraction and classification phases. To discriminate EEG signals, feature extraction is used. To determine the classes corresponding to different mental states, extracted features then pass to classification phase. Finally, categorized signals are translated to device commands such as wheelchair, drone or robotic something. To obtain better classification results in this research area, many studies have been done, and offered various methods.

CNN (Convolutional Neural Network), as one of the most popular deep learning models, serves to extract EEG features. In [16], researchers used CNN and SAE (stacked autoencoders) method in classifying signals recorded by EEG during performing mental task. Achieved accuracy result was 77.6%. Many studies have applied CSP algorithm, known as popular preprocessing stage. For example, Yang et al. [17] used CSP features in their CNN model to classify MI EEG measurements, and reached accuracy to 69.27%. Also, Aghaei et al. [18] proposed CSSP (common spatio spectral patterns) algorithm which required less computation and then Ang et al. [19] offered to use CSP with filter bank in order to obtain higher classification results.

In [7] Zhang et al. compared an algorithm that contains Morlet Wavelet Transformation(MWT) and neural nets with CNN by using BCI competition II dataset III which contains 280 trials that are obtained during the MI task of right and left hand. They have concluded that WNN's computational efficiency is limited. Therefore, CNN performs better. In [8] Jun yang and his coworkers have combined CNN, Discrete Wavelet Transformation, and RNN in their study that intents to uncover the patterns of different EEG tasks. In their experiment, EEG recordings of 6 subjects were used for classifying MI (left hand) and MI (Right Foot). Recordings of 7 subjects were used for classifying MI (left hand) and MI (Right Hand). They also classified the samples of MI tasks in which the imagination of left hand and tongue was the two different tasks. This experiment is carried out with 12 subjects. They have concluded that RNN(LSTM) combined with DWT and CNN is a relatively more accurate classifier than CNN alone and more capable of handling subject independency in multi-task BCI applications.

In [9] Kumar et. Al. By using the dataset 4a of BCI competition 3(140trials Left Hand, 140 Trials for Right Hand) tried to reveal the patterns via Autoencoders and then evaluated the performance of the network. They have minimized the maximum error while keeping the network computationally efficient by using RBM in combination with CSP.

In [10] jin zhang et al. have transformed first 10 seconds of Motor imagery signals into images by utilizing STFT(Short Time Fourier Transform) and tried a couple of activation functions with a CNN model that contains 7 layers. According to their study the activation function called SELU (scaled exponential linear unit) performs better than ELU (exponential linear unit) and RELU (Rectified Linear Unit). Having noted that SELU works better with STFT, it is not proven to work better than RELU in the cases where STFT is not used.

In another study[11], Huijuan Yang et. al. have combined CNN and a technique called ACSP (Augmented CSP) that is created by exploiting FBCSP and Wide Band CSP(4-30HZ). The purpose of this complicated(sophisticated) approach is to obtain as many features as possible and eliminate the not so important ones in the CNN structure so that the feature selection process would be automated. This approach has been more accurate in classifying some subjects. However, in terms of average accuracy, it did not reach the desired level of success(did not beat FBCSP).

In [12] Xiang Zhang et. al. employed convolutional recurrent neural network and an autoencoder for classifying Physionet database that is consisted of trials from 10 different subjects whose MI tasks are imagination of Left hand, Right Hand, Both hands and both feet. This network had considerable success in classifying physionet database (95.53 maximum accuracy). However, this scenario needs to be repeated with all subjects because subject independency and generalization ability is of a crucial importance in BCI.

In this study two main frameworks are created in order to observe and understand the type of the change in the success of the networks and discuss how to create more accurate systems in the interpretation of BCI. Dataset is taken from a publicly available BCI competition(BCI IV dataset 2b) This dataset consists of 9 subjects that imagined to move their right hands and left hands during the trials that has been repeated 280 times in the experiment.

In the first framework the raw EEG is fed into the network that consists of 4(and 5 in one subject) convolutional layers and one fully connected layer(FCL). The raw EEG of each subject have been split into train and test sets. Then, they are fed into the networks of their own. Then raw EEGs of all subjects are fed into the same network. The purpose of this was to evaluate the inter-subject pattern dependency of the framework. Observation of how successful different the behavior of the classifier is when being subjected to different sources of BCI signals.

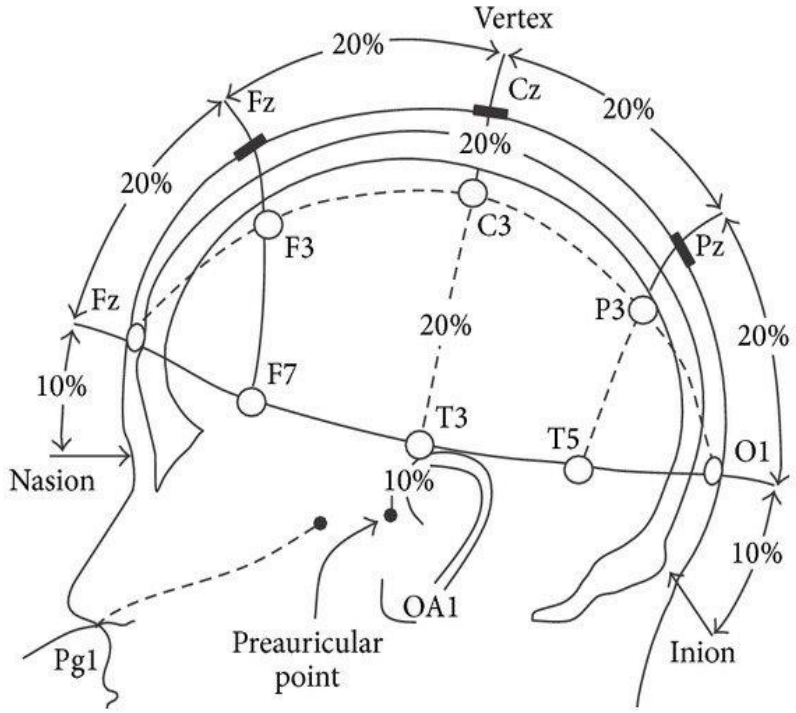
In the second framework the raw EEG is fed into the network that consists of 4 convolutional layers and one FCL. In the name of exploiting the information in different frequencies the signals are subjected to 5 different band-pass filters before they are fed into CSP and their corresponding features are extracted. The outputs of CSP filters are connected to a network that is identical to the network that is used in the first experiment. Just like the first framework the data of each subject is split into two parts as train and test dataset. 9 identical classifiers are evaluated separately, each having its own success rate in classifying motor imagery tasks. Finally, instead of feeding training sets one by one, all training sets are put together and fed into one network. Then the performance of the network is evaluated separately for each subject.

2. MOTOR ELECTROENCEPHALOGRAM SIGNALS

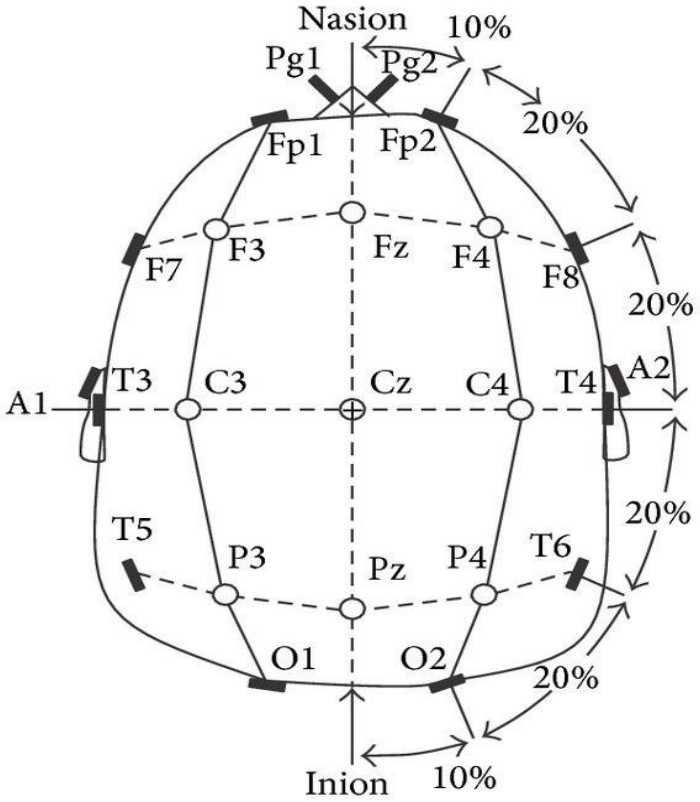
An electroencephalogram (EEG) is a measurement of electrical signals that flow in the cerebral cortex. During activation of the brain cells, the produced signals are then generating an electrical field on the scalp that can be measured by the EEG recording system. Obtained measurements by the synaptic currents from the brain can demonstrate the brain function which gives us motivation to use EEG to measure electrical activity of the brain. The term of EEG combines the concepts of electro- (recording electrical activity), encephalo- (extraction the signals from the scalp), and gram- (drawing or writing). EEG plays vital role in our life in detection, diagnosing and even in treatment of several brain disorders on human subjects.

It was Richard Caton (1942-1926) who first recorded brain generated electrical signals by using a galvanometer and two electrodes placed over the scalp in 1875, but Hans Berger (1873-1941), a psychiatrist was the one who first invented EEG in 1924[20]. Then the A.E. Kornmüller recognized the importance of multichannel recordings to cover a wider area of the brain region.

To describe the recent EEG, we think about combination of electrodes (or electrode cap) with the set of amplifiers followed by filter and pen type registrars. The electrodes used on EEG recording are usually made of high-quality silver/silver chloride discs (Ag-AgCl) with long flexible wire, plugged into the amplifier. While recording process a conductive gel (charged with ions) is used on the electrode surface to make the system able to collect potensials (nerve impulses) from the brain neurons. To start the process first, the electrodes need to be distributed to specific regions on the scalp by the standards of internationally recognized 10/20 system [21], where 10% and 20% refers the distances between adjoining electrodes.Figure 2.1 shows distribution of 20 electrodes around the circumference of the subject`s head according to the 10/20 system [22].



(a)



(b)

Figure 2.1: The placement of electrodes according to the 10-20 system. (a) side view, (b) top view.

Electrode labels denoted with capital letters represent the area of the cerebral cortex: pre-frontal (Fp), frontal (F), temporal (T), parietal (P), occipital (O), and central (C). Here, while odd-numbered electrodes (1,3,5,7) correspond the left side, even numbers (2,4,6,8) correspond the right side on the head. And also, there exist electrodes indexed with "z" which means midline lobe. Additionally, the reference electrodes are used at the earlobes indicated with A1 and A2.

After locating electorodes on individual head, he/she is instructed to close his/her eyes and relax. Switching system on, one can see brain pattern in the form of sinusoidal wave shapes on the monitor. Based on their frequency ranges brain waves seperated into five essential band groups.

(i) Alpha rhythms (α, 8-13Hz): these waves can be discovered in adults during mental inactivity or under relaxation. They can be detected in the occipital locations of the brain.

(ii) Beta rhythms (β, 13-30Hz): beta waves appear during active thinking, focusing or concentrating in normal adults. They are seen mostly around tumoural regions.

(iii) Theta rhythms (θ, 4-8Hz): Play an important role for children and infants. They can be defined especially during deep sleep. It is considered that high theta waves are abnormal for awake adults.

(iv) Delta rhythms (δ, 0.5-4Hz): they can be also detected during deep sleep.

(v) Gamma rhythms (γ, >30Hz): the occurrence of the gamma waves is rare and they have been used to detect specific diseases. The region of these waves cover frontocentral area on the brain.

Above described information tells us that not all frequency bands emphasize the same function. The following Figure 2.2 [23] depict typical brain rhythms.

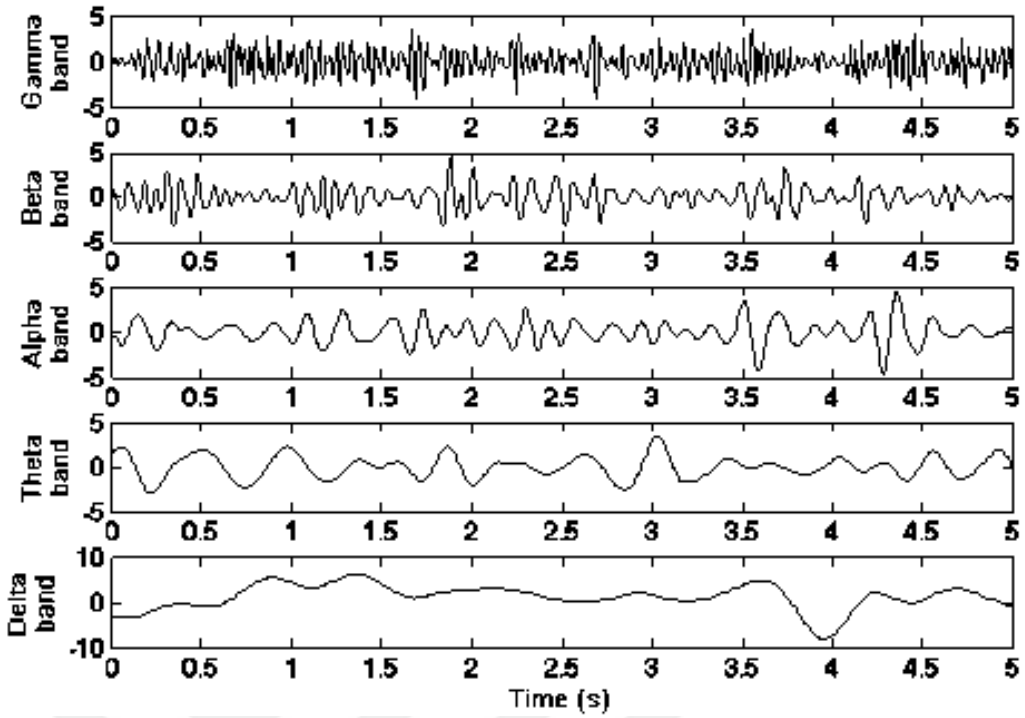


Figure 2.2: EEG frequency bands on time domain.

After obtaining upper mentioned neural signals by EEG, one can efficiently use them on various purposes:

Medical use: for diagnosing several brain disorders such as, epileptic seizures, brain tumors, sleep disorders and so on.

Research use: classification, clustering BCI based motor imagery (MI) movements recorded by EEG. Brain-computer interface (BCI) is the communication pathway where subjects are involved to perform specific motor imagery movements through EEG recording system. MI is a mental process where one imagine about for instance moving his/her arm without actually accomplishing it in real. MI signals typically, are obtained from the motor cortex part of the brain. Analyzing MI EEG signals give us an opportunity to translate the brain activity into device commands which then in the future can assist paralyzed or locked people to complete specific movements by using device commands (see Figure 2.3).

As shown in the Figure 2.3 there two stages in signal processing: feature extraction and classification stage which play important role in translating brain waves to the commands. In the subsequent sections the feature extraction and classification methods will be discussed.

It is also worth to mention that while extraction features two main frequency bands should be taken into consideration. Based on the mental task type, EEG patterns can differ in frequency bands. The one considerable band comprise the bands of μ and β which leads the decrease of EEG during imagination of left hand movement and called event-related desynchronisation (ERD).The other band includes only β band and happen just after the MI task and called event-related synchronisation (ERS).

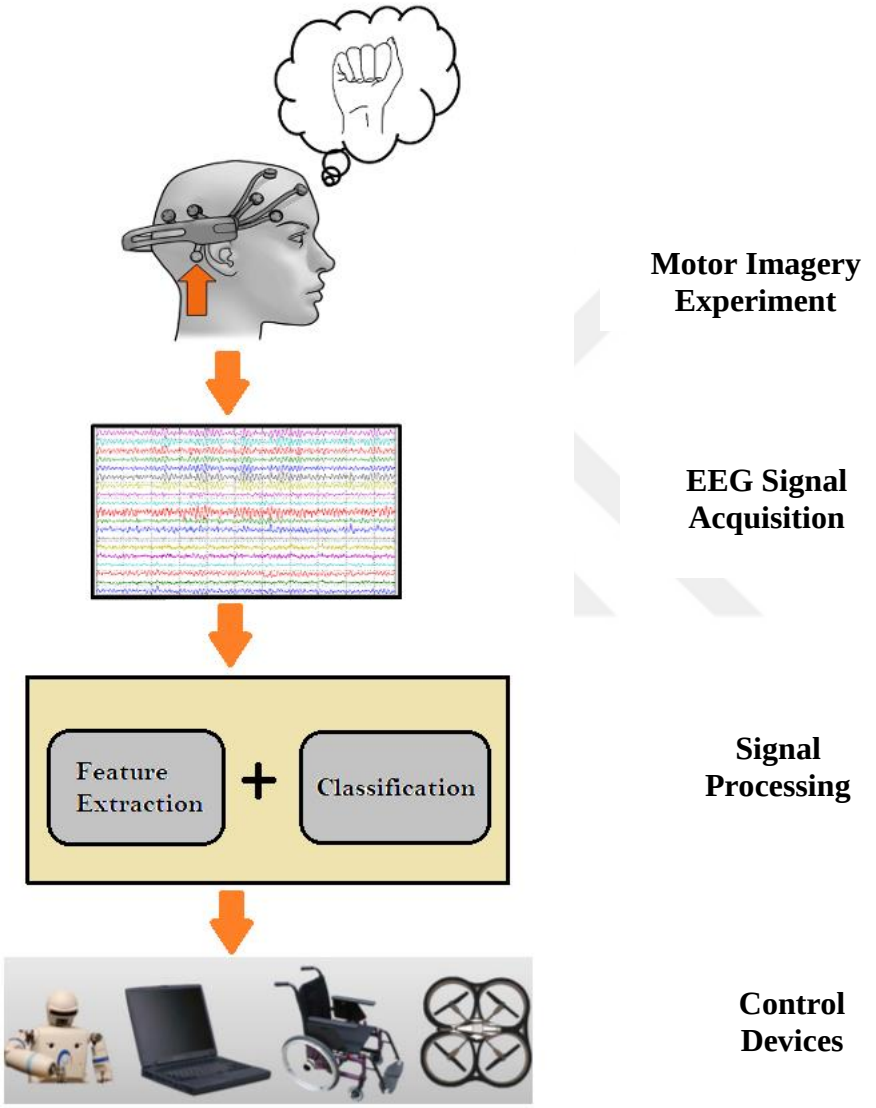


Figure 2.3: Architecture of BCI

3. COMMON SPATIAL PATTERNS AS PREPROCESSING

The first stage of EEG signal processing is to convert obtained brain waves into the action of accomplishing the subject`s intent. In this stage "feature extraction" plays an important role. As mentioned on former section, feature extraction aims to symbolize the raw data obtained from EEG signals in the form of "features". It prepares the acquired signals to be able to be translated into the BCI commands. To carry out this action feature extraction demonstrate the isolation of the important features from noise in the signal. A basic feature can be a voltage difference between two electrodes and set of these features arrange a vector, called a feature vector.

Feature extraction can be divided to number of several steps. The first step is a preprocessing. This step also includes different procedures in itself:

- Prefiltering
- Normalization
- Spatial filtering

Prefiltering procedure mainly, eliminate unuseful frequencies out of specific band and pass indicated frequency range. More detailed information about this procedure will be provided on further sections of this thesis.

Regarding normalization, here mean values first are subtracted from each signal and then divided by its variance. It is the way that signals are adjusted so the analysis of signals can be simplified in this procedure.

The process of weighting and combining the voltage signals obtained from the scalp is known as spatial filtering. Spatial filters are designed to improve EEG source localization and can be described in a matrix form [26]:

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1P} \\ y_{21} & y_{22} & & y_{2P} \\ \vdots & & \ddots & \vdots \\ y_{M1} & y_{M2} & \cdots & y_{MP} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & & w_{2N} \\ \vdots & & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1P} \\ x_{21} & x_{22} & & x_{2P} \\ \vdots & & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NP} \end{bmatrix} \quad \textbf{(3.1)}$$

or in an equation form as follow:

$$Y = WX \tag{3.2}$$

where rows and columns of X denote P signal samples from one P channel respectively; rows of W matrix combine set of N channel weights; and the Y matrix constitute M spatially filtered channels with P samples on each channel. There exist two major classes in determining W spatial filter: data-independent and data-dependent spatial filter. In this section will focus mainly to the CSP - one of the well-known methods of data-dependent filters. First, CSP [27] will briefly reviewed and then the prior knowledge about its properties will be discussed.

The development of EEG based BCIs require fast and relaible classification methods to distinguish EEG features relating with imagery movements. The method of CSP has been achieved the successful classification results on motor imagery experiments. It was first applied to detect abnormal brain waves recorded on EEG [28], but then J.Müller-Gerking et al. [29] used CSP to discriminate the different populations of EEG. It efficiently differentiates classes by maximizing variance of one class while minimizing other class. Let's go deep to CSP model and analyze its discriminative ability.

The goal of this algorithm is that to use linear transformation leading the projection of multi-dimensional EEG into the low-dimensional space by the projection matrix. Before moving to projection matrix lets first analyze the following optimization problem:

$$R_i = \frac{X_i X_i'}{trace(X_j X_j')} \tag{3.3}$$

where,R_i denotes normalized covariance matrix and X_i denotes the preprocessed EEG signal matrix in the two conditions (imagination of the left and right hand movement) with dimensions $N \times T$ ($X_i \in \mathbb{R}^{N \times T}$) in other words an epoch, where i is the epoch number per class, N is the number of channels and T is the number of samples per channel. X' is the transpose of X and trace() function computes the sum of the diagonal elements. Thinking that we have two classes we need to calculate R_i for i trials and average sum of the trials for each class. Then we need to sum averaged spatial covariances described as below:

$$R = \bar{R}_{left} + \bar{R}_{right} \quad \textbf{(3.4)}$$

Afterwards, R can be decomposed as

$$R = B\lambda B' \quad \textbf{(3.5)}$$

where λ is a diagonal matrix of eigenvalues and B is a corresponding eigenvector. Using the formula of whitening transformation for simultaneous diagonalization

$$S = \sqrt{\lambda^{-1}}B' \quad \textbf{(3.6)}$$

The spatial covariances R_{left} and R_{right} can be transformed as

$$P_{left} = S\bar{R}_{left}S' = U\lambda_{left}U' \quad \textbf{(3.7)}$$

$$P_{right} = S\bar{R}_{right}S' = U\lambda_{right}U' \quad \textbf{(3.8)}$$

where U indicates orthonormal and P_{left}, P_{right} share common eigenvectors, since

$$P_{left} + P_{right} = S\bar{R}S' = 1 \quad \textbf{(3.9)}$$

which corresponds to the sum of the two diagonal matrices of eigenvalues

$$\lambda_{left} + \lambda_{right} = I \quad \textbf{(3.20)}$$

I is the identity matrix. Assuming that eigenvalues are sorted in a descending order, the feature vectors of two population of EEG can be discriminated by the first and the last eigenvectors of U which proof discriminative ability of spatial filtering. We can then obtain projection matrix W from the whitened covariance matrices of EEG as following:

$$W = (U'S)' \quad \textbf{(3.31)}$$

where rows of W are the stationary spatial filters and columns of the W' is called the common spatial patterns, in other words the CSP. Using CSP algorithm the

decomposition of matrix W and an eigenvector can be generated as described on equation (3.2).

The use of CSP is illustrated in Figure 3.1[25] below shows four spatial filters that try to maximize the variance of signals of left hand class MI, while minimizing right hand class MI (first top two filters) and vice versa (last bottom two filters). This can be seen as in light and dark grey windows indicating of right and left hand motor imagery, respectively.

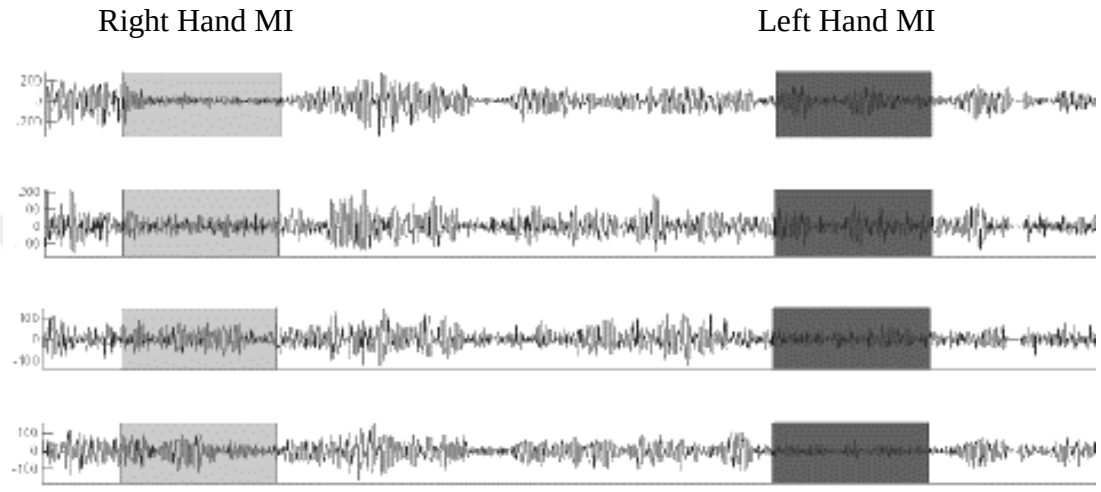


Figure 3.1: EEG signals spatially filtered using the CSP algorithm.

In a nutshell, we can summarize this section that, CSP filtering is highly recommended classification method which can efficiently discriminate class features by maximizing the variance of EEG signals relating to one class while minimizing the other.

4. CLASSIFICATION OF THE MOTOR EEG SIGNALS

4.1 Architecture of Deep Neural Network

This section aims to give the essential background information about the context of CNN and FCNN (fully connected neural network). CNN in other words, ConvNet is the feedforward neural network in the deep learning, constructed to learn the necessary features and proposed by [30].

The information in the feedforward neural network flows forward from multiple layers in which each node in the layers represents a linear combination of input. Then these linear combinations are converted to non-linear activation functions and move to the subsequent layer.

It forms the layers of neurons with three dimensions: width, height, depth (size of filter, input layer and output respectively). There exist the three main subjects one should know about CNN architecture: convolutional layer, pooling layer and activation functions.

Here on the network, convolutional layers play an important building block role that does the most computations in. Each convolutional layer consists of filters that can be convolved with the input data stride by stride. The stride for sliding filter should be specified in advance. If filter slides with one stride it means it moves one pixel at one time, or two strides then two pixels. While moving along the pixels, an activation map is produced in response which create an output layer (or input of the next layer) (see Figure 4.1.1).

Pooling or in another word sub-sampling layer, on the other hand is used between convolutional layers to reduce computation and parameter size in the network. It downsamples every slice of input spatially. There typical options of pooling layer are:

- Mean pooling
- Max pooling
- Sum pooling

To compare the types, it has been proven that max pooling can perform better performance in compare to mean pooling. Max-pooling serves to reduce the size of the input. Example for max-pooling is described on Figure 4.1.2.

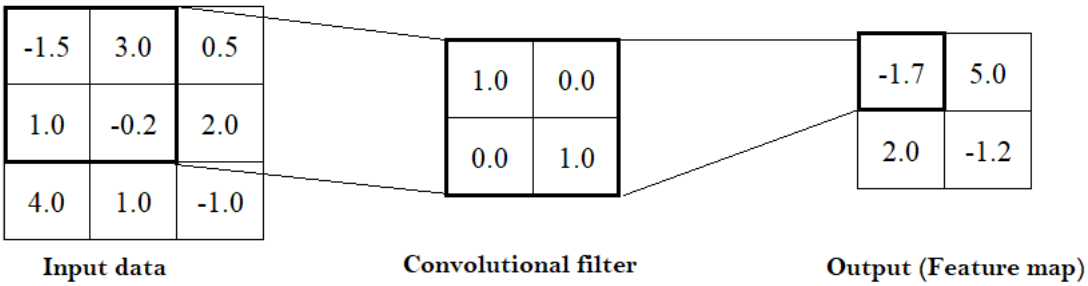


Figure 4.1.1: Example of convolutional filter (2x2) convolving with 3x3 input data with one stride which produce 2x2 feature map.

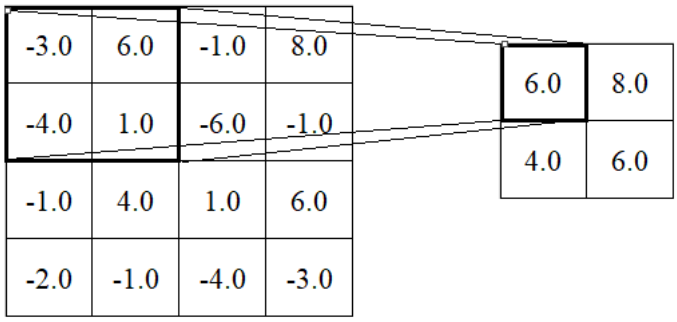


Figure 4.1.2: Example of max-pooling (2x2) on input data (4x4) with two stride.

But many people suggest to eliminate the pooling layer on the CNN architecture as it can remove the valuable parameters which can be disregarded.

Activations play the worthy part in neural network designs. They can show how on the network, inputs are transforming. It is preferable to use non-linear functions as they allow network to create a complex function which increases ability of learning. There exist several traditional activation functions that have been used by researchers. The Rectified Linear Unit (ReLU) is one of the most widely used and also utilized on the proposed frameworks in this thesis. It removes negative pixels in the activation map and sets them to zero[31]:

$$ReLU(x) = \max(0, z) \quad \textbf{(4.4)}$$

There are several advantages of ReLU as it is much more efficient and provides much more accuracy compared to other activation functions. Also, thanks to ReLU every activated neuron can pass to the following layer.

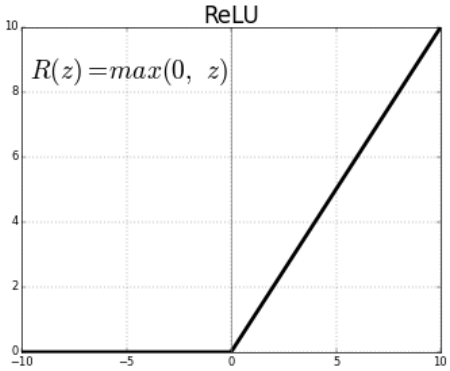


Figure 4.1.3: Plot of ReLU [32].

FCNN (Fully Connected Neural Network), the inspired variant of MLP (multilayer perceptron) and modeled with CNN which plays classifier role. It helps to classify the features extracted by CNN. All the neurons here have the connection with the previous layer. Figure 4.1.4 [33] below shows basic example for FCNN architecture:

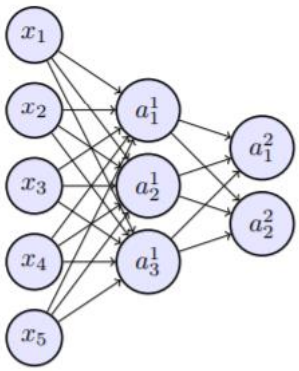


Figure 4.1.4: FCNN with three layers; first layer with five inputs; second layer with three hidden units and last layer with two outputs.

To summarize the above network as a function $f: \mathbb{R}^N \rightarrow \mathbb{R}^M$, where N and M are dimensions for input and output respectively [34]. The output function:

$$f(x) = s(W^2\big(s(W^1x + b^1)\big) + b^2) \qquad \textbf{(4.2)}$$

where $W^l \in \mathbb{R}^{N_l \times N_{l-1}}$ indicates weight matrix between the layers of l and (l-1). The weight that connects the node j with i in layers l and l-1 respectively, can be represented as $w_{ji}^l \in W^l$. $b^l \in \mathbb{R}^{N_l}$, is the bias vector and produce a linear shift of weights and inputs. An activation function that is used equation (4.2) is described as s() which plays a role of serving a non-linear transformation in data.

4.2 Classification Process by using DNN

In order to better learn and analyze the recently famous BCI-based EEG data, many researchers have applied deep learning (DL) methods. Because of its more advantageous aspects, most authors have preferred to use the CNN among the DL algorithms.

In this part of the chapter we will look the details of proposed first method called Framework1(FW1). This framework presents an application of CNN+FCNN without of any preprocessing stage to classify motor imagery intents recorded by EEG. In FW1, the CNN+FCNN is applied to raw MI-EEG data directly to extract and classify the features respectively (see Figure 4.2.1). During the training step in this classification method both whole data (all data that is collected from nine subjects) and nine separate data which belong to nine subjects were experienced. 80% of the total EEG data is used for the training set and other 20% is used for the test set. To understand the general idea about proposed framework, the model that is experienced on the whole data will only be explained in detail below. In this model the network combines 4 convolutional layers, 1 input layer and 1 fully connected layer, 1 hidden layer and 1 output layer which is shown on Figure 4.2.2 and explained below in detail.

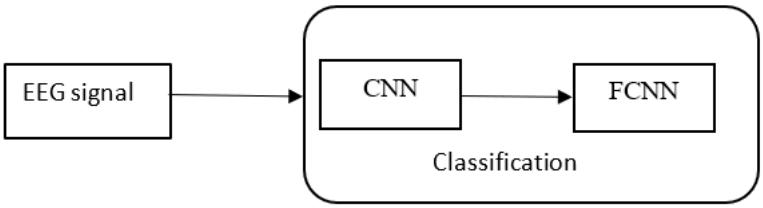


Figure 4.2.1: CNN model for proposed framework 1.

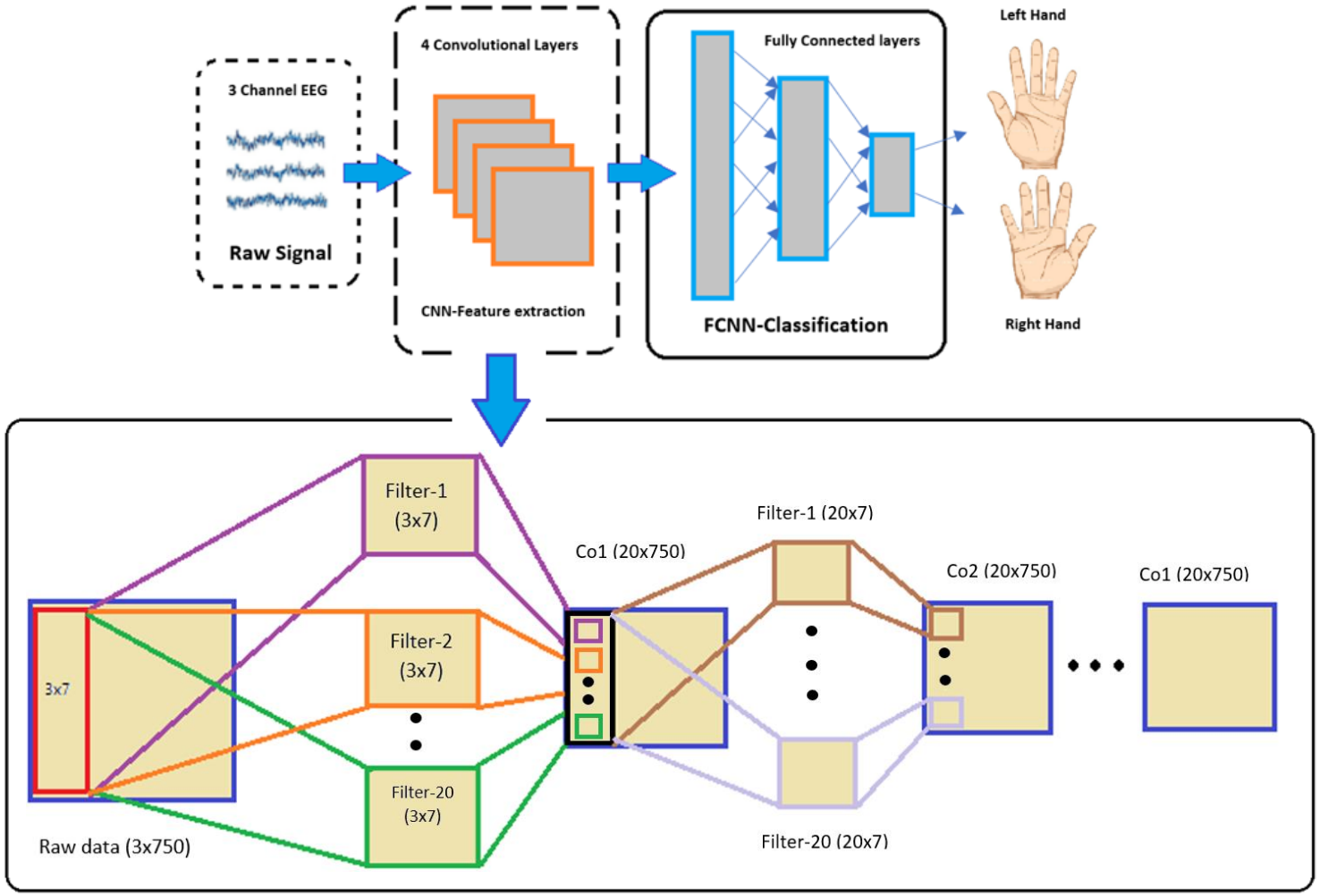


Figure 4.2.2: Architecture for proposed CNN+FCNN.

(1) The first layer (L0) is input layer in the form of 3x750 matrix, where 3 indicates number of channels (C3, C4 and Cz), and 750 is the number samples recorded from each channel.

(2) The convolutional layers (Co1, Co2, Co3, Co4) or in another word hidden layers can be called. These layers serve to extract spatial features by convolving the BCI based EEG signals with 7 filters (one stride) which results 20 feature maps at the end of the layer. Speed up the learning "batch normalization" is applied followingly. Doing so, we let layer to learn independently. As activation function, ReLU is used to all hidden layers. These layers are repeated 4 times.

(3) F5 (750x20): fully connected layer (FCL) where each neuron is connected with of all neurons on layer Co4. In this layer the output of the previous layer is flattened and then connected to H6-hidden layer. Also, dropout layer is added next to H6 to reduce overfitting.

(4) H6: One more hidden layer with 1024 neurons is followed after F5 to perform the classification. To help decreasing interdependent learning, we need dropout layer again to be used right after H6.

(5) O7: the output layer with two neurons representing two classes (lef hand and right hand) of the problem.

4.3 Classification Process by using CSP and DNN

The second proposed method, called Framework 2 (FW2) consists of 3 progressive stages: (i) prefiltering using multiple butterworth band pass filters, (ii) spatial filtering using the CSP algorithm(preprocessing) and (iii) feature extraction and classification using CNN and FCNN deep learning algorithms.

1. Prefiltering: the first step, using a filter bank that splits the EEG data into the multi-frequency bands using a fifth-order butterworth at 250 Hz sampling frequency. Total 5 band pass filters are used, namely, 6-12Hz, 12-18Hz, 18-24Hz, 24-30Hz,

30-36Hz. We use prefiltering prior to spatial filtering to deal with the sensitivity of the CSP to artifacts such as to noise or eye blinking in the EEG records and achieve better feature extraction results.Various bands of the filter bank exist that are effective, but the most effective frequency ranges in classification MI EEG are shown to encompasses beta and mu frequency bands [35].

2. Spatial filtering: the second step plays role of spatial filtering where the algorithm of CSP execute the feature extraction. This preprocessing phase is used to detect ERD (Event-Related Desynchronization) and ERS (Event-Related Synchronization), which are very important in subsequent calculations [36][37]. In order for extracting features, it is necessary to obtain the most distinctive ERD and ERS to select the best appropriate frequency band that suits for each subject.
3. Classifcation: the 3rd stage that employs CNN and FCNN to perform a feature extraction and classification of EEG features that is successfully preprocessed by CSP. Various studies in [39] [40] showed that using CNN model can yield to better results on the BCI based EEG datasets.

Figure 4.3.1 illustrates the classification of two-class MI EEG signals according to FW2.

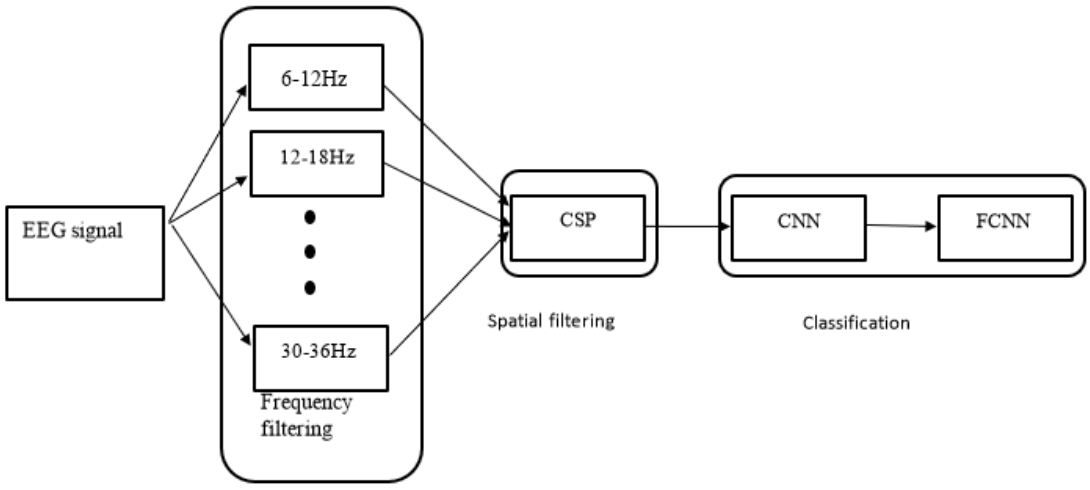


Figure 4.3.1: Framework 2 model.

5. COMPUTER SIMULATIONS

5.1 BCI Database

In this study, we used the EEG data that presented on BCI Competition Dataset IIb[41]. Dataset comprise two classes of MI tasks (left hand and right hand) recorded from nine subjects on two separate(different) days. In total five sessions were provided per subject, including 3 training and 2 evaluation sessions. We will work only with the first two sessions of training part that consist of 240 trials without feedback in total (120 trials per session, 60 trials per class). Here each trial begins with a fixation cross and a short acoustic tone (1 kHz, 70 ms) to prepare a subject to focus on the following command will be displayed on the monitor. At time t=3s a cue in the form of arrow appear pointing to the left or right on the screen to guide the subject to execute the corresponding MI tasks of left hand and right hand respectively till t=7s. Afterward a trial continues with a break that lasts 1.5s. The paradigm for one trial is illustrated in Figure 5.1.1.

The data for each session collected over bilaterally arranged three bipolar channels (C3, Cz and C4) according to the 10/20 system (see Figure 5.1.2).

In this section, the experimental results of proposed methods have been reported. The classification performances were calculated by distributing 80% of the data (each session separately) for training set and 20% for testing set. In general, two techniques employed using BCI dataset: FW1(model without preprocessing stage) and FW2 (model with preprocessing stage) have been described above. MATLAB programs are applied to raw motor EEG signals. GDF files can be loaded by using SioSig toolbox, available for free at http://biosig.sourceforge.net/ and all the calculated classification accuracies on the test set are conducted using Python codes running on Ubuntu Linux workstation. Corresponding results represented on the tables below. The workstation used in this study has 32 core CPUs of 2.7 GHz with GeForce GTR2080 Graphics card.

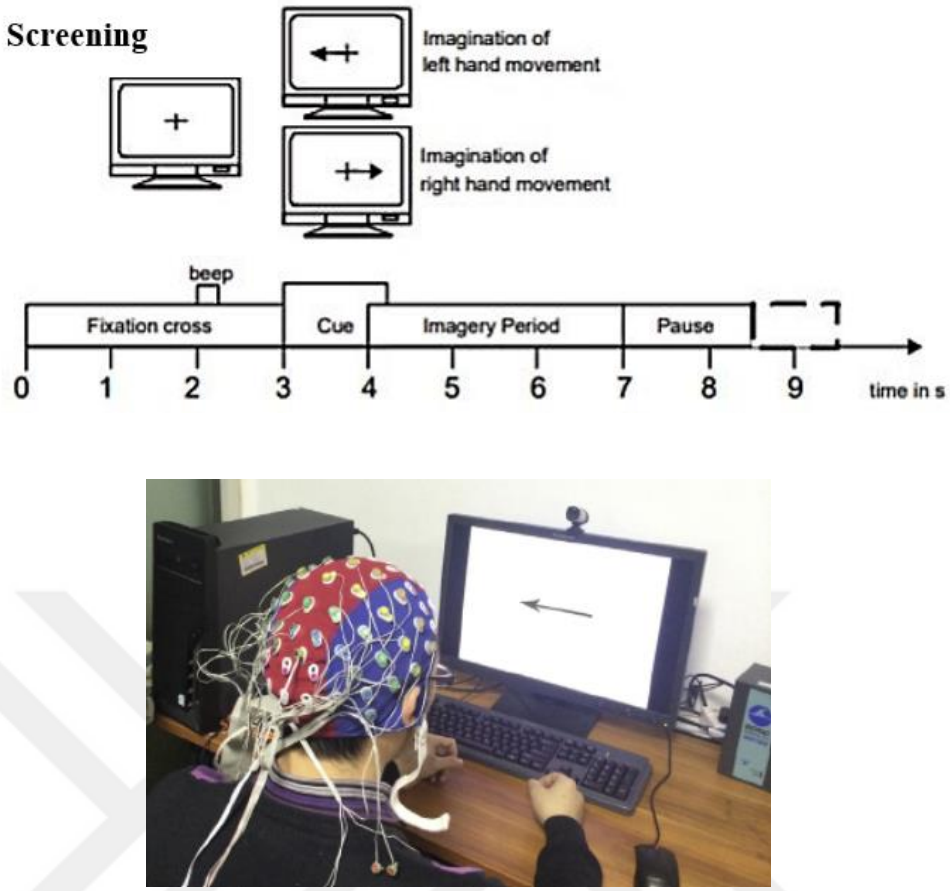


Figure 5.1.1: Timing scheme (the top) [41] and sample picture (the bottom) [42]of training data without feedback.

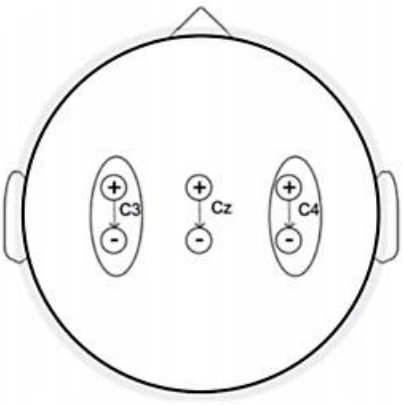


Figure 5.1.2: Locations of C3, C4 and Cz considering the conventional 10/20 system [41].

5.2 Classification Without Preprocessing Stage

The classification accuracies achieved by FW1 is described in detail as follows. During signal processing, no any preprocessing stage is used. Raw data is directly employed by CNN+FCNN. Method is experienced on every 9 datasets belonging 9 subjects (S1, S2, S3, S4, S5, S6, S7, S8, S9) and the best accuracy value for each subject has showed on the Table 5.3 Moreover, the averaged result that is calculated over 9 subjects is reported in the table too. During classification of datasets, different convolution layer size is used in CNN architecture. Table 5.5 describes layer sizes that is calculated for each subject.

5.3 Classification with Preprocessing Stage

To see the effect, FW2 was constructed with preprocessing in compare to FW1. Here, first CSP is used as feature extraction and then followed by CNN+FCNN to classify the two class MI EEG measurements. CSP interpreted to be the most effective method as feature extr

actor in classifying two class MI datasets [43]. But as it has sensitivity to noise and other artifacts [44], it is recommended to use CSP with prefiltering stage [45] where input data first filtered between 6-36 Hz bandpass frequencies. Obtained results is described on Table 5.4.

Table 5.6 show the distributed input sizes of a maximum of 5 convolutional layers (Layer 1, Layer 2, Layer 3, Layer 4, Layer 5) calculated for each 9 subjects (S1,S2,S3,S4,S5,S6,S7,S8,S9) in FW1 and FW2 respectively. Input size is indicated with three values where first value – I is filter size, second value – F is filter size and third value – O is output size.

5.4 Generalization of the Proposed Frameworks

Evaluated performances of proposed frameworks experienced on motor imagery EEG data and comparison results have been carried for FW1 and FW2 based on the performances that is obtained on whole data and subject-specific data as shown on Table 5.1.

From obtained results, that is described above on the table, it is obviously seen that proposed second method – FW2 experienced on 9 different subject specific datasets gives superior averaged accuracy value of 74% among the other results.

5.5 Performances obtained by studies in literature

Offered two methods in the literature by Dai and Tabar, experienced on BCI Competition IV dataset have been also compared with the test set accuracy value obtained in this study. It is necessary to mention that, as those authors used the whole data to calculate the performance we will only compare the result calculated using whole data where accuracy is 72.4%.

Table 5.1: Generalized comparison of FW1 and FW2.

	Whole Data	The Means of Accuracies obtained from each subject
Training Accuracy	100%	100%
Test Accuracy FW1	64.3%	61.2%
Test Accuracy FW2	72.4%	74%

Table 5.2: Performances of Test Set accuracies obtained by two methods in Literature and in this study.

Studies	Accuracies for Test Set
In Dai' study [46]	78.2%
In Tabar' study [16]	77.6%
Proposed Method	72.4%

Table 5.3: Subject-specific classification accuracies in FW1.

Subject	S1	S2	S3	S4	S5	S6	S7	S8	S9	Mean
Training Accuracy	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Test Accuracy	60.2%	59%	55.7%	87.6%	58.1%	48.2%	61.2%	60%	59.3%	61.2%

Table 5.4: Subject-specific classification accuracies in FW2.

Subject	S1	S2	S3	S4	S5	S6	S7	S8	S9	Mean
Training Accuracy	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Test Accuracy	79.1%	61.9%	63.2%	94.2%	72.5%	82.9%	66%	72.3%	65%	74%

Table 5.5: Distribution of input sizes of convolutional layers for each subject in FW1.

Subject	Layer 1 I,F,O	Layer 2 I,F,O	Layer 3 I,F,O	Layer 4 I,F,O	Layer 5 I,F,O
S1	3,5,40	40,5,40	40,5,40	40,5,40	NA
S2	3,18,20	20,18,20	20,18,20	20,18,20	NA
S3	3,15,40	40,15,40	40,15,40	40,15,40	40,15,40
S4	3,15,20	20,15,20	20,15,20	20,15,20	NA
S5	3,15,20	20,15,20	20,15,20	20,15,20	NA
S6	3,15,20	20,15,20	20,15,20	20,15,20	NA
S7	3,15,20	20,15,20	20,15,20	20,15,20	NA
S8	3,15,20	20,15,20	20,15,20	20,15,20	NA
S9	3,18,17	17,18,17	17,18,17	17,18,17	NA

Table 5.6: Distribution of input sizes of convolutional layers for each subject in FW2.

Subject	Layer 1 I,F,O	Layer 2 I,F,O	Layer 3 I,F,O	Layer 4 I,F,O
S1	20,23,20	20,23,20	20,23,20	20,23,20
S2	6,13,20	20,13,20	20,13,20	20,13,20
S3	2,3,20	20,3,20	20,3,20	20,3,20
S4	12,3,30	30,3,30	30,3,30	30,3,30
S5	10,13,30	30,13,30	30,13,30	30,13,30
S6	2,13,20	20,13,20	20,13,20	20,13,20
S7	2,13,30	30,13,30	30,13,30	30,13,30
S8	14,3,30	30,3,30	30,3,30	30,3,30
S9	10,23,20	20,23,20	20,23,20	20,23,20

6. CONCLUSION

As time passes, technological innovations become indispensable in our lives. In this case, people incessantly aim to make new discoveries or improve what others did. In general, all these technological innovations, whether in medicine, in education or in our daily lives, are aimed at facilitating human life. BCI is one method which is used exactly in this type of studies. Here, different experiments are carried out on healthy or disabled people, in order to help people with brain disorders such as paralyzed people. While conducting the experiments, subjects involved in the experiment are fitted with a number of electrodes or a cap consisting of electrodes on their heads. Obtained results are collected by EEG recorder. EEG, in other words, the electroencephalogram prints the signals produced in the human brain on paper or computer with the help of electrodes placed in different parts of the scalp. EEG can be used in many fields: medicine, experimental research laboratories and so on. Even many diseases: such as epilepsy, brain tumor, memory disorders, sleep problems, stroke and etc. can be detected using EEG. In scientific researches, for example in BCI, the most preferred signal collection method is exactly EEG. Being non-invasive makes EEG preferable, as this technique requires no surgery during signal recording. Speaking of experiments, experiments with BCI are performed in different ways, based on purposes. But since only one of them is used in this study, it is considered appropriate to mention only one of them.

Study has been completed on “BCI Competition IV, Dataset IIb” dataset, which is online available. Nine subjects took part during two-class MI experiments. And each is asked to perform MI of left hand and right hand during both training and test experiments. In total five sessions were provided per subject, including 3 training and 2 evaluation sessions. Many studies in the literature have used this dataset to calculate the classification accuracies by proposing various classifier models. In [16], researchers introduced CNN+SAE (Stacked Autoencoder) method and achieved to the accuracy value of 77.6%. In [46], researchers were able to increase the performance to 78.2% by using CNN+VAE (Variational Autoencoder). But this study presents two

separate methods, namely Framework-1 (FW1) and Framework-2 (FW2) for classification MI based EEG measurements. The first introduced method - FW1 utilize deep learning-based scheme: CNN+FCNN directly to the input raw data, while second one – FW2 use preprocessing stage with prefiltering+CSP followed by CNN+FCNN classifier. Each framework experienced for both whole dataset and 9 subject- specific datasets. It is necessary to mention that, we used only the first two sessions of training part that consist of 240 trials without feedback in total (120 trials per session,60 trials per class). Using 80% (96 trials) for training and 20% (24 trials) for test of each session and then summing (96 (session 1) +96 (session 2); 24 (session 1) +24 (session 2)) them make a homogeneous distribution between all trials. To give details about results: we obtained classification performances of 64.3% and 72.4% by using whole data; 61.2% and 74% bu using 9 differet subject specific datasets in FW1 and FW2 respectively. It is clearly seen that, using preprocessing stage in FW2 can lead to more efficient result. And in order to get better performance, study suggests classifying the collected subject-specific datasets separately.

REFERENCES

[1] D. Tan and A. Nijholt., 2010**:** "Brain-computer interfaces and human-computer interaction," in Brain-Computer Interfaces, Springer, 2010, pp. 3–19**.**

[2] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He., 2013**:** "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface," J. Neural Eng., vol. 10, no. 4, p. 46003, 2013.

[3] F. Lotte and I. B. Sud-ouest., 2012**:** "BCI's Beyond Medical Applications.pdf," pp. 26–34, 2012**.**

[4] Z. Tayeb et al., 2018**:** "Gumpy: A Python toolbox suitable for hybrid brain–computer interfaces," J. Neural Eng., vol. 15, no. 6, p. 65003, 2018**.**

[5] B. Venthur, S. Dähne, J. Höhne, H. Heller, and B. Blankertz., 2015**:**"Wyrm: A brain-computer interface toolbox in python," Neuroinformatics, vol. 13, no. 4, pp. 471–486, 2015**.**

[6] A. Gramfort et al., 2014:"MNE software for processing MEG and EEG data," Neuroimage, vol. 86, pp. 446–460, 2014.

[7] Z. Zhang et al., 2019: "A Novel Deep Learning Approach With Data Augmentation to Classify Motor Imagery Signals," IEEE Access, vol. 7, pp. 15945–15954, 2019.

[8] J. Yang, S. Yao, and J. Wang., 2018: "Deep Fusion Feature Learning Network for MI-EEG Classification," IEEE Access, vol. 6, pp. 79050–79059, 2018.

[9] S. Kumar, A. Sharma, K. Mamun, and T. Tsunoda., 2016: "A deep learning approach for motor imagery EEG signal classification," in 2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE), 2016, pp. 34–39.

[10] J. Zhang, C. Yan, and X. Gong., 2017:"Deep convolutional neural network for decoding motor imagery based brain computer interface," in 2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), 2017, pp. 1–5.

[11] H. Yang, S. Sakhavi, K. K. Ang, and C. Guan., 2015: "On the use of convolutional neural networks and augmented CSP features for multi-class motor imagery of EEG signals classification," Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS, vol. 2015-Novem, pp. 2620–2623, 2015.

[12] X. Zhang, L. Yao, Q. Z. Sheng, S. S. Kanhere, T. Gu, and D. Zhang., 2018: "Converting your thoughts to texts: Enabling brain typing via deep feature learning of eeg signals," in 2018 IEEE International Conference

on Pervasive Computing and Communications (PerCom), 2018, pp. 1–10.

[13] Dr.Mridha., "EEG", http://drmridha.com/services/eeg

[14] Sage,G. (1971). Introduction to motor behavior: a neuropsychological approach, Addison-Wesley series in physical education, Addison-Wesley Pub. Co., http://books.google.com.tr/books?id=F0VqAAAAMAAJ.

[15] G. Xu et al., 2019: "A Deep Transfer Convolutional Neural Network Framework for EEG Signal Classification", IEEE Access, vol.7, pp. 112767-112776.

[16] Y. R. Tabar and U. Halici., 2017: "A novel deep learning approach for classification of EEG motor imagery signals," J. Neural Eng., vol. 14, no. 1, p. 16003.

[17] H. Yang, S. Sakhavi, K. K. Ang, C. Guan., 2015: "On the use of convolutional neural networks and augmented CSP features for multi-class motor imagery of EEG signals classification", *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, pp. 2620-2623, Aug**.**

[18] A. S. Aghaei, M. S. Mahanta, K. N. Plataniotis., 2013: "Separable common spatio-spectral pattern algorithm for classification of EEG signals", *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 988-992, May.

[19] A. Ashok, A. K. Bharathan, V. R. Soujya, P. Nandakumar., 2013: "Tikhonov regularized spectrally weighted common spatial patterns", *Proc. Int. Conf. Control Commun. Comput. (ICCC)*, pp. 315-318, Dec.

[20] Massimo, A., 2004: "In Memoriam Pierre Gloor (1923-2003): an appreciation', Epilepsia, 45(7), July,882

[21] H.H. Jasper. 1958: The ten-twenty electrode system of the International Federation. Electroencephalography and Clinical Neurophysiology, 371-375

[22] S. Sanei and J. A. Chambers., 2007: EEG Signal Procesing, JohnWiley& Sons, New York, NY, USA.

[23] M. Abo-Zahhad, Sabah M. Ahmed, Sherif N. Abbas., 2015: A New EEG Acquisition Protocol for Biometric Identification Using Eye Blinking Signals, I.J. Intelligent Systems and Applications, 06, 48-54

[24] Blankertz B., Tomioka R., Lemm S., Kawanabe M., Müller K.-R., (2008b). Optimizing spatial filters for robust EEG single-trial analysis. IEEE Signal Process. Mag. 25, 41–5610.1109/MSP.2008.440844

[25] Fabien Lotte., 2014: "A Tutorial on EEG Signal Processing Techniques for Mental State Recognition inBrain-Computer Interfaces", Eduardo Reck Miranda; Julien Castet. Guide to Brain-ComputerMusic Interfacing, Springer.

[26] Jonathan Wolpaw and Elizabeth Winter Wolpaw., 2012: "Brain–Computer Interfaces: Principles and Practice", Oxford Scholarship Online: May

[27] Fukunaga, K. (1990).," Introduction to Statistical Pattern Recognition, 2nd Ed" NewYork:Academic Press.

[28] Z. J. Koles., 1991 : "The quantitative extraction and topographic mapping of the abnormal components in the clinical EEG," Electroenc. Clin. Neurophys., vol. 79, pp. 440–447.

[29] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg., 1999 : "Designing optimal spatial filters for single-trial EEG classification in a movement task," Electroenc. Clin. Neurophys.

[30] Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner., 1998: "Gradient-based learning applied to document recognition", proceeding of the IEEE, vol.86, pp.2278-2324. Nov. Pages 8,9,15.

[31] F.-F. Li and A. Karpathy., 2015: "Stanford CS231n course materials." http://cs231n. github.io/convolutional-networks/. Accessed: 03-09-2015. pages 9, 12, 15

[32] Saugat Bhattarai., "What is Activation Functions in Neural Network (NN)?", https://saugatbhattarai.com.np/what-is-activation-functions-in-neural-network-nn/

[33] Ian Walker., 2015: "Deep Convolutional Neural Networks for Brain Computer Interface using Motor Imagery", Department of Computing Imperial College of Science, Technology and Medicine. 4 September.

[34] T.D. Team., 2013: "Deep learning tutorials", http://deeplearning.net/tutorial/index.html, pages 9,18

[35] G.Pfurtscheller and et al., 1997: "EEG-based discrimination between imagination of right and left hand movement", Electroencephalography and Clinical Neurophysiology Volume 103, Issue 6, December, Pages 642-651

[36] Pfurtscheller G, Aranibar A., 1979: "Evaluation of event-related desynchronization (ERD) preceding and following voluntary self-paced movement", Electroencephalogr Clin Neurophysiol. Feb;46(2):138-46

[37] Pfurtscheller G1, Lopes da Silva FH., 1999 : "Event-related EEG/MEG synchronization and desynchronization: basic principles", Clin Neurophysiol. Nov;110(11):1842-57.

[38] Y Lu, H Jiang, W Liu., 2017: "Classification of EEG Signal by STFT-CNN Framework: Identification of Right-/left-hand Motor Imagination in BCI Systems", CENet2017 22-23 July Shanghai, China.

[40] Hyeon Kyu Lee ; Young-Seok Choi and et al., 2018: "A convolution neural networks scheme for classification of motor imagery EEG based on wavelet time-frequency image", International Conference on Information Networking (ICOIN)

[41] Robert Leeb, Claudia Keinrath, Reinhold Scherer, et al., 2007: "Brain–Computer Communication: Motivation, Aim, and Impact of Exploring a Virtual Apartment", IEEE Transactions on Neural Systems And Rehabilitation Engineering, Vol. 15, No. 4, December

[42] Zhichuan Tang and et al., 2016: "A Brain-Machine Interface Based on ERD/ERS for an Upper-Limb Exoskeleton Control", 6(12), 2050; https://doi.org/10.3390/s16122050

[43] Pfurtscheller G, Aranibar A., 1979: "Evaluation of event-related desynchronization (ERD) preceding and following voluntary self-paced movement", Electroencephalogr Clin Neurophysiol. Feb;46(2):138-46.

[44] Grosse-Wentrup M, Liefhold C, Gramann K, Buss M., (2009), Beamforming in non-invasive brain–computer interfaces. IEEETrans Biomed Eng 56(4):1209–1219

[45] Kai Keng Ang, ZhengYang Chin and et al., 2012: "Filter bank common spatial pattern algorithm on BCI competition IV Datasets 2a and 2b", Front. Neurosci., 29 March**, https://doi.org/10.3389/fnins.2012.00039**

[46] Dai M. et al., (2019), EEG Classification of Motor Imagery Using a Novel Deep Learning Framework, Sensors, 19, 551, pp.1-16.

APPENDICES

APPENDİX A. Preparation of EEG Data Sets in MATLAB

A1. Preparation of EEG Data Sets for First Method

A2. Preparation of EEG Data Sets for Second Method

APPENDİX B. Traing and Testing Algorithms of DNN in Python Language

B1. Traing and Testing Algorithms

B2. Loading MATLAB EEG Data from the file Directories into Python Program

APPENDİX A. Preparation of EEG Data Sets in MATLAB

The GDF files can be loaded using the open-source toolbox BioSig, available for free at http://biosig.sourceforge.net/. There are versions for MATLAB as well as a library for C/C++. A GDF file can be loaded with the BioSig toolbox with the following command in Octave/MATLAB (for C/C++, the corresponding function HDRTYPE* sopen and size t sread must be called):

```
[s, h] = sload('B0101T.gdf');
```

A1. Preparation of EEG Data Sets for First Method

```
clear all
close all

fparams.filterFreq   = [6,40];
fparams.filterType   = 'butter';
fparams.filterDegree = 5;
fparams.fs           = 250;

Cha = {'EEG:C3','EEG:Cz','EEG:C4', 'EOG:ch01', 'EOG:ch02', 'EOG:ch03'};

k1=0;
k2=0;
l1=0;
l2=0;

%--------- Reading EEG GDF File
EPOCHS1x = parseEpochsIIII2a('B0101T', Cha) ;
KK           = numel(EPOCHS1x.EPDT);
EPOCHS1  = filterEpochs(EPOCHS1x,fparams);
%number of samples
TT           = size(EPOCHS1.EPDT{1},2);
%number of channels
NN           = size(EPOCHS1.EPDT{1},1);

KK1 = floor(KK*0.8);

%----------- Save EEG Data for testing set
uu1 = 1;
for k=1:1:KK1
   DD = EPOCHS1.EPDT{k};
   EPOCHST.EPDT{uu1} = DD(1:NN, 1:750);
   EPOCHST.EPLB(uu1) = EPOCHS1.EPLB(k);
   uu1 = uu1 + 1;
```

```
end

for k=1:1:uu1-1
  CC = EPOCHST.EPDT{k};
  if (EPOCHST.EPLB(k)==1)
    DD =CC(1:NN,:);
    file1x                =['C:\Users\ITU\Documents\CSP\BB\S2008HTra21\S1\ATr_'
num2str(k1,'%03d') '.mat' ];
    save (file1x, 'DD');
    k1 = k1 + 1;
  end
  if (EPOCHST.EPLB(k)==2)
    DD =CC(1:NN,:);

file2x=['C:\Users\ITU\Documents\CSP\BB\S2008THTra21\S2\BTr_'um2str(k2,'%03
d') '.mat' ];
    save (file2x, 'DD');
    k2 = k2 + 1;
  end
  k
end
%-----------------------------------------------

%----------- Save EEG Data for Testing set
uu1 = 1;
for k=KK1:1:KK
   DD = EPOCHS1.EPDT{k};
   EPOCHSE.EPDT{uu1} = DD(1:NN, 1:750);
   EPOCHSE.EPLB(uu1) = EPOCHS1.EPLB(k);
   uu1 = uu1 + 1;
end

for k=1:1:uu1-1
  CC = EPOCHSE.EPDT{k};
  if (EPOCHSE.EPLB(k)==1)
    DD =CC(1:NN,:);
    file1x                =['C:\Users\ITU\Documents\CSP\BB\S2008HTes21\S1\ATe_'
num2str(l1,'%03d') '.mat' ];
    save (file1x, 'DD');
    l1 = l1 + 1;
  end
  if (EPOCHSE.EPLB(k)==2)
    DD =CC(1:NN,:);
    file2x                =['C:\Users\ITU\Documents\CSP\BB\S2008HTes21\S2\BTe_'
num2str(l2,'%03d') '.mat' ];
    save (file2x, 'DD');
    l2 = l2 + 1;
  end
  k
end
```

```
k1
k2
l1
l2

%-------------------------------------------------------------------------------------------------
```

A2. Preparation of EEG Data Sets for Second Method

```
clear all
close all

fparams.filterFreq   = [6,36];      %[6,15];
fparams.filterType   = 'butter';
fparams.filterDegree = 5;           %7
fparams.fs           = 250;

Cha = {'EEG:C3','EEG:Cz','EEG:C4', 'EOG:ch01', 'EOG:ch02', 'EOG:ch03'};
uu1 = 1;
uu2 = 1;

k1=0;k2=0;k3=0;k4=0;
l1=0;l2=0;l3=0;l4=0;

% ---------- Reading EEG GDF file

EPOCHS1x = parseEpochsIIII2a('B0101T', Cha) ;
KK       = numel(EPOCHS1x.EPDT);
EPOCHS1  = filterEpochs(EPOCHS1x,fparams);
TT       = size(EPOCHS1.EPDT{1},2);
NN       = size(EPOCHS1.EPDT{1},1);

KK1 = floor(KK*0.8);
for k=1:1:KK1
   DD = EPOCHS1.EPDT{k};
   EPOCHST.EPDT{uu1} = DD;
   EPOCHST.EPLB(uu1) = EPOCHS1.EPLB(k);
   uu1 = uu1 + 1;
end

for k=KK1:1:KK
   DD = EPOCHS1.EPDT{k};
   EPOCHSE.EPDT{uu2} = DD;
   EPOCHSE.EPLB(uu2) = EPOCHS1.EPLB(k);
   uu2 = uu2 + 1;
end

% ---------- Reading EEG GDB file

EPOCHS2x = parseEpochsIIII2a('B0102T', Cha) ;
KK       = numel(EPOCHS2x.EPDT);
```

```
EPOCHS2  = filterEpochs(EPOCHS2x,fparams);
TT       = size(EPOCHS2.EPDT{1},2);
NN       = size(EPOCHS2.EPDT{1},1);

KK2 = floor(KK*0.8);
for k=1:1:KK2
   DD = EPOCHS2.EPDT{k};
   EPOCHST.EPDT{uu1} = DD;
   EPOCHST.EPLB(uu1) = EPOCHS2.EPLB(k);
   uu1 = uu1 + 1;
end

for k=KK2:1:KK
   DD = EPOCHS2.EPDT{k};
   EPOCHSE.EPDT{uu2} = DD;
   EPOCHSE.EPLB(uu2) = EPOCHS2.EPLB(k);
   uu2 = uu2 + 1;
end

%------------ Filtering Process
fparams.filterType      = 'butter';
fparams.filterDegree    = 5;
fparams.fs              = 250;
%fparams.showFilter      = 'showFilter';

fparams.filterFreq=[6,12];
A1  = filterEpochs(EPOCHST,fparams);
fparams.filterFreq=[12,18];
A2  = filterEpochs(EPOCHST,fparams);
fparams.filterFreq=[18,24];
A3  = filterEpochs(EPOCHST,fparams);
fparams.filterFreq=[24,30];
A4  = filterEpochs(EPOCHST,fparams);
fparams.filterFreq=[30,36];
A5  = filterEpochs(EPOCHST,fparams);

KK       = numel(EPOCHST.EPDT);
uu1=1;
for k=1:1:KK
   DD = [A1.EPDT{k}' A2.EPDT{k}' A3.EPDT{k}' A4.EPDT{k}' A5.EPDT{k}' ];
   EPOCHSX.EPDT{uu1} = DD(:, 1:750);
   EPOCHSX.EPLB(uu1) = EPOCHST.EPLB(k);
   uu1 = uu1 + 1;
end

%------------ Filtering Process
fparams.filterFreq=[6,12];
B1  = filterEpochs(EPOCHSE,fparams);
fparams.filterFreq=[12,18];
B2  = filterEpochs(EPOCHSE,fparams);
```

```
fparams.filterFreq=[18,24];
B3  = filterEpochs(EPOCHSE,fparams);
fparams.filterFreq=[24,30];
B4  = filterEpochs(EPOCHSE,fparams);
fparams.filterFreq=[30,36];
B5  = filterEpochs(EPOCHSE,fparams);

KK      = numel(EPOCHSE.EPDT);

uu1=1;
for k=1:1:KK
   DD = [B1.EPDT{k}' B2.EPDT{k}' B3.EPDT{k}' B4.EPDT{k}' B5.EPDT{k}' ];
   EPOCHSY.EPDT{uu1} = DD(:, 1:750);
   EPOCHSY.EPLB(uu1) = EPOCHSE.EPLB(k);
   uu1 = uu1 + 1;
end
%----------------------------------------

%----------------- CSP Process
trainparams.m        = 5;
[WCSP,L]             = train_csp(EPOCHSX.EPDT, EPOCHSX.EPLB, trainparams);
testparams.classifier = 'LDA';
[LABELS,ZTR,ZTSS]              =   test_csp(EPOCHSY.EPDT,   EPOCHSX.EPDT,
EPOCHSX.EPLB, WCSP, testparams);
PERF                 = perfCalc(LABELS,EPOCHSY.EPLB)

%-------------------------------------------------------

uu1      = numel(EPOCHSX.EPDT);

%------------ Save EEG Data for training set
for k=1:1:uu1
  CC = ZTR{k};
  if (EPOCHSX.EPLB(k)==1)
     DD =CC(:,:);
     file1x              =['C:\Users\ITU\Documents\CSP\BB\S2008FCSPTra21\S1\ATr_'
num2str(k1,'%03d') '.mat' ];
     save (file1x, 'DD');
     k1 = k1 + 1;
  end
  if (EPOCHSX.EPLB(k)==2)
     DD =CC(:,:);
     file2x              =['C:\Users\ITU\Documents\CSP\BB\S2008FCSPTra21\S2\BTr_'
num2str(k2,'%03d') '.mat' ];
     save (file2x, 'DD');
     k2 = k2 + 1;
  end
  k
end
uu1      = numel(EPOCHSY.EPDT);
```

```
%------------ Save EEG Data for testing set
for k=1:1:uu1
  CC = ZTSS{k};
  if (EPOCHSY.EPLB(k)==1)
    DD =CC(:,:);
    file1x            =['C:\Users\ITU\Documents\CSP\BB\S2008FCSPTes21\S1\ETe_'
num2str(l1,'%03d') '.mat' ];
    save (file1x, 'DD');
    l1 = l1 + 1;
  end
  if (EPOCHSY.EPLB(k)==2)
    DD =CC(:,:);
    file2x            =['C:\Users\ITU\Documents\CSP\BB\S2008FCSPTes21\S2\FTe_'
num2str(l2,'%03d') '.mat' ];
    save (file2x, 'DD');
    l2 = l2 + 1;
  end
  k
end
```

APPENDİX B. Traing and Testing Algorithms of DNN in Python Language

B1. Traing and Testing Algorithms

```
import          tensorflow as tf
import          time
from            datetime import timedelta
import          math
import          random
import          numpy as np
import          os
import          dataset2


#Adding Seed so that random initialization is consistent
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)

#------------------------------------------------------------------------------------------

train_path      = './T2008FCSPTra/'
valid_path      = './T2008FCSPTes/'
batch_size1     = len(os.listdir(valid_path+'S1/')) + len(os.listdir(valid_path+'S2/'))
print ("batch_size:",batch_size1)

##Network graph params

filter_size_conv1  = 6
```

```
num_filters_conv1  = 40

filter_size_conv2  = 6
num_filters_conv2  = 40

filter_size_conv3  = 6
num_filters_conv3  = 40

filter_size_conv4  = 6
num_filters_conv4  = 40

fc_layer_size1     = 1024
keep_rate          = 0.8
keep_prob=tf.placeholder(tf.float32)
#------------------------------------------------------------------
batch_size         = 16
validation_size    = 0.2
img_size           = 750
classes            = os.listdir(train_path)
num_classes        = len(classes)

#------------------------------------------------------------------------------------------------
count1 = 0
count2 = 0

# We shall load all the training and validation images and labels into memory using openCV
data = dataset2.read_train_sets(train_path, valid_path, img_size, classes, validation_size=validation_size)

print("Complete reading input data. Will Now print a snippet of it")
print("Number of files in Training-set:\t\t{}".format(len(data.train.labels)))
print("Number of files in Validation-set:\t{}".format(len(data.valid.labels)))


session     = tf.Session()
x           = tf.placeholder(tf.float32, shape=[None, img_size,num_channels], name='x')

y_true      = tf.placeholder(tf.float32, shape=[None, num_classes], name='y_true')
y_true_cls  = tf.argmax(y_true, dimension=1, name ='y_true_cls')

is_training = tf.placeholder(tf.bool , name ='is_training')

def create_weights(shape):
   return tf.Variable(tf.truncated_normal(shape, stddev=0.05))

def create_biases(size):
   return tf.Variable(tf.constant(0.05, shape=[size]))

def create_convolutional_layer(inputx,
```

```
            num_input_channels,
            conv_filter_size,
            num_filters, MP):

    global is_training, count1
    ## We shall define the weights that will be trained using create_weights function.
    weights = create_weights(shape=[conv_filter_size, num_input_channels,
num_filters])
    ## We create biases using the create_biases function. These are also trained.
    biases  = create_biases(num_filters)

    ## Creating the convolutional layer
    layer1  = tf.nn.conv1d(value = inputx,
                filters     = weights,
                stride      = 1,
                padding     = 'SAME')

    layer1  += biases

    BN          = tf.layers.batch_normalization(
       inputs   = layer1,
       training = is_training
    )

    count1 = count1 + 1
    ad     = "Layer-CNN" + str(count1)

    ## Output of pooling is fed to Relu which is the activation function for us.
    layer2     = tf.nn.relu(BN)

    print ("layer ------>",layer2.shape)
    return layer2

def create_flatten_layer(layer, NN):

    #We know that the shape of the layer will be [batch_size img_size img_size
num_channels]
    # But let's get it from the previous layer.
    layer_shape  = layer.get_shape()

    num_features = layer_shape[1:4].num_elements()

    ## Now, we Flatten the layer so we shall have to reshape to num_features
    if (NN==1):
      layer3 = tf.reshape(layer, [-1, num_features], name = "layer_fc2")

    if (NN==2):
      layer3 = tf.reshape(layer, [-1, num_features], name = "layer_flat")

    print ("layer-flat ------>",layer3.shape)
```

```
    return layer3

def create_fc_layer(inputx,
            num_inputs,
            num_outputs,
            use_relu):

    global count2

    #Let's define trainable weights and biases.
    weights  = create_weights(shape=[num_inputs, num_outputs])
    biases   = create_biases(num_outputs)

    layer1   = tf.matmul(inputx, weights) + biases

    count2   = count2 + 1
    ad       = "Layer-FNN" + str(count2)

    if use_relu==0:
       layer3 = tf.nn.relu(layer1, name = ad)
    if use_relu==1:
       layer3 = tf.nn.tanh(layer1, name = ad)
    if use_relu==2:
       layer3 = tf.sigmoid(layer1, name = ad)

    print ("layer-fc ------>",layer3.shape)
    return layer3

layer_conv1 = create_convolutional_layer(inputx= x,
            num_input_channels=                  num_channels,
            conv_filter_size  =              filter_size_conv1,
            num_filters       =              num_filters_conv1,
            MP                = 0)

layer_conv2 = create_convolutional_layer(inputx= layer_conv1,
            num_input_channels=                  num_filters_conv1,
            conv_filter_size  =              filter_size_conv2,
            num_filters       =              num_filters_conv2,
            MP                = 0)

layer_conv3 = create_convolutional_layer(inputx= layer_conv2,
            num_input_channels=                  num_filters_conv2,
            conv_filter_size  =              filter_size_conv3,
            num_filters       =              num_filters_conv3,
            MP                = 0)

layer_conv4 = create_convolutional_layer(inputx= layer_conv3,
            num_input_channels=                  num_filters_conv3,
            conv_filter_size  =              filter_size_conv4,
            num_filters       =              num_filters_conv4,
```

```
            MP              = 0)
#--------

layer_flat   = create_flatten_layer(layer_conv4, 2)
print ("layer_flat=",layer_flat.shape)

layer_fc1x   = tf.nn.dropout(layer_flat,keep_rate)
layer_fc1    = create_fc_layer(inputx=                    layer_fc1x,
                num_inputs     =
          layer_flat.get_shape()[1:4].num_elements(),
                num_outputs    =                    fc_layer_size1,
                use_relu       =                    0)
#--------

layer_fc1y   = tf.nn.dropout(layer_fc1,keep_rate)
layer_fc2    = create_fc_layer(inputx=                    layer_fc1y,
                num_inputs     =                fc_layer_size1,
                num_outputs    =                    num_classes,
                use_relu       =                    0)

#---------------------------------------------------------
y_pred       = tf.nn.softmax(layer_fc2, name='y_pred')
y_pred_cls   = tf.argmax(y_pred, dimension=1 ,name='y_pred_cls')

cross_entropy       =       tf.nn.softmax_cross_entropy_with_logits(logits=layer_fc2,
labels=y_true, name='cross_entropy')

cost        = tf.reduce_mean(tf.square(layer_fc2 - y_true), name='cost')

optimizer               =   tf.train.AdamOptimizer(learning_rate=0.0001,   name   =
'optimizer').minimize(cost)

correct_prediction = tf.equal(y_pred_cls, y_true_cls, name='correct_prediction')
accuracy                =  tf.reduce_mean(tf.cast(correct_prediction,  tf.float32),name
='accuracy')

dogru           = tf.reduce_mean(tf.square(layer_fc2 - y_true),name ="dogru")

session.run(tf.global_variables_initializer())

def  show_progress(epoch,  feed_dict_train,  feed_dict_validate,  val_loss,x_batch,
y_true_batch, x_valid_batch,y_valid_batch):
    global val_acc,acc

    acc = session.run(accuracy, feed_dict=feed_dict_train)
    acc1 = session.run(dogru, feed_dict=feed_dict_train)
    acc2 = session.run(dogru, feed_dict=feed_dict_validate)
    val_acc = session.run(accuracy, feed_dict=feed_dict_validate)

    print ("Epoch:",epoch + 1, "acc:", acc, "val_acc:",val_acc, "val_loss:",val_loss)
```

```
    print ("Egitim-Hatasi   :", acc1, "Test-Hatasi   :",acc2)

total_iterations = 0
saver = tf.train.Saver()


def train(num_iteration):
    global total_iterations

    for i in range(total_iterations, total_iterations + num_iteration):

        x_batch, y_true_batch, _, cls_batch = data.train.next_batch(batch_size)
        x_valid_batch, y_valid_batch, _, valid_cls_batch = data.valid.next_batch(batch_size1)

        feed_dict_tr  = {x: x_batch, y_true: y_true_batch,is_training:True}
        feed_dict_val = {x: x_valid_batch, y_true: y_valid_batch,is_training:False}

        extra_update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)

        session.run([optimizer, extra_update_ops],feed_dict=feed_dict_tr)

        if i % int(data.train.num_examples/batch_size) == 0:
            val_loss = session.run(cost, feed_dict=feed_dict_val)
            epoch = int(i / int(data.train.num_examples/batch_size))
            show_progress(epoch, feed_dict_tr, feed_dict_val, val_loss,x_batch, y_true_batch, x_valid_batch,y_valid_batch)

            saver.save(session, './EEG-model')

    total_iterations += num_iteration

train(num_iteration=24000)
```

B2. Loading MATLAB EEG Data from the file Directories into Python Program

```
import      cv2
import      os
import      glob
from        sklearn.utils import shuffle
import      numpy as np
import      scipy.io as sio

on    = 750

def load_train(train_path, image_size, classes):
    global image
    images = []
    labels = []
    img_names = []
    cls = []
```

```
    image = np.zeros(on).astype(np.float64)

    print('Going to read training images')
    for fields in classes:
        index = classes.index(fields)
        print('Now going to read {} files (Index: {})'.format(fields, index))
        path = os.path.join(train_path, fields, '*')
        files = glob.glob(path)
        for fl in files:
#-----
            Buf1 = sio.loadmat(fl)
            Buf2 = np.transpose(Buf1['DD'])
#-----
            images.append(Buf2.copy())
            label = np.zeros(len(classes))
            label[index] = 255.0
            labels.append(label.copy())
            flbase = os.path.basename(fl)
            img_names.append(flbase)
            cls.append(fields)
    images = np.array(images)
    labels = np.array(labels)
    img_names = np.array(img_names)
    cls = np.array(cls)

    return images, labels, img_names, cls


class DataSet(object):

  def __init__(self, images, labels, img_names, cls):
    self._num_examples = images.shape[0]

    self._images = images
    self._labels = labels
    self._img_names = img_names
    self._cls = cls
    self._epochs_done = 0
    self._index_in_epoch = 0

  @property
  def images(self):
    return self._images

  @property
  def labels(self):
    return self._labels

  @property
```

```
  def img_names(self):
    return self._img_names

  @property
  def cls(self):
    return self._cls

  @property
  def num_examples(self):
    return self._num_examples

  @property
  def epochs_done(self):
    return self._epochs_done

  def next_batch(self, batch_size):
    """Return the next `batch_size` examples from this data set."""
    start = self._index_in_epoch
    self._index_in_epoch += batch_size

    if self._index_in_epoch > self._num_examples:
      # After each epoch we update this
      self._epochs_done += 1
      start = 0
      self._index_in_epoch = batch_size
      assert batch_size <= self._num_examples
    end = self._index_in_epoch

    return self._images[start:end], self._labels[start:end], self._img_names[start:end],
self._cls[start:end]

def read_train_sets(train_path, valid_path, image_size, classes, validation_size):
  class DataSets(object):
    pass
  data_sets = DataSets()

  images, labels, img_names, cls     = load_train(train_path, image_size, classes)
  images, labels, img_names, cls     = shuffle(images, labels, img_names, cls)

  Vimages, Vlabels, Vimg_names, Vcls = load_train(valid_path, image_size, classes)
  Vimages, Vlabels, Vimg_names, Vcls = shuffle(Vimages, Vlabels, Vimg_names,
Vcls)

  if isinstance(validation_size, float):
    validation_size = int(images.shape[0])

  validation_images    = Vimages
  validation_labels    = Vlabels
  validation_img_names       = Vimg_names
  validation_cls       = Vcls
```

```
  train_images                = images
  train_labels          = labels
  train_img_names       = img_names
  train_cls             = cls

  data_sets.train = DataSet(train_images, train_labels, train_img_names, train_cls)
  data_sets.valid       =       DataSet(validation_images,       validation_labels,
validation_img_names, validation_cls)

  return data_sets
```

CURRICULUM VITAE

Name Surname **:** Leyla Abilzade

Place and Date of Birth **:** Azerbaijan, 28 December 1993

E-Mail **:** suleymanli17@itu.edu.tr

EDUCATION

- **Bachelor Degree** **:**2011, Azerbaijan Technical University, Mobile Communication Pogram, Radio Engineering Telecommunication and Electronic Engineering