ON THE FEASIBILITY OF USING CONNECTIVITY MEASURES FOR DECODING BRAIN STATES

BY ANTHONY YANG

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Laleh Najafizadeh

and approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2018

ABSTRACT OF THE THESIS

On the Feasibility of Using Connectivity Measures for Decoding Brain States

by Anthony Yang

Thesis Director: Professor Laleh Najafizadeh

Optical brain imaging using functional near infrared spectroscopy (fNIRS) offers a non-invasive imaging tool for monitoring brain activity. fNIRS is a safe imaging technique offering high temporal resolution, making it an attractive choice for brain-computer interfaces, as well as for real-time and long-term monitoring of brain function. While often lauded for its portability, the application of fNIRS has been mostly limited to laboratory environments.

In order to improve the portability and wearability of fNIRS systems, it would be desirable to reduce the number of optodes, without compromising the performance. Using a large of number of optodes, while resulting in better coverage, increases the setup time, causes added discomfort, and hence, reduces the duration of wear time.

This work focuses on the application of fNIRS systems for decoding brain states and brain computer interfaces, and considers the problem of two-class classification in the case where the number of optodes are limited to those covering only the prefrontal cortex. Pairwise functional connectivity for various time windows have been computed as features. Extensive classification experiments have been performed using Support Vector Machines (SVM) and Convolutional Neural Networks (CNN) as classifiers. Results obtained from 8 subjects' data, indicate that it is feasible to create predictive models based on fNIRS-based functional connectivity measures of prefrontal cortex.

Acknowledgements

I would like to give my sincerest thanks to all of those who have contributed their advice and supported me during my academic journey.

First off, I cannot fully express my gratitude to my advisor, Dr. Laleh Najafizadeh, for all of the time and effort that she has put into my research. Her encouragement and direction was invaluable in helping me navigate the challenges of this study. I could not have asked for a more patient and understanding mentor.

I also give my thanks to my Thesis defense committee members, Dr. Mehdi Javanmard and Dr. Saman Zonouz, for contributing their time and insight to review my work.

I would also like to thank Tianjiao Zeng for teaching me about fNIRS and allowing me to use the data she collected, as well as Li Zhu for lending his advice and time to help me with training classifiers. They are both fantastic researchers and I am grateful that I had the opportunity to collaborate with them.

Additional thanks goes to Dr. Kristin Dana for being an excellent machine learning instructor, and Dr. Zoran Gajic for his guidance as the Graduate Program Director of the Rutgers University ECE department.

Most of all, I would like to express my appreciation for my parents, family, friends, and girlfriend, who have always motivated me to pursue my dreams and supported me through my entire life. This work would not have been possible without them.

Table of Contents

Chapter 1

Introduction

The goal of this study is to develop a functional near-infrared spectroscopy (fNIRS) classifier based on functional connectivity in the prefrontal cortex of the brain. Many fNIRS classification tasks involve imaging all or most of the brain, a task which requires many optodes and results in a long setup time. A reliable method involving fewer fNIRS optodes would reduce this setup time, as well as increase wearer comfort, which in turn increases the wear duration. Relying on fewer optodes also reduces the chance of introducing artifacts along the optical-tissue interfaces. Some fNIRS studies have shown promising results with narrowing down the number of optodes to just those that are needed to map the prefrontal cortex. However, no study has yet investigated if functional connectivity in the prefrontal cortex can be used in reduced-optode fNIRS classification. In addition, this study examines if deep learning is suitable for this classification task, by comapring support vector machine (SVM) and convolutional neural networks (CNN). To the best of the author's knowledge, no work as been done involving CNN for fNIRS functional connectivity.

fNIRS utilizes near-infrared light to measure activity in the brain. This technology is relatively recent compared to the more established electroencephalography (EEG) and functional magnetic resonance imaging (fMRI) techniques. Compared to these two modalities, fNIRS has had a rise in popularity due to its spatial and temporal resolutions. However, the quantity of optodes required limits its potential for use in applications other than research.

One prominent application of fNIRS-based classifiers is in brain computer interfaces (BCI). A BCI requires a highly accurate classifier in order to be successful. A large number of studies have been done to investigate what classifier, such as SVM, is best

to use in BCI. However, few have discussed the use of CNNs with fNIRS-based BCI. The studies that do investigate using CNN have shown promising results that can meet or exceed rival classification techniques. This study will explore using CNN and deep learning for fNIRS BCI based on functional connectivity in the prefrontal cortex, and compare the results with SVM.

The second chapter contains background information on fNIRS and its fundamentals. The third chapter reviews machine learning topics, including feature extraction and classification methods with a focus on SVM and CNN. The fourth chapter examines other previous studies relating to fNIRS-based classifiers and BCI. The fifth chapter describes the experimental paradigm and data collection for this study, followed by a detail of the classification method and results in the sixth and seventh chapters. The eighth chapter contains a discussion about the results of the study and possible future work.

Chapter 2

Review of functional Near-Infrared Spectroscopy

2.1 Introduction

This chapter presents a brief overview of how brain activity is measured and monitored using fNIRS. The modified Beer-Lambert Law and its role in fNIRS is discussed. The chapter continues with a comparison between fNIRS and other commonly-used neuroimgaging techniques, followed by a commentary on how fNIRS would need to be improved in order to see use in more practical applications.

2.2 Monitoring Brain Activity Using Near Infrared Light

fNIRS is a non-invasive neuroimaging technique which measures brain activity using optical light in the near-infrared (NIR) range (700 to 1000 nm) [1]. An fNIRS device utilizes NIR light sources to send photons into the head. Photons in this range pass largely uninhibited through the skin and skull, but may be attenuated by absorption or scattering by certain substances, in particular oxygenated hemoglobin (HbO_2) and deoxygenated hemoglobin (HbR). Fig. 2.1 shows how the photons travel through the brain from a light source to a light detector. Knowing the absorption coefficients of HbO_2 and HbR, we can estimate their concentrations using the modified Beer-Lambert law [1]. However, because the photon path length through the brain is unknown, the absolute value of the hemoglobin concentrations cannot be found and only the change in hemoglobin concentration from the baseline can be calculated. Change in oxygenated hemoglobin concentration is notated as ($\Delta[HbO_2]$) while change in deoxygentated hemoglobin concentration is notated as ($\Delta[HbR]$).

$\Delta[HbO_2]$ and $\Delta[HbR]$ are used in determining changes in brain activity. When

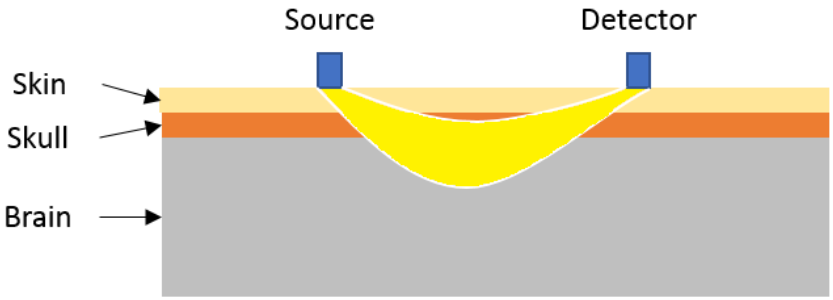

Figure 2.1: This illustration shows an example of the path of photons, shown in yellow, traveling from a light source to a light detector. The photons take a curved path through the brain, passing through the skin and skull before being absorbed by hemoglobin inside the brain.

neurons activate, they take in oxygen from the bloodstream. When brain activity increases, local oxygenation levels decrease at first due to increased oxygen consumption [1]. Shortly afterwards, the body reacts to increase cerebral blood flow (CBF), resulting in a net increase of oxygen concentration. Since most of the oxygen in the bloodstream is carried by hemoglobin, mapping the $\Delta[HbO_2]$ and $\Delta[HbR]$ concentrations in regions of the brain can show what parts of the brain are being activated. Relationships can then be drawn between stimuli and the activation of regions of the brain.

Measurement of light attenuation is performed by emitting photons from a source at a known intensity and measuring the received photon intensity at the detector. The light sources and detectors are referred to as optodes and are placed in a configuration on the head known as a montage. The light attenuation measured by the optodes is used to calculate the concentrations of the substances in the photon path using the modified Beer-Lambert Law, as shown in (2.1) [1].

$$A = \ln \frac{I_0}{I} = \varepsilon \times c \times d \times B + G \tag{2.1}$$

In (2.1) A is the measured attenuation, I_0 is the initial light intensity, I is the measured light intensity at the detector, ε is the specific extinction coefficient of the substance, c is the substance concentration, d is the distance from source to detector, B is the differential path length factor (DPF), and G is the loss due to light scattering.

ε is determined by the substance that is being measured (i.e. HbO_2 or HbR) and the wavelength of light that is being used. In the original Beer-Lambert law, the photon path is a straight line, and therefore the path length was equal to d. However, with fNIRS, the photons take a curved path to the detector, and thus the term B is introduced to calculate the curved path length $d \times B$. If G is assumed to be constant, then the change in substance concentration (Δc) can be expressed as (2.2).

$$\Delta A = \varepsilon \times \Delta c \times d \times B \tag{2.2}$$

In the form of the modified Beer-Lambert Law in (2.1), the term G drops out. B can be set as some estimated path length, which allows Δc to be calculated.

2.3 Comparison of fNIRS with other Techniques

In addition to fNIRS, there are other well established methods for mapping brain function. Among them, two commonly-used techniques are EEG and fMRI. EEG measures brain activity using electrical signals, and fMRI measures brain activations through the blood oxygen level-dependent (BOLD) response [2]. There are a number of advantages and disadvantages of fNIRS compared with EEG and fMRI. Between fNIRS and EEG, EEG has a significantly higher temporal resolution, but fNIRS has a better spatial resolution and is immune to electromagnetic interference [3]. Comparing fNIRS and fMRI, fNIRS can attain temporal resolution on the order of 40 ms versus fMRI with around 2000 ms, but fMRI yields better spatial resolution and coverage [3], as well as higher SNR [4].

2.4 Improving Usability of fNIRS

The advantages of fNIRS make it a useful tool, but currently it generally sees application in laboratory settings where the environment is closely controlled. Enabling fNIRS to be used for in more practical situations would require improvement of the headgear used in fNIRS. For example, NASA has said that it would like to use fNIRS for sensing the mental state of pilots, but several challenges currently prevent this

from being possible [5]. The largest issues include long setup time due to the required quantity of optodes, introduction of signal artifacts at the optical-tissue interface [6], and comfort to the wearer over long periods of time. One innovation NASA suggests is to use a headband with small comblettes with several optodes attached that part hair more easily than single optodes to facilitate rapid setup. Another study shows that using optodes with multiple fibers, called brush optodes, may be more effective and easier for setup than regular optodes [7].

A different possible approach to improving fNIRS equipment would be to reduce the number of optodes used by measuring only certain areas of the head. The easiest location on the head to measure with fNIRS is the prefrontal cortex where there is less hair. If prefrontal cortex signals alone are enough for classification, this would decrease the setup time significantly, while keeping the most reliable optode placement locations. However, it needs to be determined whether the fNIRS measurement of only the prefrontal cortex is sufficient to generate useful data that is capable of differentiating between various brain states. This study aims to address this question.

2.5 Conclusion

In summary, functional near-infrared spectroscopy is a promising tool for measuring brain activation through changes in concentrations of oxygenated and deoxygenated hemoglobin. A comparison of fNIRS with EEG and fMRI find that the different modailities each have their own characteristics, making them all valuable for differing reasons. While fNIRS has much utility already as it is, some possible improvements would allow fNIRS to be used outside of a lab setting for places like NASA.

Chapter 3

Machine Learning

3.1 Introduction

In this chapter, some of the basics of machine learning methods and techniques are presented. Feature extraction is discussed, followed by a general review of the classification problem and the techniques used to overcome the common problem of overfitting. Then, two classifiers are introduced, support vector machine and convolutional neural networks. Details about convolutional neural network layers, training, testing, and transfer learning with pretrained networks are given. The chapter closes with an examination of the metrics used to evaluate the performance of classifiers.

3.2 Review of Machine Learning

Machine learning is a technique used to train classification functions, or classifiers. Classifiers are mathematical models which place unlabeled data into classes. To do so, the classifier must learn the classes beforehand from a set of training data. Modern computers are capable of learning from enormous datasets to acquire the best fitting model. Building a classifier involves several steps. First, specific features may be extracted from the data depending on if the input data contains more information than needed. Then once the features are prepared, the dataset split into training and testing sets. It is crucial that the testing set does not contain any data samples from the training set to ensure that the performance is evaluated correctly. Finally, a classifier is learned from the training set, and evaluated using the testing set. Each step has a significant impact on the final result, and the designer chooses how to carry out each stage.

3.3 Feature Extraction

Input data often contains irrelevant or redundant data which can slow down the training of a classifier, or have a negative impact on its performance. The solution to this is to reduce the dimensionality of the data by extracting only features important to the classification problem. Some features that have been found to be useful for BCI include correlation coefficient, Principal Component Analysis (PCA), and Independent Component Analysis (ICA) [8]. No one feature has yet been found to be suitable for every type of data.

Although feature extraction is often helpful, removing too much information also tends to reduce the performance of the classifier. Because of this, some classifiers do not use any feature extraction and instead use just the raw input data. This is typically only the case with deep learning methods, where features are shaped simply through training the neural network without the need to manually select them. As a result, using raw data with a neural network such as a convolutional neural network, can meet or exceed the performance of handpicked features [9].

3.4 Classification Methods

In machine learning, there is supervised learning and unsupervised learning. In supervised learning, the classifier is trained with known labels. These labels are used to create a function which separates the labels into classes within the feature space. Examples of supervised learning methods include SVM and CNN, which will be discussed in more detail below. Unsupervised learning does not train with known labels, and instead the learner organizes and groups the data by similarities in the feature space. An example of this is K-means clustering.

Since the goal of classification is create a model which fits the data as accurately as possible, two possible problems with classifiers is underfitting and overfitting. In the underfitting case, the model is not a high enough order to accurately describe the boundary between classes. Overfitting is the opposite, where the model is too complex and is thus intolerant to noise. Both scenarios result in poor classification results. If

the classifier is underfitting the data, then a more complex model should be used, for example, a nonlinear classifier rather than a linear one. Overfitting, on the other hand, is often caused by an insufficient quantity of training data.

Data augmentation and k-fold cross-validation are two common methods used to remedy overfitting [10]. Data augmentation involves synthesizing data from the original dataset to increase the overall size of the training data. Some methods of doing this include splitting original samples into multiple samples, adding jitter and noise, or data warping [11]. In k-fold cross validation, the entire dataset is divided into k equal or nearly equal subsets, called folds [12, 13]. One fold is selected as a testing set, while the remaining $k-1$ folds are used as the training set. By using each of the k folds as the testing set once, k classifiers can be trained. Then, the accuracy for a classifier trained on the entire dataset can be estimated as the average accuracy of the k classifiers. This allows the testing set to be used in training and increase the size of the training data. k-fold cross validation is shown in Fig. 3.1

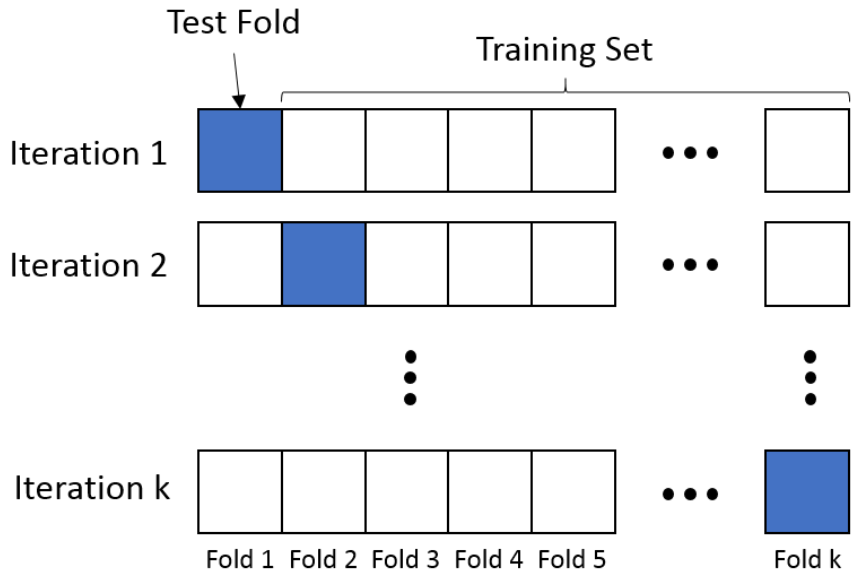

Figure 3.1: This figure shows how k-fold cross validation is applied to a dataset. The dataset is split into k folds and one fold is selected as the test fold which is highlighted in blue. Each iteration selects a different fold as the test fold, while the remaining folds form the training set.

3.4.1 Support Vector Machine

Support Vector Machine is a binary classifier that has become popular because it is relatively simple and fast while also yielding strong results. The algorithm seeks to find the optimal hyperplane boundary separating two classes of data [14, 15]. First, the training data is mapped from input space into a higher-dimensional feature space. Next, a small amount of training data near the boundary between the classes is selected as the support vectors. Then, the optimal hyperplane is found by minimizing the distance between the hyperplane and the support vectors. This is done by minimizing the sum of the dot products between the support vectors and the hyperplane vector. The hyperplane can then be mapped back from feature space into input space. SVM then uses the hyperplane to label the test data samples in the feature space.

3.4.2 Convolutional Neural Networks

Convolutional neural networks are neural networks that rely on a type of layer called convolutional layers. The convolutional layers allow a CNN to extract features from raw data or generate higher level features in a deeper network architecture. This ability means that hand-picked features may not be necessary to create an effective classifier as in [9]. However, selecting features may still be useful, as in [16].

A neural network is a network structure that transforms an input through a series of hidden layers to generate an output prediction. Each hidden layer consists of neurons that may be connected to some or all of the neurons in the following and preceding layers. The first layer is an input layer which accepts the input data in a specific arrangement (e.g. a 32x32x3 matrix for a 32x32 image with 3 color channels). The final layer is the output layer, which is a fully-connected layer, meaning that it is connected with all of the neurons in the previous layer. This layer contains scores for each class, where the top score is the CNN's best prediction. The hidden layers typically consist of convolutional layers, pooling layers, rectified linear unit (ReLU) layers, and dropout layers. Each layer performs a different transformation on the data. An illustration of the layers in a neural network are shown in Fig. 3.2.

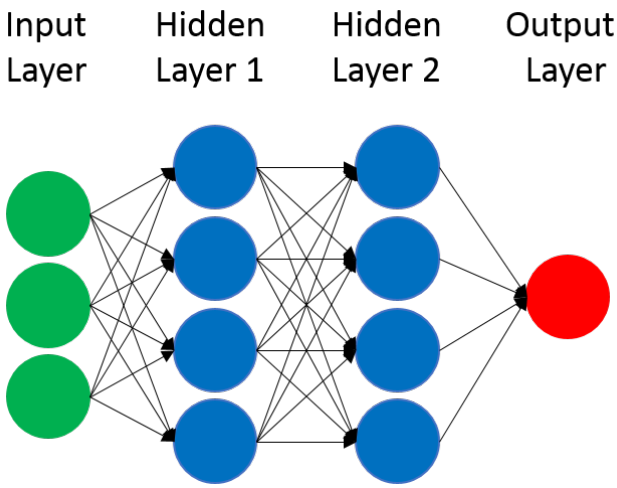

Figure 3.2: Visualization of the layers in a neural network. The input layer is the same dimensions as the input data. The output layer has one neuron for each class. The hidden layers may have any number of neurons. The neurons from each layer may be connected to one, many, or no neurons in the next layer. A network with N layers has $N-2$ hidden layers.

3.4.3 CNN Layers

The convolutional layer is a powerful tool which gives the CNN the ability to connect neurons locally in the data. In this layer, a convolution filter slides across the height and width of the input data. During each step, the filter convolves the data inside of it, through the entire depth, producing a connection to a neuron on the next layer [17]. This filter passes across entire extent of the data, which creates a two dimensional map of features from the input. The values in the filters are called weights, which are tuned as the CNN trains. The value of an element in the output layer $y_{i,j}$ is expressed mathematically in (3.1) for a convolutional layer, x, of size $H \times W$ using a $m \times n$ filter of weights ω. The resulting layer, y, is of size $(H-(m-1)) \times (W-(n-1))$.

$$y_{i,j} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} \omega_{ab} y_{(i+a),(j+b)} \tag{3.1}$$

Here, $0 \leq i < H$, $0 \leq j < W$. The result of performing convolution is shown in Fig. 3.3, which depicts an example of a 19×19 convolutional layer with a 5×5 filter creating a 15×15 resulting feature map. The strength of using convolution is that the neurons

are only locally connected instead of fully connected to every neuron in the next layer. This allows the CNN to save on computational power while preserving the most critical neuron connections. Multiple convolutional layers creates features which become more complex as the number of layers increases. Convolutional layers have proven to be so powerful that CNNs built primarily on just convolutional layers have shown state of the art performance [18].

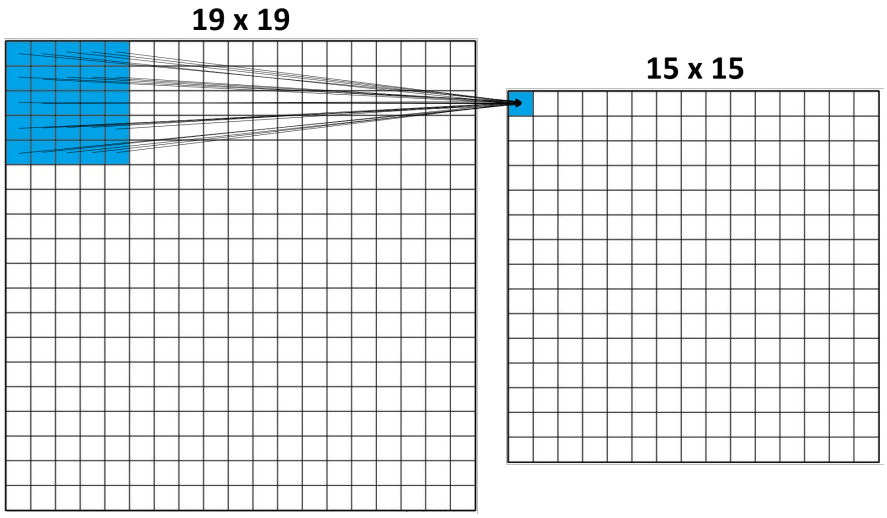

Figure 3.3: Illustration of the convolutional layer. The blue squares show the 5×5 convolutional filter convolving the input region on the left to create the feature map on the right. With this filter size, the 19×19 input becomes 15×15 at the output.

Typically, convolutional layers are complemented by pooling layers. Pooling is used to downsample data, which removes dimensionality and adds generality, helping to reduce overfitting. The operation takes a layer of size $H \times W$ and divides it into square pooling regions of size $k \times k$, resulting in an output of size $\frac{H}{k} \times \frac{W}{k}$. The most common type of pooling is max pooling, as shown in (3.2).

$$y_{i,j} = \max_{0 \leq m < k, 0 \leq n < k} x_{i \times k + m, j \times k + n} \tag{3.2}$$

In (3.2), $y_{i,j}$ is an element in downsampled layer after pooling, $0 \leq i < \frac{H}{k}$, $0 \leq j < \frac{W}{k}$ and x is the input layer. Max pooling is usually implemented using a 2×2 filter. This process divides the entire length and width of the data into 2×2 regions and only

preserves the max value in each region, resulting in an output that is one-quarter of the original size. This type of max pooling is shown in Fig. 3.4. Pooling layers are often placed after one or more convolutional layers.

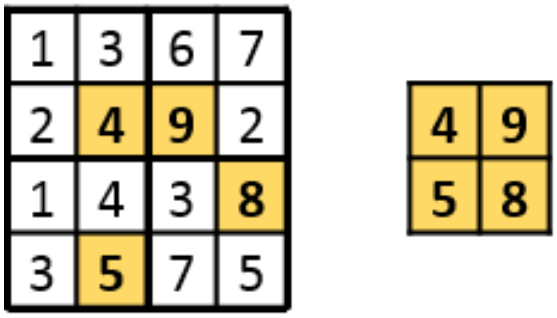

Figure 3.4: The figure shows 2×2 max pooling being performed on a 4×4 input layer. The highest value in each 2×2 section of the input is retained in the output, compressing the input to a 2×2 output.

Rectified linear unit (ReLU) layers apply an activation function to each element in the data. This is done after convolution to introduce some non-linearity into the system, which can improve classification results [19]. The most common function is

$$y_{i,j,d} = max\{0, x_{i,j,d}\} \tag{3.3}$$

where $0 \leq i < H$, $0 \leq j < W$, $0 \leq d < D$ for an input layer x and output layer y, both of which have height (H), width (W), and depth (D). There are other activation functions which may be used instead of $max(0, x)$, such as $tanh(x)$, but those are infrequently used since it is more rapid to compute $max(0, x)$ than the alternatives.

Dropout layers randomly drop neurons and their connections with a certain probability by setting the neuron to zero [20]. This is utilized during network training as a way to reduce the chance that the network will fit to noise, and thus reduce the possibility of overfitting. During testing, the dropout layers are removed from the network.

3.4.4 CNN Training and Testing

CNNs may use shallow architecture, with just a few layers, or a deep one with dozens of layers. Shallow architectures have fewer weights and train faster, while deeper architectures have more weights and take longer to train. As the CNN trains, the weights in its layers are tuned and optimized with the goal of minimizing the loss function. This function is calculated from the output of the last layer. The most commonly used loss function is the softmax function, shown in (3.4).

$$y_{i,j,k'} = \frac{e^{x_{i,j,k'}}}{\sum_k e^{x_{i,j,k}}} \tag{3.4}$$

The softmax function returns a set of probabilities that the input data belongs to each of the classes from a set of k' different classes. Softmax is typically combined with the negative log-likelihood function, as in (3.5),

$$y_{i,j} = -\log x_{i,j,c_{i,j}} \tag{3.5}$$

where $c_{i,j}$ is the index of the location of the correct class. This returns a single value describing the loss from the classification of a particular input data sample.

The CNN may train on the data for any number of passes, which are called epochs. Each epoch, the CNN tunes parameters to minimize the loss function by a rate defined by the learning rate. The learning rate is set by the designer and heavily impacts how successful the network is able to converge to the optimal weights. If the learning rate is too high, then the network may fail to converge, or it may not be able to tune precisely enough to find the lowest loss. If the learning rate is too low, then the network will take a long time to converge. The behavior of different learning rates is depicted in Fig. 3.5. The learning rate can be decayed during training to find a compromise between training time and minimization of the loss function. One well known technique to decay learning rate is using a method called Adam [21].

The more epochs the CNN trains for, the better tuned the weights will be. However, if the CNN trains for too many epochs, the network may overfit. Therefore, there is typically a dataset called the validation set which is used as the network trains

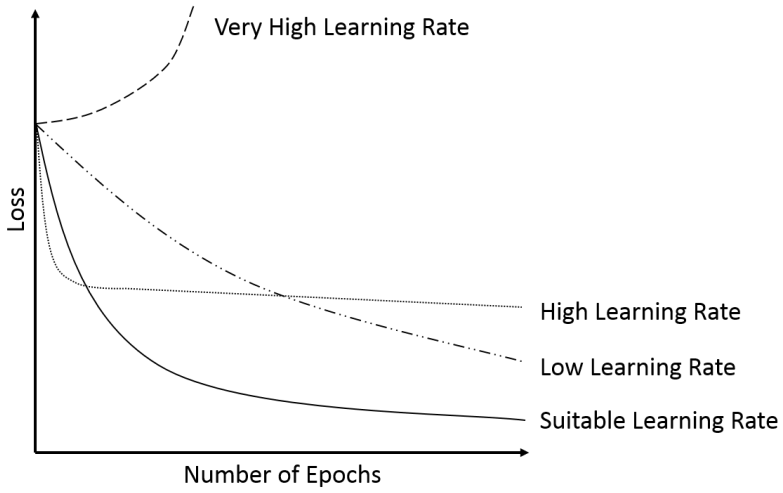

Figure 3.5: This sketch represents examples of losses from training a CNN with different learning rates. A suitable learning rate will converge to the minimum loss rapidly, while a low learning rate will approach the minimum loss more slowly, requiring more epochs than the suitable learning rate. A high learning rate will often settle at a loss which is not optimal, and a very high learning rate will fail to converge.

to determine when the network is starting to overfit and should stop training. The validation accuracy is compared to the training accuracy. A classifier with zero overfitting will have a validation accuracy equal to the training accuracy, but in practice it is expected that the validation accuracy will be slightly lower than the training accuracy. This is because the CNN is tuned only to the train dataset, so it should naturally perform better with the train dataset than the validation dataset. If the validation accuracy is much lower than the training accuracy, this is a sign of overfitting, as the CNN has become tuned too tightly to the training data to classify other data accurately. To avoid this, the network can be set to stop training once the validation accuracy no longer improves. An illustration of overfitting indicated by the training and validation accuracy is shown in Fig. 3.6.

3.4.5 Transfer Learning and Pretrained Networks

Creating a CNN from scratch can often be challenging due to insufficient quantities of training data. In this situation, the limited size of the training dataset often results

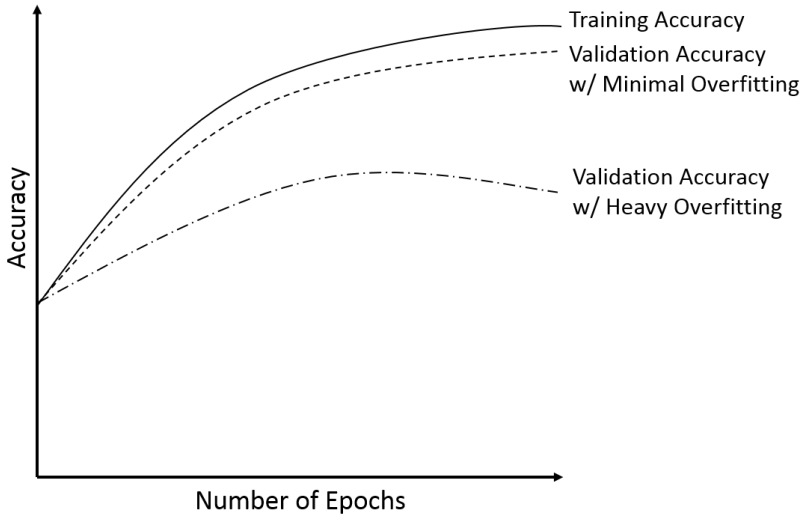

Figure 3.6: This sketch shows training and validation curves to illustrate the effect of overfitting on the validation accuracy. A classifier with minimal overfitting has a validation accuracy just slightly below that of the training accuracy. If the classifier has heavy overfitting, then the validation accuracy will be far below the training accuracy, and the validation accuracy may decrease with increased training.

in heavy overfitting. However, it has been found that it is possible to effectively apply features from a deep network trained on a large and separate dataset to a significantly different classification task, which is called transfer learning [22, 23, 24]. This method adapts the original pretrained network to predict the classes for the new classification task by altering the output layer. In addition, the input data is processed to match the input layer of the original network. The architecture of the pretrained network may altered by removing or adding layers, or it can be left as is. Then, the pretrained network is trained on the new dataset to fine-tune the network to the new task to create the new classifier.

There are many deep pretrained networks available to use for transfer learning, with varying accuracies and training speeds. Perhaps the most well known deep network is AlexNet which was created in 2012, and named after one of its creators, Alex Krizhevsky [25]. This network vastly outperformed its competition on the ImageNet dataset, and brought deep learning methods into the forefront of machine learning. Since then, many improvements have been made to AlexNet in networks such as VGG-F, which was

developed in 2013 [26]. VGG-F has a similar architecture and performance as AlexNet, but runs quicker due to the use of a large 4 pixel stride in the first convolutional layer, and smaller convolutional filter depth. Published in 2014, VGG Net 16 is another notable network which used 3x3 convolutional filters, smaller than the filters in AlexNet, which helped it perform better than AlexNet at the expense of speed [27]. The architecture of AlexNet, VGG-F, and VGG Net 16 are shown in Table 3.1.

Table 3.1: Popular Pretrained Convolutional Neural Networks

AlexNet (2012)	**VGG-F (2013)**	**VGG Net 16 (2014)**
8 weight layers	8 weight layers	16 weight layers
input (224 x 224 RGB Image)	input (224 x 224 RGB Image)	input(224 x 224 RGB Image)
conv11-96	conv11-64	conv3-64
maxpool	maxpool	conv3-64
		maxpool
conv5-256	conv5-256	conv3-128
maxpool	maxpool	conv3-128
		maxpool
conv3-384	conv3-256	conv3-256
conv3-384	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256
maxpool	maxpool	maxpool
		conv3-512
		conv3-512
		conv3-512
		maxpool
		conv3-512
		conv3-512
		conv3-512
		maxpool
fc-4096	fc-4096	fc-4096
fc-4096	fc-4096	fc-4096
fc-1000	fc-1000	fc-1000
soft-max	soft-max	soft-max

Notes: This table shows the network architecture of the AlexNet, VGG-F and VGG Net 16 networks. AlexNet and VGG-F both include five convolutional (conv) layers, three max pooling (maxpool) layers, and three fully connected (fc) layers. VGG Net 16 contains 13 convolutional layers, 5 max pooling layers, and 3 fully connected layers. The convolutional layer parameters are represented as "conv(filter size)-(number of filters)". Fully connected layers are similarly written as "fc-(number of filters)".

3.5 Performance Evaluation Metrics

As test data samples are labeled by the classifier, the result is compared with the actual label. There are four possible outcomes depending on whether the label is correct and what class is being assessed as the positive class. The four possibilities are true positive (tp), true negative (tn), false positive (fp), and false negative (fn). The quantity of each of the outcomes is used to determine four main performance evaluation metrics, which are accuracy, precision, sensitivity, and specificity [28, 29]. Accuracy is the most common metric and is found using (3.6).

$$Accuracy = \frac{tp + tn}{tp + fn + fp + tn} \tag{3.6}$$

Accuracy describes the overall performance of the classifier and is a useful metric for comparing different classifiers. However, other metrics offer some additional insight. Precision is another common metric, as in (3.7).

$$Precision = \frac{tp}{tp + fp} \tag{3.7}$$

Precision gives some input on how well the classifier selects the positive class. If precision is high, then a large number of the test samples the classifier deemed as positive were correctly labeled. This is a particularly useful metric when it is important to avoid false positives. Two more performance measures are sensitivity, also known as recall, and specificity.

$$Sensitivity = \frac{tp}{tp + fn} \tag{3.8}$$

$$Specificity = \frac{tn}{fp + tn} \tag{3.9}$$

Sensitivity, as in (3.8), measures the true positive rate of the classifier and describes how effective it is at correctly labeling the positive class. On the other hand, specificity, in (3.9), measures the true negative rate, which is how well the classifier labels the negative class. To elaborate, sensitivity is useful in measuring the how well the classifier avoids false negatives, while specificity can be helpful for measuring the avoidance of

false positives. Together, sensitivity and specificity and give a measure of bias in the classifier. A sensitivity higher than the specificity indicates that the classifier prefers to label samples as the positive class. The opposite would signify that the classifier prefers to label samples as a class other than the positive class. Combined together, the four values of accuracy, precision, sensitivity, and specificity give useful information into the characteristics of the classifier.

3.6 Conclusion

This chapter gave an introduction to machine learning and some of the techniques for feature extraction and classification. Two popular classifiers are explored, support vector machine and convolutional neural networks. Then, details are given on how to build and train a classifier, as well as how to evaluate performance using several measures.

Chapter 4

Previous Works

4.1 Introduction

This chapter examines recent research done on fNIRS functional connectivity as well as fNIRS-based classification work. Some methodologies used, feature extraction, and classification are briefly reviewed. In addition to fNIRS studies, a few EEG classification studies are also included in this chapter as an alternative neuroimaging technique. BCI studies comprise the majority of the research done on fNIRS and EEG classification.

4.2 Functional Connectivity Studies

Functional connectivity refers to the strong temporal correlations in signals of distinct regions of the brain [30, 31]. Measuring functional connectivity requires a method that examines the statistical dependence between time series data [32], the simplest of which is calculating the correlation coefficients [33]. There are a number of other measurements of functional connectivity, such as wavelet coherence and wavelet phase coherence as used in [30], and phase locking value in [34]. Functional connectivity can exhibit statistically significant changes during a cognitive task [33] making it potentially useful as a feature for classification.

4.3 Classification Studies with fNIRS

4.3.1 Data Collection and Feature Extraction

Much of the previous work done with classification on neuroimaging data is for the purpose of developing a BCI. Most fNIRS- and EEG-based BCI studies share a similar procedure for data acquisition and preprocessing before the classification task. First,

the data is collected in an experiment using fNIRS, EEG, or both technologies. Then, the data is filtered for artifacts from instrumental, experimental, and physiological noise using filtering and other signal processing methods. Depending on the task, the experiment, and the neuroimaging modality, the data can be analyzed in the time domain or in the frequency domain [16]. In fNIRS recordings, $\Delta[HbO_2]$ signals are more commonly used for classification as opposed to $\Delta[HbR]$ due to higher reliability and stability [35].

There are many features which can be extracted from fNIRS or EEG signals. Common features used in classification of fNIRS-based BCIs include mean, slope, variance, amplitude, skew, and kurtosis [35]. Some EEG studies use short windows from the raw time series data. In [9], 2 second wide crops are taken from 4.5 second EEG trials. The crops are taken spanning the entire trial so that one single trial produced 625 crops, each which was used in their classification dataset. Each crop is shifted by one sample from the last, so that they overlap, creating the most possible crops from one trial. This sliding time window technique generated a very large dataset from a much smaller one, which can help reduce overfitting, especially in a CNN. Another study used a slightly different method with non-overlapping time windows and 3 EEG frequency bands to create 3-channel spectral topography maps [36]. The time windows in this study were 0.5 seconds long, and each 3.5 second trial yielded 7 spectral topography maps.

4.3.2 Classification

Classification is most commonly performed using a supervised learning method, such as Linear Discriminant Analysis (LDA), SVM, and artificial neural networks (ANN). From 2004 to 2014, LDA was the most frequently used method, with use in 42.2% of studies, due to its agility and simple implementation, while SVM was the second most widely used, at 25.5%, for its high effectiveness [35]. ANNs have also become popular in recent years because of promising new results. One fNIRS study found that both SVM and ANN perform well compared to other popular methods [37]. In addition to ANNs, CNNs have recently found application for fNIRS-Based BCIs, with one study finding

that CNN improved classification accuracy over SVM and ANN by up to 6.49% and 3.33% respectively [38]. A summary of the results from these papers is given in Table 4.1. These papers show that it is feasible to achieve over 80% accuracy with CNN and SVM, with some papers reaching above 90%.

Overfitting is a common problem faced in fNIRS and EEG classification problems due to the difficulty in acquiring a large dataset. As a result, it is common for papers to implement k-fold cross validation to reduce the overfitting [39]. For CNN, another technique for reducing overfitting is to use transfer learning and tune an already pretrained network. One study used AlexNet to train a classifier for medical image analysis, which met or exceeded the performance of CNNs designed from scratch [40].

4.4 Conclusion

Much work has been done with fNIRS and EEG classification, typically for use in BCI. Functional connectivity has been found to be a useful statistical measure, but has not yet been explored for use as a feature in fNIRS classification. Techniques such as using a sliding time window can be used to generate a much larger dataset from a small one. A number of classifiers have shown promise for use with fNIRS and EEG data, including SVM and CNN. There are also a few known methods for reducing overfitting, including k-fold cross validation and CNN transfer learning.

Table 4.1: List of Previous fNIRS and EEG Classification Works

Source	**Feature Space**	**Classification Method**	**Modality**	**Average Accuracy %**
[16]	Fast Fourier Transform	Pretrained Stacking Restricted Boltzmann Machines	EEG	84
[37]	3-D combination of standard statistical features*	linear discriminant analysis	fNIRS	79.6
[37]	3-D combination of standard statistical features	quadratic discriminant analysis	fNIRS	95.2
[37]	3-D combination of standard statistical features	k-nearest neighbor	fNIRS	64.5
[37]	3-D combination of standard statistical features	naïve bayes	fNIRS	94.8
[37]	3-D combination of standard statistical features	support vector machine	fNIRS	95.2
[37]	3-D combination of standard statistical features	artificial neural networks	fNIRS	96.3
[9]	raw EEG, 2 second sliding windows	CNN with 5 weighted layers	EEG	92.40
[9]	raw EEG, 2 second sliding windows	CNN with 2 weighted layer	EEG	89.28
[36]	spectral topology map, single window	recurrent-CNN	EEG	87.61
[36]	spectral topology map, multiple windows	recurrent-CNN	EEG	91.11
[38]	standard statistical features with PCA	support vector machine	fNIRS	86.19
[38]	standard statistical features with PCA	artificial neural network with 2 weighted layers	fNIRS	89.79
[38]	standard statistical features with PCA	CNN with 1 weighted layer	fNIRS	93.08

*standard statistical features refers to the six statistical properties of mean, variance, kurtosis, skewness, peak and slope.

Chapter 5

Classification Results Based on Connectivity-Based Features

5.1 Introduction

In this chapter, results for using connectivity-based measures for decoding brain states are presented. First, the experimental task and data collection process are described, followed by a description of feature extraction procedure. Finally, classification results for SVM and CNN are presented.

5.2 Experimental Paradigm

This research utilizes data collected in an investigation performed by T. Zeng *et al.* described in [41]. The study used fNIRS to investigate if the somatosensory and prefrontal cortices of the brain show activation due to irritation. The paradigm used two solutions as the stimuli: irritant-contained and irritant-free. The experiment consisted of a total of ten blocks. During each block, the participant is given 10 mL of one of the solutions. The solution is held in the participant's mouth for 30 seconds while he or she remains still and focuses on a plus symbol displayed on a computer monitor. After the 30 second period, the solution is spit out. Following this is 2 minutes of rinsing with water, 1 minute of break, and 1 minute of rest before the next block begins, as shown in Fig. 5.1. Each experiment consisted of ten such trials of 30 second blocks followed by 4 minute intervals.

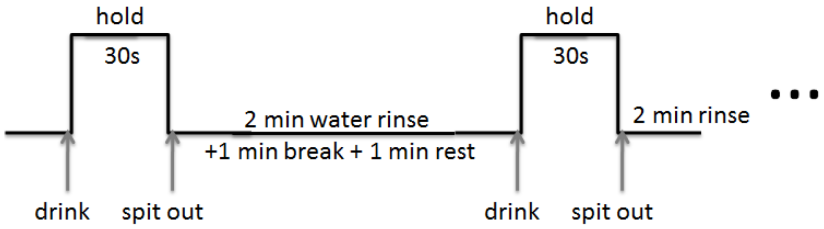

Figure 5.1: An illustration of the timing procedure for the experiment. Solutions are held in the mouth for 30 seconds in each block. The blocks are separated by 4 minute intervals. Image is used with permission by T. Zeng *et al.* [41].

Figure 5.2: The image shows the experimental setup used. The participant is seated in front of the computer monitor in a relaxed position. The solutions are given in a random order. Image is used with permission by T. Zeng *et al.* [41].

5.3 Data Collection

The experiment included 9 right-handed subjects (3 female, 6 male) who signed the informed consent forms approved by the Rutgers Institutional Review Board (IRB). The participants were seated comfortably in front of a computer. The experimental setup is shown in Fig. 5.2.

The system used to collect the fNIRS data was the NIRScout XP by NIRx Medical Technologies, LLC. A sampling rate of 7.81 Hz was used with wavelengths of 760 nm and 850 nm. Four sources and eight detectors were placed on the prefrontal cortex,

while six sources and eight detectors were placed on each side of the somatosensory cortex, totaling in 16 sources and 24 detectors. The optodes were arranged as shown in Fig. 5.3. The optode arrangement created 14 channels covering the prefrontal cortex, and 36 channels covering the somatosensory region, as shown in Fig. 5.4. Only the data from the 14 prefrontal cortex channels are used in this research.

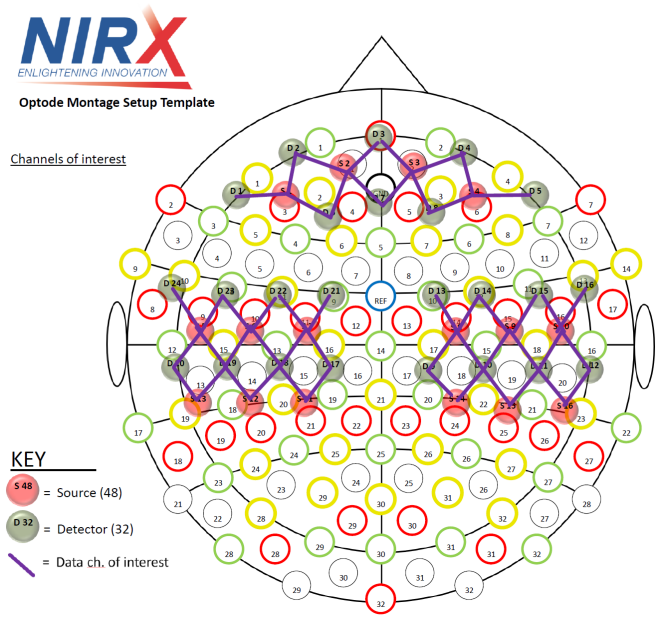

Figure 5.3: The fNIRS optode montage includes coverage of the prefrontal and somatosensory cortices. There are 4 sources and 8 detectors in the prefrontal cortex, giving 14 prefrontal channels. The somatosensory cortex contains 6 sources and 8 detectors per side, resulting in a 36 somatosensory channels. There are a total of 16 sources and 24 detectors creating 50 channels. Image is used with permission by T. Zeng *et al.* [41].

5.4 Data Preprocessing

$\Delta[HbO_2]$ and $\Delta[HbR]$ were calculated using the modified Beer-Lambert law, then band-pass filtered from 0.01 Hz and 0.5 Hz to remove artifacts and physiological noise. If the correlation coefficient between $\Delta[HbO_2]$ and $\Delta[HbR]$ for a channel was greater than 0.5, then the channel was labeled a bad channel and the data from those channels was replaced by averages of the neighboring channels. One subject had a majority of

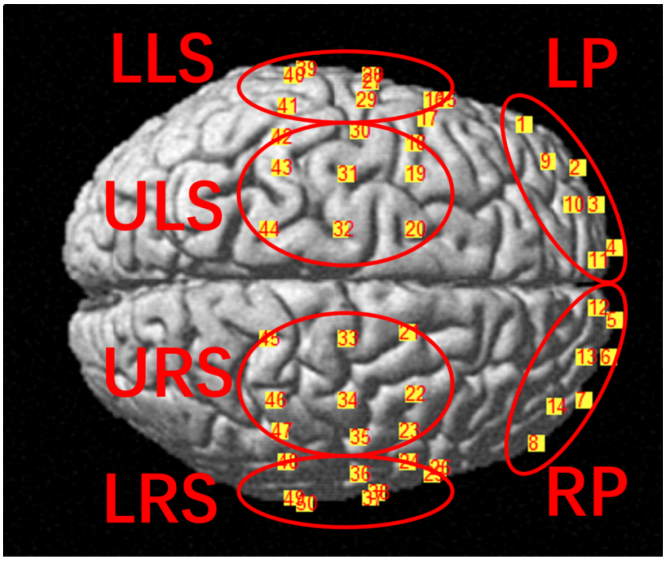

Figure 5.4: The image shows the positions of each of the 14 prefrontal cortex channels and 36 somatosensory cortex channels. There are six circled regions, denoted as LP (left prefrontal), RP (right prefrontal), LLS (lower left somatosensory), ULS (upper left somatosensory), LRS (lower right somatosensory), and URS (upper right somatosensory). Image is used with permission by T. Zeng *et al.* [41].

bad channels, and was thus removed from the dataset.

5.5 Feature Extraction

The data consists of 8 subjects, after removal of one subject with many bad channels. Each subject contributed 10 blocks (5 irritant-contained, 5 irritant-free). Every block is 30 seconds in duration, sampled at 7.81 Hz, which results in 234 samples per block. The features used in this classifier will be based on measures of pair-wise functional connectivity for all 14 fNIRS channels in the prefrontal cortex. Correlation coefficient is chosen as the measure of functional connectivity, primarily due to its simplicity of implementation. The correlation coefficients are calculated between all pairs of channels among the 14 prefrontal cortex channels, resulting in a 14×14 correlation coefficient matrix. An example is shown in Fig. 5.5.

The correlation coefficient matrix can be calculated for the entire 30 second time block, or for smaller time windows depending on the design of the classifier. In addition,

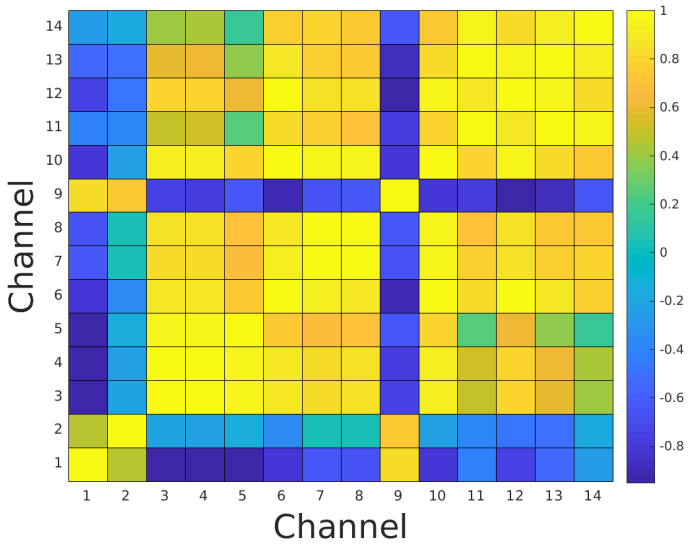

Figure 5.5: A visualization of a 14×14 correlation coefficient matrix extracted from the fNIRS raw time series. Each element in the matrix is the value of the correlation coefficient between two pairs of prefrontal cortex channels. The correlation coefficient, which measures functional connectivity, is the feature used in classification. A color legend is shown to the right.

if the size of the time window is smaller than the entire 30 second block, then it is possible to create many sliding time windows for each 30 second block, which yields multiple correlation coefficient matrices for each block. For example, if the entire time block is used for each correlation coefficient matrix, it would result in 80 possible matrices. On the other hand, if the size of the time windows is approximately 1 second (8 samples), and overlaps are permitted, then there would be $234 - 8 = 226$ possible windows, resulting in 18080 matrices. Since each correlation coefficient matrix is used as an input data sample to the classifier, generating as many matrices as possible helps to reduce overfitting.

5.6 Classification Experiments

Four types of classification experiments were performed with SVM and CNN as classifiers. Table 5.1 summarizes them. In experiments type A, B, and C, one single time window from each 30 second block is used to extract features. The time window

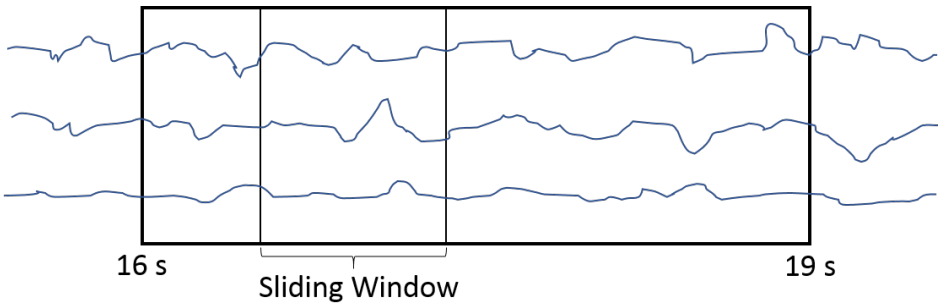

Figure 5.6: The figure demonstrates an example of a 1 second sliding window that is used to create correlation coefficient matrices in the time interval of 16 to 19 seconds in the time series data. Not drawn to scale.

of 15 to 20 seconds was chosen for classifier type A, as it was the time window proposed in [41] to have the most statistically significant difference between the two classes. In type B correlation coefficients are extracted from 16 to 16.5 seconds, while in type C they are extracted from a window of 18.5 to 19 seconds. These time windows were found to have the best performance out of the possible 0.5, 1, and 2 second windows from 15 to 20 seconds. In type D, 2 second wide sliding time windows are employed. The time range selected was from 16 to 19 seconds, which is a subset of the statistically significant 15 to 20 second range. This generated 480 total images for type D, rather than 80 images as with types A, B, and C. All four types use k-fold cross validation with $k = 8$, where each fold is the data from one of the 8 subjects. By splitting the folds by subject, the validation sets would be most independent from the training sets.

Table 5.1: SVM and CNN Classifier Types

Type of Experiment	**A**	**B**	**C**	**D**
Description	Single window	Single window	Single window	Sliding windows
Time Range (s)	$15 \rightarrow 20$	$16 \rightarrow 16.5$	$18.5 \rightarrow 19$	$16 \rightarrow 19$
Window Size (s)	5	0.5	0.5	2
No. Matrices/Images	80	80	80	480

Note: k-fold ($k = 8$) cross validation is used for validation for all types of classification experiments. Each fold is one of the 8 subjects. To input the correlation coefficient matrices to a CNN, each matrix is first converted to an image.

5.7 SVM Results

The SVM results for all four types of classification experiments are shown in Table 5.2. SVM-B had the best results with 62.5% testing accuracy, and also scored the best in precision, sensitivity, and specificity. However, SVM-B strongly overfit the data with a difference between training and testing accuracy of 18.39%. In addition, it also had the highest variance in accuracy between subjects, with a standard deviation of 16.69%. So while it may succeed in classification for some subjects, it may also perform poorly for some others. The performance for SVM-A and C were quite poor with $< 55\%$. SVM-D also had low performance at 56.26%, but the slight improvement over SVM-A and C could possibly suggest that the sliding window may have potential if explored further.

The average precision of the classifiers was low for SVM-A, C, and D and highest for SVM-B, which corresponds to the respective test accuracies of those classifiers. The average sensitivity and specificity show that each classifier type usually results in approximately equal weighting to both classes.

Overall, the SVM classifiers had achieved only limited performance. The best classifier with 62.5% accuracy can be used as a starting point for further improvement, but is not currently capable of reliably classifying data. The CNN classifiers designed in the next section seek to improve upon the SVM results.

5.8 CNN Architecture

Classification with CNN was first explored using a shallow architecture from scratch. However, this soon proved to be insufficient for the classification task, and had problems with overfitting as the SVM classifier did. In addition, with just one or two convolutional layers, the CNN from scratch was unable to develop higher level features which would be able to increase performance. So instead of building the CNN, transfer learning with a pretrained network was used. VGG-F was selected to be the pretrained network as it is relatively lightweight in computation requirements, while also showing good performance. The network architecture can be found in Table. 3.1 from Chapter 3.

Table 5.2: SVM Classifier Results

Type of Experiment	**SVM-A**	**SVM-B**	**SVM-C**	**SVM-D**
S1 Val. Accuracy	50	90	60	56.67
S2 Val. Accuracy	50	40	50	51.67
S3 Val. Accuracy	60	50	40	58.33
S4 Val. Accuracy	50	60	40	43.33
S5 Val. Accuracy	50	50	30	61.67
S6 Val. Accuracy	60	60	50	63.33
S7 Val. Accuracy	50	80	40	63.33
S8 Val. Accuracy	40	70	50	51.67
Average Testing Accuracy	51.25	62.5	45	56.25
St. Dev. Accuracy	6.41	16.69	9.26	7.00
Average Training Accuracy	80.54	80.89	72.32	85.33
Average Precision	30	82.5	52.5	45.83
Average Sensitivity	61.31	68.68	42.53	67.05
Average Specificity	54.14	79.17	46.88	56.84

Note: S1 $\rightarrow$ S8 denote subject 1 thru 8. All metrics are shown as percents (%).

The input to VGG-F is an RGB image of size $224 \times 224 \times 3$. In order to utilize VGG-F for this classification task, the input data must be resized to the format of these images. To achieve this, the 14×14 correlation coefficient matrix is zero-padded on all four sides by 105 elements. Then, the resulting matrix is replicated 3 times to resemble a 3-channel image with 3 identical layers. The final padded and 3-channel $224 \times 224 \times 3$ matrices, which will be referred to as input images, are shown in Fig. 5.7. With the input data resized, the output layer of VGG-F is replaced to output 2 classes instead of 1000 classes. An overview of the modified VGG-F is shown in Fig. 5.8. During network training, none of the layers were frozen and the entire network was tuned to the input data.

5.9 CNN Results

CNN classification again involved the types of experiments described in Table 5.1. The classification performance of all four types is summarized in Table 5.3. The learning rate was a constant rate selected manually for each experiment. The network was

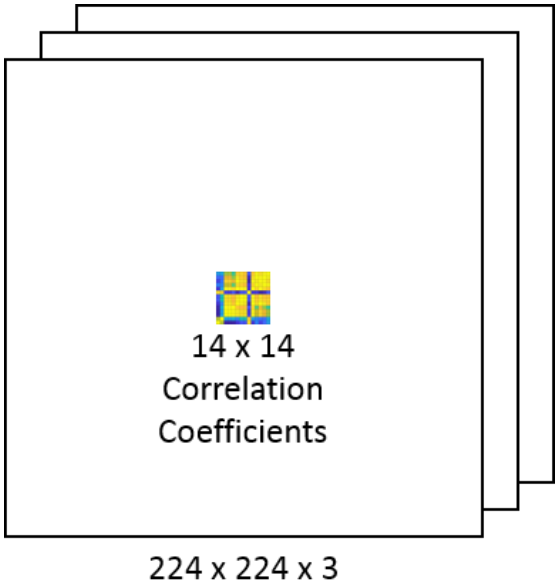

Figure 5.7: An illustration of the 14×14 correlation coefficient matrices after padding with 105 zeros on all sides and replicating to generate a $224 \times 224 \times 3$ input image that can be used with VGG-F. Each of the three replicated layers are identical.

trained for 74 epochs in all classifiers. This number of epochs was chosen as it was also the number used in the VGG Net paper [27]. The table shows both the training accuracy and the testing accuracy. The testing accuracy is most important as a measure of classification performance, while the training accuracy is useful to determine if the classifier had overfitting. All experiments used k-fold cross validation ($k = 8$), and the average testing accuracy was calculated by averaging the validation accuracy from each fold.

The best performing CNN classifier was CNN-B, with an average testing accuracy of 66.25%. However, CNN-B has a large standard deviation of 18.47% in testing accuracy. CNN-C has the second highest performance with an average testing accuracy of 61.25%. Both of these classifiers used single time windows of 0.5 seconds. In contrast, CNN-A used a single time window of 5 seconds from 15 to 20 seconds and had the lowest performance out of the others at just 50%. The performance of CNN-D is close to, but lower than that of CNN-C with 59.89%. As such, it seems that the sliding time window method was unsuccessful at improving the CNN results. The accuracies of the CNN

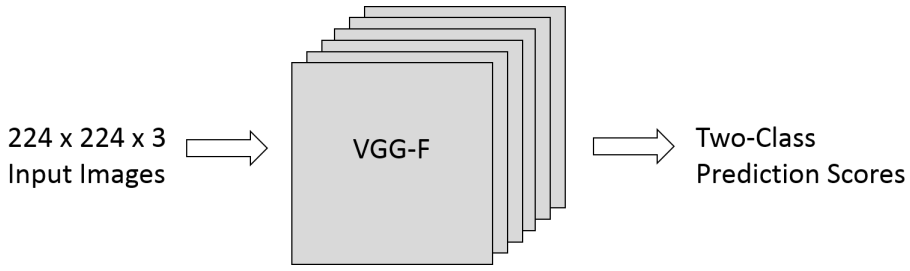

Figure 5.8: A depiction of the transfer learning with VGG-F. The input images are used to train and test the network, and there are two classes in the resulting output scores.

classifiers are improved or similar to their SVM counterparts.

The average precision of the classifiers was low for CNN-A and better for CNN-B, C, and D, which aligns with the respective testing accuracies. However, the average sensitivity and specificity show that each classifier type usually results in one of the classes being weighted higher than the other. Interestingly, each class is favored by two classifiers equally.

Though CNN-B and C were the two most successful CNN classifiers, they both showed overfitting, with the average testing accuracy falling short of the average training accuracy by 13.21% and 20.54% respectively. This is likely due to the small dataset used in both classifiers of just 80 images. Yet, CNN-D also had heavy overfitting with a 14.72% lower average testing accuracy than average training accuracy when it utilized 480 images. This could mean that even more than 480 images are needed, which is quite possible with a CNN. Another reason for overfitting may be that the learning rate or number of epochs was not optimal for the dataset.

The high standard deviation in average testing accuracy for CNN-B mostly resulted from the poor testing accuracy of 30% in subject 4. Meanwhile, CNN-C achieved 70% testing accuracy with subject 4. This suggests that developing a CNN which uses both of the time windows from CNN-B (16-16.5 s) and CNN-C (18.5-19 s) might yield improved classification.

In addition to classifier types A through D, all of which create one classifier using the data from all eight subjects, it is also of interest to see whether classifiers can be

Table 5.3: CNN Classifier Results

Type of Experiment	**CNN-A**	**CNN-B**	**CNN-C**	**CNN-D**
Learning Rate	1×10^{-5}	1×10^{-5}	1×10^{-5}	5×10^{-6}
Epochs	74	74	74	74
S1 Val. Accuracy	60	80	50	68.8
S2 Val. Accuracy	60	80	60	60.4
S3 Val. Accuracy	50	80	50	56.2
S4 Val. Accuracy	50	30	70	50
S5 Val. Accuracy	50	60	70	58.3
S6 Val. Accuracy	40	50	50	68.8
S7 Val. Accuracy	40	70	70	58.3
S8 Val. Accuracy	50	80	70	58.3
Average Testing Accuracy	50	66.25	61.25	59.89
St. Dev. Testing Accuracy	7.56	18.47	9.91	6.30
Average Training Accuracy	60.71	79.46	81.79	74.61
Average Precision	31.05	63.55	67.04	70.88
Average Sensitivity	35	80	52.5	34.38
Average Specificity	65	52.5	70	85.42

Note: S1 $\rightarrow$ S8 denote subject 1 thru 8. Average testing accuracy is calculated by averaging the validation accuracy of all subjects. All metrics are shown as percents (%).

created for individual subjects. It is expected that data within each subject should be fairly similar which may help improve accuracy. Accordingly, six experiment types were designed for single subject classifiers. Since the training data is now divided among the 8 subjects, sliding window must be used to generate more images. Shown in Table 5.4, sliding time windows of different time range and window sizes are used for extracting features in experiment types E through J. The table includes the number of images used in training each subject, which is now limited to the data from one subject per classifier, reducing the size of each dataset. The testing accuracy for each classifier is shown, along with the averages for testing accuracy, training accuracy, precision, sensitivity, and specificity among all subjects. All of the classifiers are trained with a learning rate of 1×10^{-5} and for 74 epochs. k-fold cross validation is used by separating the dataset into 5 folds where each fold is two of the ten trial blocks from each subject. Each fold contains one block from each class.

The results of these CNNs differed greatly between different subjects and classifier

types. 39 of the 48 classifiers had a testing accuracy less than 60%, 8 had an testing accuracy between 60% and 70%, and one had a testing accuracy above 70% with 73.61%, which was the best accuracy. This classifier was for subject 7 using CNN-J. However, none of the average testing accuracies over all 8 subjects were above 60% for any of the six classifier types, the highest being CNN-J with just 56.42%. Average precision, sensitivity and specificity are near 50% in most classifier types, which is expected with low accuracy. Average sensitivity and specificity are balanced, meaning that the classifiers predict each class with approximately equal weight. The classifiers all showed heavy overfitting, with a minimal difference between average training and testing accuracy of 17.38% in CNN-J, and a maximum difference of 30.57% in CNN-H. The overfitting in these classifier types is overall more severe than the classifiers from types A through D.

5.10 Conclusion

This chapter presented results on how measures of functional connectivity extracted as features perform with SVM and CNN classifiers to discriminate brain states. After collecting fNIRS data and preprocessing it, functional connectivity features are extracted in the form of correlation coefficient matrices, either for a single time window, or using a sliding time window. Four main types of classification experiments were performed for both SVM and CNN classifiers. The best performing SVM and CNN both used classifier type B, which included a single time window from 16 to 16.5 seconds. While the CNN classifiers did improve on the SVM classifier as anticipated, the classifier with the best performance just had moderate accuracy. In addition, CNN classifiers that were trained per subject showed improved averaged training accuracy, but suffered from increased overfitting. While there are areas for improvement, the results show that it is feasible to create SVM and CNN classifiers based on measures of functional connectivity in the prefrontal cortex. Consequently, the approach used for creating classifiers in this Thesis may have potential for adaptation to other types of classification tasks.

Table 5.4: CNN Single Subject Classifier Results

Type of Experiment	**CNN-E**	**CNN-F**	**CNN-G**	**CNN-H**	**CNN-I**	**CNN-J**
Time Range (s)	$14 \rightarrow 21$	$14 \rightarrow 21$	$14 \rightarrow 21$	$16 \rightarrow 19$	$16 \rightarrow 19$	$16 \rightarrow 19$
Window Size (s)	2	1	0.5	2	1	0
Images	3040	3680	4000	480	1120	1440
Images/Sub	380	460	500	60	140	180
S1 Testing Accuracy	55.53	56.09	49.60	43.33	59.29	64.44
S2 Testing Accuracy	55.53	55.65	53.80	61.67	65.71	57.22
S3 Testing Accuracy	44.74	44.57	49.20	65.00	49.29	54.44
S4 Testing Accuracy	44.47	48.04	50.40	66.67	39.29	48.33
S5 Testing Accuracy	65.79	54.57	57.80	50.00	68.57	54.44
S6 Testing Accuracy	54.47	48.04	52.00	60.00	42.14	50.00
S7 Testing Accuracy	52.89	50.00	52.00	38.33	48.57	73.61
S8 Testing Accuracy	55.53	48.26	48.20	53.33	52.14	48.89
Avg. Testing Acc.	53.62	50.65	51.63	54.79	53.13	56.42
St. Dev. Testing Acc.	6.79	4.25	3.08	10.33	10.58	8.72
Avg. Training Acc.	75.77	70.75	71.03	85.36	80.47	73.80
Avg. Precision	53.37	53.09	53.47	30.42	47.99	50.28
Avg. Sensitivity	53.16	45.00	52.40	60.00	45.54	53.05
Avg. Specificity	54.08	56.30	50.85	49.58	60.72	54.86

Note: Learning rate is 1×10^{-5} and number of epochs is 74 for all classifiers in this table. k-fold cross validation is done using 5 folds belonging to one subject, where each fold contains two experimental trial blocks. Accuracies are all shown as percents (%)

Chapter 6

Conclusion

This Thesis explored the feasibility of using measures of functional connectivity in the prefrontal cortex as features to develop models of discriminating brain states from fNIRS recordings.There are numerous benefits to fNIRS compared with other neuroimaging techniques, such as EEG and fMRI, but fNIRS still suffers from drawbacks that are directly related to the number of optodes used. Some of these issues include long setup time and wearer discomfort. The development of an fNIRS classification technique which only requires prefrontal cortex signals would reduce the number of optodes needed and decrease the effect of these drawbacks, especially since unlike other areas, there is little or no hair in the prefrontal cortex region which allows for ease of setup and better signal quality. This study attempts to find a solution for this challenge by using functional connectivity measures of the prefrontal cortex as features for SVM and CNN classifiers.

To measure functional connectivty, correlation coefficient matrices were used. The correlation coefficient matrices were calculated using the fNIRS time series data from 14 prefrontal cortex channels. Four different classification experiments were performed by varying the time windows used in calculating the correlation coefficient matrices. Three of the four classification experiments use one single time window for feature extraction, while a sliding time window method generating many more windows and augmenting the training and testing datasets was used for the fourth experiment. k-fold cross validation was used to evaluate the performance of the classifiers, where the folds are divided by subject. The sliding time window method and the k-fold cross validation aim to reduce the presence of overfitting in the data, which is a common problem for neuroimaging datasets since obtaining large quantities of data is a challenge.

The experimental task used for data collection involved measuring the fNIRS responses to irritant-contained and irritant-free solutions. The results with both SVM and CNN classifiers show that it is possible to create a classifier based on measures of functional connectivity in the prefrontal cortex.

6.1 Future Work

This work explored the possibility of using functional connectivity in the prefrontal cortex for discriminating brain states, but further work can be done to improve the classification performance, especially for the CNN classifier. The main impediment to achieving higher accuracy was overfitting. The small quantity of data is most likely the cause of the overfitting. Therefore, if additional overfit reduction strategies can be used during classifier training then the performance will likely improve.

The tuning of the CNN can be improved using early stopping of training and decay of learning rates. The CNN proposed in this Thesis trains for a certain fixed number of epochs. In contrast, the early stopping method checks the validation accuracy during each epoch as the CNN trains, and stops the training when the validation accuracy no longer increases. This is very useful for reducing the overfitting in the model. The learning rates for the CNN classifiers here also had fixed learning rates. Decaying the learning rate using Adam [21] or a similar technique could potentially result in a faster and more optimal training of the network.

The results from this Thesis found that the choice of features has a large impact on the performance of the CNN classifier. In many cases, a classifier would perform more accurately with some subjects than others. Therefore, there may be an improvement in performance if features extracted from different time windows were can be together in one feature space. Another possible method for adjusting the feature space is to choose what channels to include when calculating the correlation coefficient matrix. This study included data from all available prefrontal cortex channels, but perhaps the use of all 14 channels was not necessary and contributed to the heavy overfitting seen in most of the classifiers.

The CNN architecture also may be improved upon. VGG-F was chosen as the pretrained network for transfer learning because of its speed, not necessarily because of its exceptional accuracy. For example, Microsoft's ResNet architecture has significantly greater performance than that of VGG-F when it was published in 2015 [42]. Hence, ResNet may also have superior results in transfer learning, especially if combined with improved network tuning and feature selection.

References

[1] A. Villringer and B. Chance, "Non-invasive optical spectroscopy and imaging of human brain function," *Trends in Neurosciences*, vol. 20, no. 10, pp. 435–442, Oct. 1997.

[2] S. Ogawa, T. M. Lee, A. R. Kay, and D. W. Tank, "Brain magnetic resonance imaging with contrast dependent on blood oxygenation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 87, no. 24, pp. 9868–9872, Dec. 1990.

[3] A. Bakker, B. Smith, P. Ainslie, and K. Smith, "Near-infrared spectroscopy," *Applied Aspects of Ultrasonography in Humans*, pp. 65–86, Apr. 2012.

[4] X. Cui, S. Bray, D. M. Bryant, G. H. Glover, , and A. L. Reiss, "A quantitative comparison of nirs and fmri across multiple cognitive tasks," *NeuroImage*, vol. 54, no. 4, pp. 2808–2821, Nov. 2010.

[5] A. R. Harrivel, A. G. Hylton, and T. A. Hearn, "Best practices for the application of functional near infrared spectroscopy to operator state sensing," *NTRS*, vol. 54, no. 4, July 2012.

[6] M. Schweiger, I. Nissilä, D. A. Boas, and S. R. Arridge, "Image reconstruction in optical tomography in the presence of coupling errors," *Appl. Opt.*, vol. 46, no. 14, pp. 2743–2756, May 2007.

[7] B. Khan, C. Wildey, R. Francis, F. Tian, M. R. Delgado, H. Liu, D. MacFarlane, , and G. Alexandrakis, "Improving optical contact for functional nearinfrared brain spectroscopy and imaging with brush optodes," *Biomedical Optics Express*, vol. 3, no. 5, May 2012.

[8] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *Science and Information Conference (SAI)*, Aug. 2014, pp. 372–378.

[9] R. T. Schirrmeister, L. Gemein, K. Eggensperger, F. Hutter, and T. Ball, "Deep learning with convolutional neural networks for decoding and visualization of eeg pathology," *Human Brain Mapping*, pp. 5391–5420, Aug. 2017.

[10] J. Lever, M. Krzywinski, and N. Altman, "Model selection and overfitting," *Nature Methods*, vol. 13, Aug. 2016.

[11] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–6, Nov. 2016.

[12] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, Aug. 1995, pp. 1137–1143.

[13] L. Breiman and P. Spector, "Submodel selection and evaluation in regression. the x-random case," vol. 60, Mar. 1992.

[14] M. Hearst, "Support vector machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, July 1998.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[16] N. Lu, T. Li, X. Ren, and H. Miao, "A deep learning scheme for motor imagery classification based on restricted boltzmann machines," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 566–576, June 2017.

[17] M. Lin, Q. Chen, and S. Yan, "Network in network," *ICLR*, Mar. 2014.

[18] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, Dec. 2014.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, June 2010, pp. 807–814.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," vol. 15, pp. 1929–1958, June 2014.

[21] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, Dec. 2014.

[22] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328.

[23] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[24] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, ser. ICML'14, 2014, pp. 647–655.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Curran Associates Inc., 2012, pp. 1097–1105.

[26] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference*, 2014.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, Sept. 2014.

[28] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, pp. 427–437, Mar. 2009.

[29] R. Parikh, A. Mathai, S. Parikh, G. C. Sekhar, and R. Thomas, "Understanding and using sensitivity, specificity and predictive values," *Indian Journal of Ophthalmology*, vol. 56, pp. 45–50, Feb. 2008.

[30] L. Xu, B. Wang, G. Xu, W. Wang, Z. Liu, and Z. Li, "Functional connectivity analysis using fnirs in healthy subjects during prolonged simulated driving," *Neuroscience Letters*, vol. 640, Jan. 2017.

[31] S. Sasai, F. Homae, H. Watanabe, and G. Taga, "Frequency-specific functional connectivity in the brain during resting state revealed by nirs," vol. 56, pp. 252–7, May 2011.

[32] H. E. Wang, C. G. Bnar, P. P. Quilichini, K. J. Friston, V. K. Jirsa, and C. Bernard, "A systematic framework for functional connectivity measures," *Frontiers in Neuroscience*, vol. 8, p. 405, Dec. 2014.

[33] A. V. Medvedev, J. M. Kainerstorfer, S. V. Borisov, and J. VanMeter, "Functional connectivity in the prefrontal cortex measured by near-infrared spectroscopy during ultrarapid object recognition," *Journal of Biomedical Optics*, vol. 16, Jan. 2011.

[34] B. Molavi, L. May, J. Gervain, M. Carreiras, J. Werker, and G. Dumont, "Analyzing the resting state functional connectivity in the human language system using near infrared spectroscopy," *Frontiers in Human Neuroscience*, Jan. 2014.

[35] N. Naseer and K.-S. Hong, "Fnirs-based brain-computer interfaces: A review," *Frontiers in Human Neuroscience*, Jan. 2015.

[36] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from eeg with deep recurrent-convolutional neural networks," in *ICLR 2016*, Feb. 2016.

[37] N. Naseer, N. K. Qureshi, F. M. Noori, and K.-S. Hong, "Analysis of different classification techniques for two-class functional near-infrared spectroscopy-based brain-computer interface," *Computational Intelligence and Neuroscience*, June 2016.

[38] T. Trakoolwilaiwan, B. Behboodi, J. Lee, K. Kim, and J.-W. Choi, "Convolutional neural network for high-accuracy functional near-infrared spectroscopy in a brain-computer interface: three-class classification of rest, right-, and left-hand motor execution," *Neurophotonics*, Sept. 2017.

[39] J. Hennrich, C. Herff, D. Heger, and T. Schultz, "Investigating deep learning for fnirs based bci," in *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug. 2015, pp. 2844–2847.

[40] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, Mar. 2016.

[41] T. Zeng, D. Peru, V. P. Maloney, and L. Najafizadeh, "Cortical activity changes as related to oral irritation-an fnirs study," in *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*, July 2017, pp. 2558–2561.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Dec. 2015.