THE INTERDISCIPLINARY CENTER,

HERZLIA

MASTER THESIS

Using Recurrent Neural Networks for P300-based BCI

Author:

Ori TAL

Advisor:

Dr. Doron FRIEDMAN

A thesis submitted in fulfillment of the requirements for the degree of Master of Science (M.Sc.) Research Track in Computer Science

December 4, 2017

The Interdisciplinary Center, Herzlia

Abstract

Efi Arazi School of Computer Science

Computer science

Master of Science (M.Sc.) Research Track in Computer Science

Using Recurrent Neural Networks for P300-based BCI

by Ori TAL

P300-based spellers are one of the main methods for EEG-based brain computer interface, and the detection of the P300 target event with high accuracy is an important prerequisite. The rapid serial visual presentation (RSVP) protocol is of high interest because it can be used by patients who have lost control over their eyes. In this study we wish to explore the suitability of recurrent neural networks (RNNs) as a machine learning method for identifying the P300 signal in RSVP data. We systematically compare RNN with alternative methods such as linear discriminant analysis (LDA) and convolutional neural network (CNN). Our results indicate that RNN shows good results only with large amounts of data, and we show that a network combining CNN and RNN is significantly more resilient to temporal noise than other methods.

Acknowledgements

I wish to express my sincere thanks to Dr. Doron Friedman who gave me the freedom to explore the new field of deep learning and supported me along the way.

Special thanks to my wife Shira, for her utmost support, patience and belief in my work.

Contents

Abstract iii

Acknowledgements v

1 Introduction 1

1.1 Deep Neural Networks - Overview 2

1.1.1 Layer Types 3

2 Previous work 7

3 MATERIALS AND METHODS 9

3.1 P300 speller experiments settings 9

3.1.1 Formulating the BCI task 10

3.1.2 RSVP P300 speller trials 10

3.1.3 Loss function 11

3.1.4 Evaluated Models 12

3.2 Implementation Details 14

4 Results 17

5 Discussion and Future Directions 23

6 References 25

Chapter 1

Introduction

This thesis is based on an article presented in the 7th Graz Brain-Computer Interface Conference. While the article is focused on describing our results, in this report we added a more detailed explanation about the method we were using in our experiments.

Neural networks have recently been shown to achieve outstanding performance in several machine learning domains such as image recognition [17] and voice recognition [14]. Most of these breakthroughs have been achieved with convolutional neural networks (CNNs) [18], but some promising results have also been demonstrated by using recurrent neural networks (RNNs) for tasks such as speech and handwriting recognition [12, 11], usually when using the long short-term memory (LSTM) architecture [15].

There have been some studies using 'deep neural networks' for P300 classification [5, 21]. The results reported, despite some success, do not show the same dramatic progress achieved by 'deep learning' methods as compared to the previous state of the art; while in areas such as image or voice recognition 'deep' neural networks have resulted in classification accuracy exceeding other methods by far, this has not yet been the case with EEG in general and in P300 detection specifically. The small number of samples typically available in neuroscience (or brain computer interface - BCI) is most

likely one of the main reasons for theses results. In addition, the high dimensionality of the EEG signal, the low signal to noise (SNR) and the existence of outliers in the data, pose other difficulties when trying to use neural networks for BCI tasks (see [20]). The main question in this research is whether the RNN model, and particularly LSTM, can enhance the accuracy of P300-based BCI systems and if so, under what conditions.

P300-based BCI systems rely on identifying the times when a subject is required to pay attention toward a rare event, by examining the subject's electroencephalogram (EEG) data. The first BCI system that used the P300 effect was presented by Farwell and Donchin [9] and since then different versions of P300 based BCI systems were suggested. One example of such a paradigm is the P300 rapid serial visual presentation (RSVP) speller. In this paradigm letters are presented one after the other in a random order, and the subject is asked to pay attention only to one of the letters called *target* letter or *target stimuli* (by counting them silently, for example). Whenever a subject pays attention to the target letter, a special waveform called P300 is expected to occur. It is called P300 since there is usually a peak in the EEG amplitude 300ms after the presentation of a target event. The advantage of the RSVP paradigm is that it does not require any eye movements, and can thus be operated by patients who have lost control of their eye gaze completely.

1.1 Deep Neural Networks - Overview

Deep neural networks (DNN) are relatively large artificial neural networks (ANN) with multiple layers. There are two main types of ANN architectures: feed forward neural networks (FFNN or FF) and recurrent neural network (RNN). In FFNN directed cycles are not allowed (i.e., data can flow only to the next layer), while the RNN architecture allows directed cycles within the network (specifically, data can also flow between 'neurons' in the same

layer). The directed cycles in RNN allows the network to 'remember' past events and making it suitable for sequence learning [25].

1.1.1 Layer Types

The architecture we propose is a combination of several ANN layer types described below:

Fully connected layer - FC - a layer where each neuron in the input is connected to each neuron in the output is often called fully connected layer or FC. In an FC layer, the output is obtained by the following equation:

$$y = \sigma\left(xW + b\right)$$

where x is the input vector, W is a matrix that represents a linear mapping and b is a vector that reflects the transformation called the $bias$ (b holds the value for each output unit). $\sigma\left(\cdot\right)$ represent an element-wise non-linearity such as rectified linear unit (ReLU), sigomid or hyperbolic tangent (TanH):

$$\mathrm{ReLu}\left(x\right) = \max\left(x, 0\right) \quad , \quad \mathrm{sigmoid}\left(x\right) = \frac{1}{1+e^{-x}} \quad , \quad \mathrm{TanH}\left(x\right) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-z}}$$

Fig.1.1 shows an example of a simple network with two inputs and outputs, and one fully connected layer with 3 cells. All the nodes in the hidden layer are connected to all input and output nodes. Since it is a feed forward architecture, the data flows only in one direction: from the input, to the hidden layer, and then to the output.

Convolutional Neural Network Layer (CNN) CNN is a layer that utilizes the local correlation between adjacent cells of the input layer. Since, unlike the FC layer, the output can be of multiple dimensions, we refer to the output as *feature map*. The feature maps are obtained by activating trainable

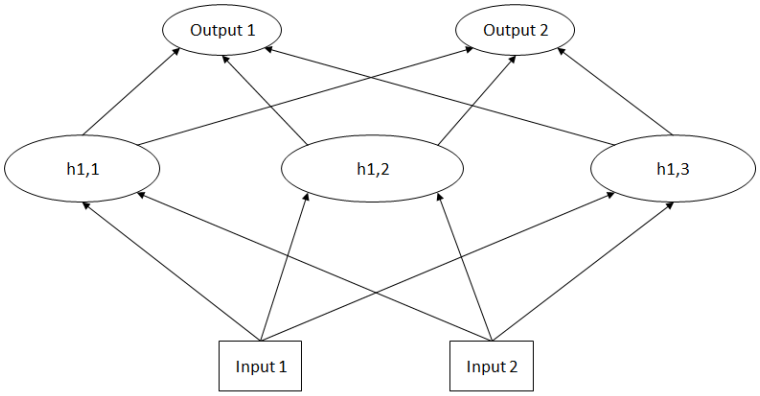


FIGURE 1.1: Illustration of a simple feed-forward network with one hidden layer. $h_{i,j}$ means the i-th hidden cell on the j-th layer.

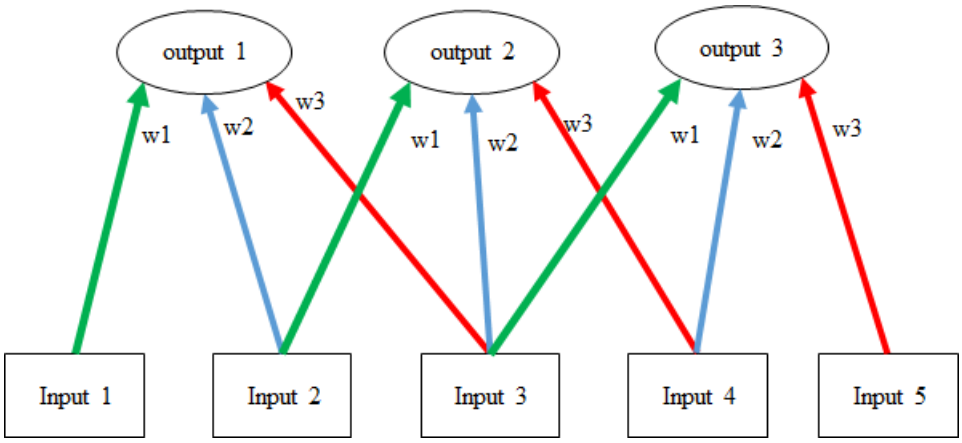


FIGURE 1.2: A Schematic diagram of a 1D CNN layer. The input number reflects the sample's order in time (i.e., "input 1" is the first input and "input 5" is the last input).

multi-dimension kernels across the input layer. Fig.1.2 is an example of a 1D CNN. The red, blue and green colors represent the trainable kernel weights ($w1$, $w2$ and $w3$ respectively) across the input layer. Equation 1.1 describes the output of element i in a 1D CNN layer feature map:

$$y(i) = \sigma(W * x[i-k, ..., i+k] + b) \tag{1.1}$$

If the kernel is a 1D filter with a length of k and the next layer has M outputs, then W is a matrix of size $M \times k$.

Recurrent Neural Network Layer - RNN - The simplest form of a recurrent neural network [25, 31] is a layer where the output is connected to the

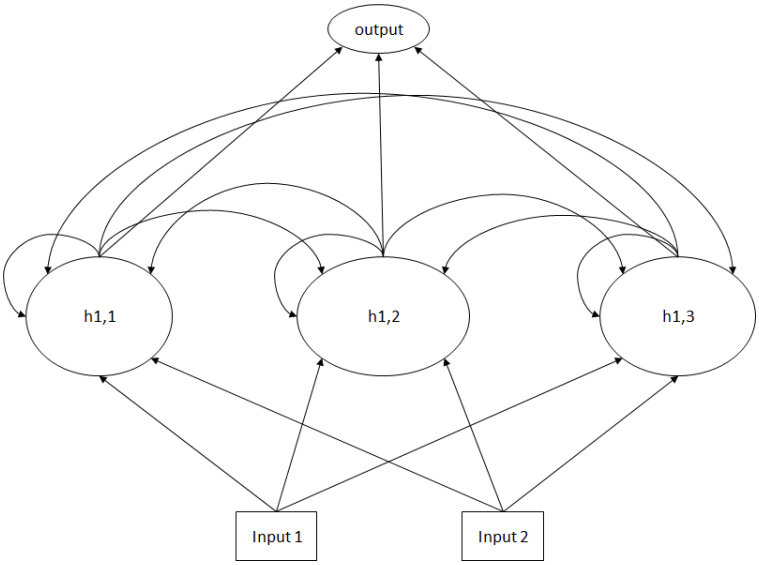


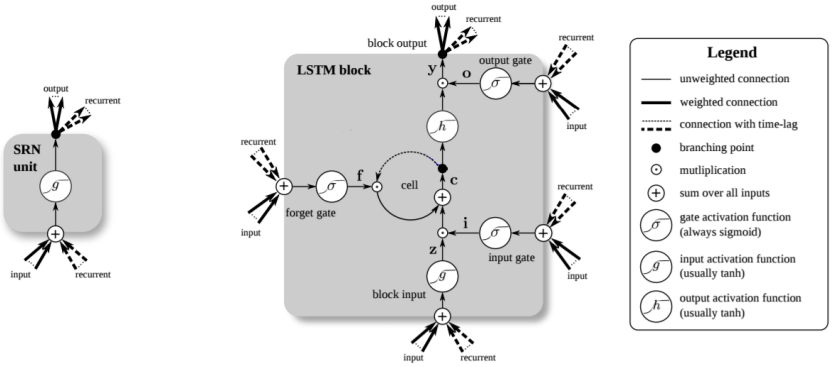
FIGURE 1.3: A schematic diagram of a simple recurrent neural network with one hidden layer.

input. Unlike feed forward layers, in RNN, $y[t]$ (the layer output at time t) is a function of both the current input $x[t]$ and $y[t-1]$ (the previous state):

$$y\left(t\right)=\sigma\left(x\left[t\right]W+y\left[t-1\right]U+b\right) \tag{1.2}$$

Here, again σ is a non-linear activation function, W is the input weight matrix and U is the previous results weight matrix. The structure of an RNN layer, allows the network to contain memory, since it has access to information from previous time-stamps within each predicted sample. RNN is known to suffer from a phenomena called "vanishing gradient" and "exploding gradient" [23]: while training, the gradient of the loss function may not propagate to the first layers (i.e., layers closer to the input layer) or may have very large values (thus update the layer weight too much). These problems prevent learning long temporal dependencies (see [3]) .A common solution for these problem is called Long Short Term Memory layer.

Long Short Term Memory Layer - LSTM - LSTM is a type of RNN with a special architecture designed to overcome RNN's difficulties in learning long term dependencies. In addition to the output - $y[t]$, LSTM also has a



(a) A schematic diagram of the vanilla RNN (b) A schematic diagram LSTM block. [13]

memory cell - $c[t]$. LSTM uses an architecture with a set of "gates" that allow it to decide whether a data should be stored in the memory unit or not. The gating mechanism has been prove to overcome the vanishing and exploding gradients problem mentioned above (see [15]). Fig.1.4a and Fig.1.4b shows the difference between vanilla RNN and LSTM.

Chapter 2

Previous work

There are many methods for building systems that can identify the P300 target for a BCI task. Blankertz et al. [4] suggest selecting the time intervals with maximal separation between the target and non target samples, averaging their *electro-potential* value and use shrinkage LDA to classify these features. Using this method has a drawback due to the low complexity of the LDA model [7]. The winner of the BCI competition III: dataset II used an ensemble of support vector machines (SVM) [24], and other methods include hidden Markov model, k-nearest neighbours, and more [7].

More recently, given the success of deep neural networks [17], there have been several attempts to apply 'deep learning' for BCI related tasks. Cecotti and Graser [5] were the first to use CNNs for a P300 speller. In their work, they train an ensemble of CNN-based P300 classifiers to identify the existence of a P300 event. Manor and Geva [21] used CNN for the RSVP P300 classification task and suggested a new spatio-temporal regularization, which have shown improvement in the performance.

Unlike feed forward network models such as CNN and multi-layer perceptron (MLP), the RNN architecture allows directed cycles within the network, which enable the model to "memorize past events". LSTM [15] is a type of RNN, which includes a special node that can be described as a differentiable memory cell. The specific architecture of LSTM enables it to overcome some of the weakness of simple RNNs [3].

There are several reasons why LSTM is a good candidate for modelling the P300 pattern. First, RNN and LSTM have shown success when modeling time series for tasks such as handwriting and speech recognition [12, 11, 33]. In addition, RNN is known to have the capability to approximate dynamical systems [19], which makes it a natural candidate for modelling the dynamics of EEG data. Finally, RNN can be seen as a powerful form of hidden Markov models (HMM), which have been shown to classify EEG successfully [27, 22, 7]; RNNs can be seen as HMMs with an exponentially large state space and an extremely compact parametrization [28].

LSTM was already used for analysing EEG data for emotion detection [26] and a phenomena called behavioral microsleeps [8]. Bahshivan et al. [2] modeled inter-subject EEG features for identifying cognitive load by using convolutional LSTM. They created a video from three different band powers in each electrode. One of the major differences between their work and ours is that we use the original signal without any feature extraction (such as band power), and we focus specifically on the P300 speller domain.

Chapter 3

MATERIALS AND METHODS

We compared the performance of LSTM based methods with other methods on a dataset from a RSVP P300 speller study [1]. We used average prediction across 10 trials to measure the P300 speller accuracy as applied in [1].

3.1 P300 speller experiments settings

The dataset includes 55 channels of EEG recordings from 11 subjects. Each subject is presented with 10 repetitions of 60 to 70 sets of 30 different letters and symbols. In total there are approximately 20,000 samples for each subject where 1/30 of them are supposed to contain a P300 wave. While the original experiment contains 3 different settings (interval of 116ms with/without colors and 83ms with color), we used the experiment setting of 116ms intervals with letters in different colors. For more details, see [1].

In addition to the filters applied in [1], all models that we used share the same pre-processing stage of down-sampling the input frequency from 200Hz to 25 Hz. The result is that each learning sample is a matrix of 55 channels with 25 time samples each, or $55 * 25 = 1375$ features. Each sample thus covers exactly 1 second around the target event, at times [-200,800] ms.

3.1.1 Formulating the BCI task

In P300 speller the task is to identify the letter to which the subject paid his attention by identifying a P300 pattern in the EEG. This can be done by finding a function $f(x)$ that when given an EEG sample – x, returns the probability that a P300 pattern is found in it. By identifying the EEG sample with the maximal $f(x)$ score among the EEG samples of all the different letters we can identify the target stimulus (i.e., the letter that the subject focused on). First, we will formulate the single letter prediction task.

3.1.2 RSVP P300 speller trials

Notations

$f(x)$	P300 identification function
X_c	An EEG sample data that was recorded when presented character c
C	The set of all the available stimuli
$\hat{c}$	The predicted target stimuli
$c*$	The true target stimuli (i.e. the letter the subject was suppose to focus on)
R	Group of trials (see below)
$x_{c,r}$	An EEG sample data that was recorded when presented character c on trial r

A trial is the presentation of all the characters in an alphabet, one after the other, in random order. In each trial, only one letter is the actual target stimuli.

The predicted character $\hat{c}$ in a single trial is computed by finding the character with the maximal value of $f(x)$ among the group of letters (C):

$$\hat{c} = \arg\max \{f(x_c)\}_{c \in C} \tag{3.1}$$

In order to achieve robust character recognition prediction, a common approach is repeating each trial several times and averaging the prediction for each different stimulus. The set of repetitions of the same trial will be called R . An EEG recording where stimuli c presented in trial r is called $x_{c,r}$. The formula for predicting the target stimuli across R attempts is:

$$\hat{c} = \arg\max \left\{ \frac{1}{R} \sum_{r \in R} f(x_{c,r}) \right\}_{c \in C} \tag{3.2}$$

3.1.3 Loss function

In neural networks, the weights are updated by deriving the loss function with respect to the network weights. If we train the neural network to identify P300 directly (as in [5]), the error only depends on whether the sample contains P300 or not. Here $y_{r,c}$ refers to the true label of sample $x_{r,c}$:

$$E = e(f(x_{r,c}), y_{r,c}) \tag{3.3}$$

In this research we use the binary log loss function:

$$e(f(x_{r,c}), y_{r,c}) = -(y_{r,c} \log(f(x_{r,c})) - (1 - y_{r,c}) \log(1 - f(x_{r,c}))) \tag{3.4}$$

The error in a neural network is typically calculated on multiple samples. In this case the error is:

$$E = \frac{1}{M} \sum_{i}^{M} e(f(x_i, y_i)) \tag{3.5}$$

Here M indicates the size of the samples batch.

3.1.4 Evaluated Models

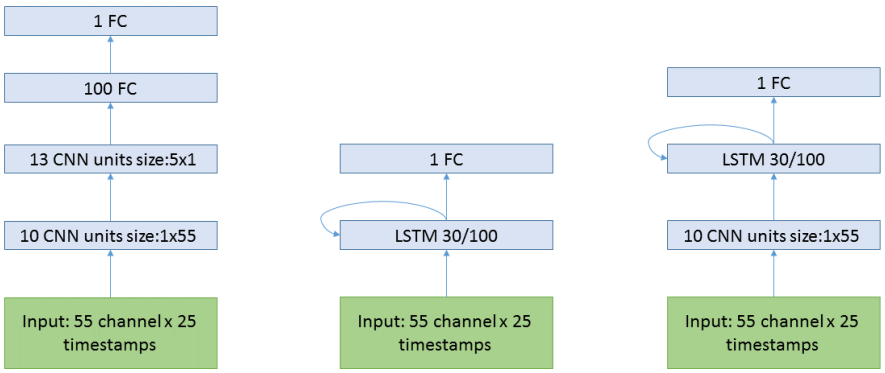
The models evaluated in this experiment are:

- LDA - A common method used in P300 classification for BCI is linear discriminative analysis [1, 4]. Here we will use a simplified version; unlike [1] we use all the timestamps as features, and we use a non-shrinkage version of LDA.

- CNN (7924 free parameters) (Fig.3.1a and Fig.3.2a) – The CNN model we use is similar to the one used in [5]. The first layer is composed of 10 spatial filters, each of size $55 * 1$ – the number of channels. The second layer contains 13 different temporal filters with size of $1*5$. Each one of the temporal filters processes 5 subsequent time stamps without overlapping. The third and fourth layers are simple fully connected layers followed by a single cell with a sigmoid activation function that emits a scalar.

- LSTM large/small (62501/10351 free parameters) (Fig.3.1b) – LSTM large/small are both composed of single LSTM layers with 100 and 30 hidden cells in each, correspondingly. Both models end with a single cell with a sigmoid activation layer that emits a scalar.

- LSTM-CNN large/small (49051/5511 free parameters) (Fig.3.1c and Fig.3.2b) – The model has CNN as a first layer (the spatial domain layer) and LSTM as the second layer for the temporal domain. The first convolutional layer is the same as in the CNN model. Unlike the CNN model, the temporal layer is an LSTM layer with 100/30 hidden cells. The last layer contains a single cell a with a sigmoid activation layer that emits a scalar.



(a) CNN model (b) LSTM model (c) CNN-LSTM model

FIGURE 3.1: Schematic diagrams of the neural networks evaluated. FC stands for fully connected layers.

In order to examine the power of each method in modelling the inter-subject and intra-subject variance we have conducted the following experiments:

1. Training and testing on each subject's data separately in order to explore intra-subject generalization.

2. Training and testing on different subjects data combined in order to investigate the impact of larger amounts of data.

3. Transfer learning between subjects - training on all subjects expect one. We conduct this experiment in order to explore the value of using a model that was trained off-line, on different subjects, and then use this model on a new subject, with or without additional calibration.

In addition to the three experiments above, we also examine the model's tolerance to time domain noise. A highly desired property from BCI systems is tolerance to a small degree of noise in the stimuli onset time. In order to evaluate the resistance to such noise, we use a model trained on the original stimuli onset (i.e, noise level = 0ms) and evaluate its performance on different stimuli onset to simulate small errors on the stimuli onset: noise levels of -120ms,-80ms,-40ms, +40ms, +80ms, and 120ms. We conducted this experiment using 10-fold cross validation in order to be able to get statistically

significant results. This last experiment was conducted only on the CNN and LSTM-CNN models and used data from all subjects (as in experiment 2 described above).

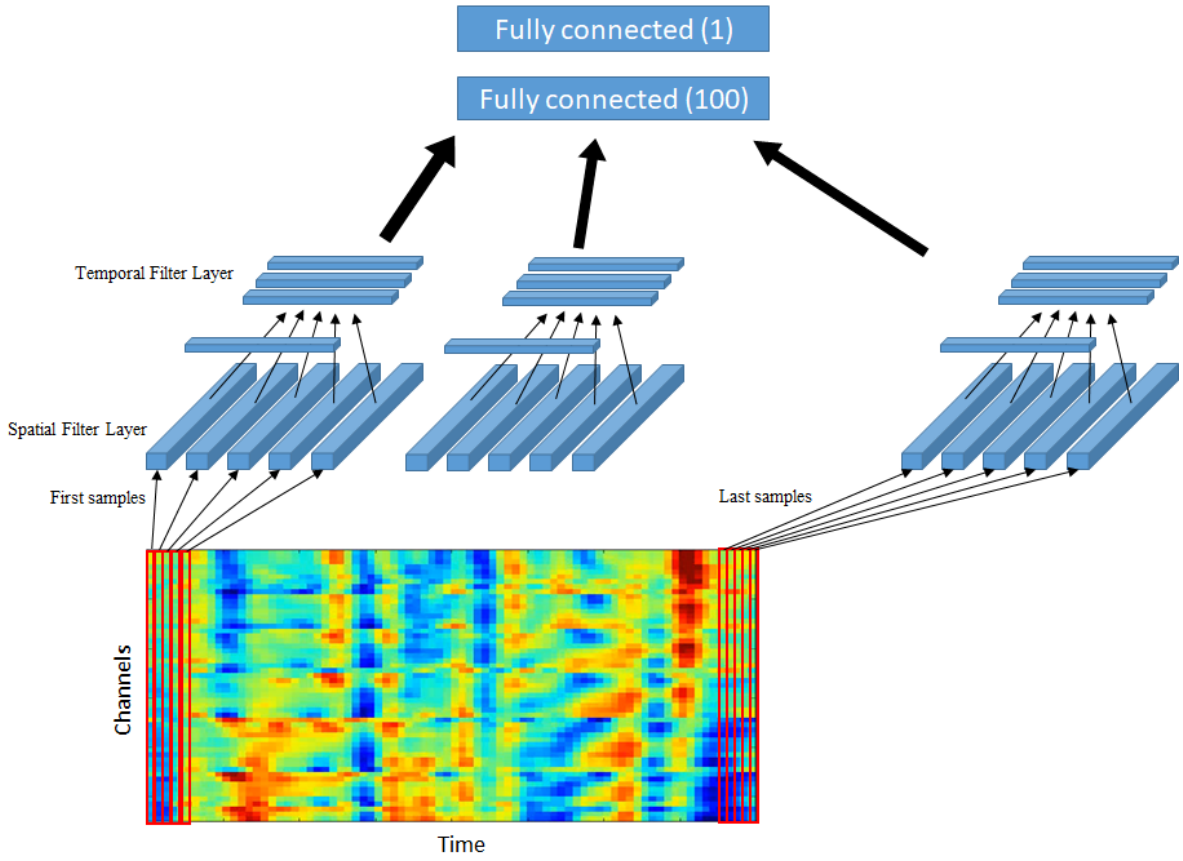
For all experiments, the different models were trained using the RMSProp [30] optimizer, for 30 epochs with a learning rate of 0.001 and then continue to train for 30 epochs with the a learning rate of 0.00001.

RMSProp [30] is a stochastic gradient descent (SGD) method. Unlike simple SGD, the method can adapt different learning rates for each parameter separately and use a moving average across the past gradient in order to scale the learning rate per feature. We decided to use RMSProp since it is said to be robust and fast [32, 16, 29].

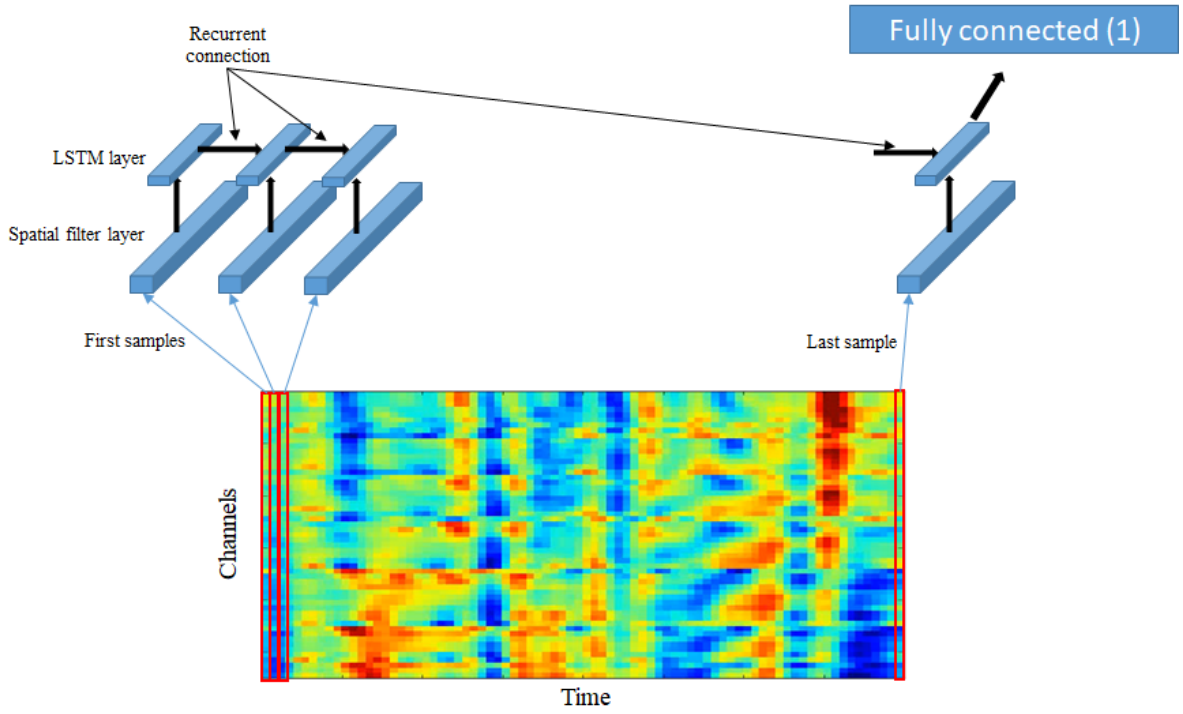
3.2 Implementation Details

The code was implemented using the Keras framework [6]. Training was conducted using a 4-core i7 laptop with 16Gb RAM. Training took 110 seconds on 192000 samples for the small LSTM-CNN model and 24 seconds for the CNN model. The considerable difference is due to the distributed nature of CNN which allows much of the computation to be computed in parallel. Predicting on a single example takes about 0.6 milliseconds. In term of space, both models require less than 70kb of disk space.

One of the advantages of using 'deep learning' models is that they allow compressing knowledge from many samples into a compact form. As we show in our experiments, it is possible to pre-train on multiple subjects and then fine tune to a specific subject's calibration data. For example, training on 3000 calibration samples using the 4-core i7 laptop will take less than a minute (fine-tuning for 30 epochs). After the model is trained, using it for real-time prediction is feasible as well, since predicting each sample takes 0.6 milliseconds.



(a) A diagram of the CNN model used in our research. All the connections are directed towards the next layer.



(b) A diagram of the CNN-LSTM model used in our experiments. Unlike the CNN model, there are connections between cells from the same layer. Only the last sequence is connected to the fully connected layer.

FIGURE 3.2: An additional illustration highlighting the main differences between the CNN and LSTM-CNN architectures.

Chapter 4

Results

Tab.4.1 summarizes the results of the different experiments; all results are based on an average of 10 consecutive trials to detect the target letter, as in [1]. The results for training and testing on the same subject (Tab.4.2) indicate that LSTM is inferior (82%), and even the LSTM_CNN combined model performs less than the the simple LDA method (86 and 93% in the LSTM_CNN models and 96% using LDA) . A possible advantage for LSTM only becomes apparent with larger amounts of data – when training and testing on all the subjects together (Tab.4.1). The large LSTM model performs poorly – 77%; we suspect it is due to the large number of trainable parameters – 62501 ("over-fitting"); this is why we introduced CNN as a first layer and reduced the number of hidden LSTM cells.

Tab.4.2 summarizes the results per single subject. When comparing the accuracy result of each subject separately, we can see that there is a significant difference among subjects, across the different models. For example, subject *fat* results in higher accuracy than *icn* regardless of the tested model. Eventually, the best network method – using training on other subjects and recalibration with a combined CNN-LSTM large model, is able to boost the results of the subject with the lowest accuracy to 86%.

In the second stage, we continue to train the model on the rest $3/4$ of the **test subject's** data using a smaller learning rate (0.0001 using RMSProp) for

TABLE 4.1: Average accuracy across all experiments.

model	number of parameter	accuracy per subjects	accuracy all subjects	all but one	all but one after fine tuning
LDA	1375	0.96	0.79	0.65	x
LSTM large	62501	0.82	0.77	x	x
LSTM small	10351	0.89	0.9	x	x
CNN	7924	0.98	0.92	0.84	0.97
LSTM-CNN large	49041	0.93	0.9	x	x
LSTM-CNN small	5511	0.86	0.93	0.84	0.97

TABLE 4.2: Average accuracy per subject.

subject	LDA	LSTM large	LSTM-CNN large	CNN	LSTM small	LSTM-CNN small
fat	1.00	0.98	0.98	0.98	1.00	0.95
gcb	0.91	0.82	0.88	0.92	0.74	0.75
gcc	1.00	0.84	0.92	1.00	0.92	0.97
gcd	0.97	0.80	0.90	1.00	0.76	0.93
gcf	1.00	0.92	0.94	0.95	0.97	0.95
gcg	0.94	0.74	0.96	0.96	0.80	0.87
gch	0.97	0.93	0.96	0.97	0.97	0.96
iay	0.94	0.62	0.92	0.98	0.75	0.86
icn	0.94	0.62	0.86	0.98	0.77	0.77
icr	0.93	0.97	0.98	0.98	0.98	0.98
pia	0.97	0.82	0.94	1.00	0.77	0.81
mean	0.96	0.82	0.93	0.98	0.86	0.89

30 epochs. The second training stage results are presented in columns *CNN and LSTM-CNN all except one fine tune*. The results indicate that as in the other cross-subject evaluation, the LDA accuracy is much poorer than those of the CNN and LSTM-CNN models (65% as opposed to 84%). When we allow calibrating the model for each subject, we achieve an average accuracy of 97% for both CNN and LSTM-CNN.

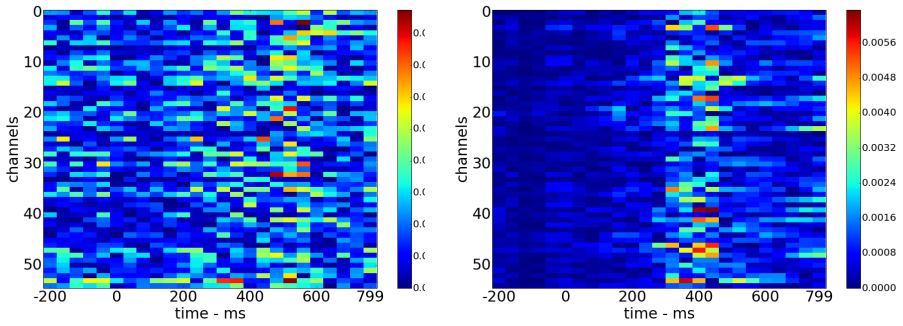
TABLE 4.3: Accuracy when training and testing on different subjects.

subject	LDA all except one	CNN all except one	CNN all except one fine tune	SMALL LSTM-CNN all except one	SMALL LSTM-CNN all except one fine tune
fat	0.94	1.00	1.00	0.98	1.00
gcb	0.43	0.83	0.91	0.86	0.92
gcc	0.79	0.98	0.98	0.95	0.97
gcd	0.66	0.80	0.99	0.83	0.97
gcf	0.68	0.89	0.98	0.79	0.98
gcg	0.52	0.81	0.94	0.77	0.90
gch	0.87	0.97	0.97	0.97	0.99
iay	0.48	0.69	0.98	0.67	0.97
icn	0.44	0.58	0.92	0.61	0.95
icr	0.63	0.81	1.00	0.89	1.00
pia	0.77	0.87	0.96	0.91	0.97
mean	0.65	0.84	0.97	0.84	0.97

Resistance to temporal noise is displayed in Tab.4. LSTM-CNN shows a significant advantage over both LDA and CNN when testing with stimuli onset different than the one used for training. LSTM-CNN-small achieves an accuracy higher by 3% and 6% when adding or removing 40ms to the original stimuli onset, and a t-test indicates that the difference between each pair of groups is statistically significant ($p < 0.05$ - marked in bold). LDA accuracy falls by more than 20% when facing temporal noise.

A possible explanation can be seen when looking at the two network's saliency map (Fig.4.1). In order to investigate the "attention", or the sensitivity of the LSTM model, and compare it to the CNN model, we used a technique suggested by [10] and draw the absolute gradient of the neural network with respect to the input.

If $f(x_1, ..., x_n)$ is a differentiable, scalar-valued function, its gradient is the vector whose components are the n partial derivatives of f, which is a vector-valued function. In our case of $f(x|\theta)$ is the neural network with fixed



(a) CNN (b) LSTM-CNN

FIGURE 4.1: Average gradient across target samples.

weights θ and input x. The partial derivatives of $f(x|\theta)$ with respect to x can be interpreted as "how changing each value of x will change the prediction score". This gradient should not be confused with the gradient used for training, where the goal is to optimize the model parameters θ when x is fixed.

In the case of P300 prediction, x is a matrix of $C \times T$ (C - number of channels, T - number of time steps) and $f(x|\theta)$ is the neural network where θ is the model's weights after training. The gradient $\nabla f(x|\theta)$ (see Eq.4.1) is a matrix with the same size as the input x, where the amplitude of each cell reflects its impact on the function value. Cells with high absolute value can be interpreted as the cells that have a significant influence on the prediction function.

$$\nabla f\left(x|\theta\right) = \begin{bmatrix} \frac{\partial f(x|\theta)}{\partial x(c_1,t_1)} & \dots & \frac{\partial f(x|\theta)}{\partial x(c_1,t_T)} \\ \dots & \dots & \dots \\ \frac{\partial f(x|\theta)}{\partial x(c_C,t_1)} & \dots & \frac{\partial f(x|\theta)}{\partial x(c_C,t_T)} \end{bmatrix} \tag{4.1}$$

The results displayed in Fig.4.1a and Fig.4.1b show the average absolute gradient across all *target* samples of a single cross validation test data: the warm colors correspond to high gradient values, indicating that the model is more sensitive to change in this input feature. We can see the sensitivity of the CNN model spread across the recording relatively evenly as opposed to the LSTM-CNN which is focused around the 250ms and 450ms time-stamps.

Noise	CNN	LSTM_CNN	LDA
-120	0.058	0.044	0.016
-80	0.275	0.299	0.016
-40	**0.825**	**0.864**	**0.565**
40	**0.848**	**0.896**	**0.608**
80	0.335	0.390	0.260
120	0.042	0.042	0.059

TABLE 4.4: Accuracy when introducing temporal noise. Statistically significant results are highlighted in bold.

Chapter 5

Discussion and Future Directions

In this work we examined the use of LSTM neural networks for the task of BCI P300 speller. Despite its temporal nature, no version of LSTM investigated in this work has shown a significant advantage compared to the CNN model suggested by [5]. We did see LSTM results improve with large amounts of data from multiple subjects, and superior results with a combined CNN-LSTM model; moreover, we have shown that this combined model is significantly more robust to temporal noise in the stimuli onset. We also show that the sensitivity of the LSTM based model is much more focused on the area between 250ms to 450ms than CNN based model. This sensitivity is correlated with what we know about the P300 ERP (a peak around 300ms after the stimuli onset). We suggest that the smaller area of sensitivity explains the robustness of the LSTM model to noise in the time domain, since it is less sensitive to the data outside the P300 phenomena.

In our research we found that the effectiveness of 'deep learning' models is seen as we increase our dataset size. Ideally, one could use a large database with many samples but this may be very expensive and impractical. We have shown, that one of the advantages of deep neural networks is their ability for transfer learning between subjects. It can be interesting to explore the network ability to achieve transfer learning between different experiments, task and recording devices, and by that, increase the data set size.

Chapter 6

References

[1] L. Acqualagna and B. Blankertz., "Gaze-independent BCI-spelling using rapid serial visual presentation (RSVP)," in *NeuroImage*, vol. 124, 2013, pp. 901–908.

[2] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from EEG with deep recurrent-convolutional neural networks," *arXiv preprint arXiv:1511.06448*, 2015.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in *IEEE transactions on neural networks*, vol. 5, no. 2. IEEE, 1994, pp. 157–166.

[4] B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Müller, "Single-trial analysis and classification of erp components—a tutorial," *NeuroImage*, vol. 56, no. 2, pp. 814–825, 2011.

[5] H. Cecotti and A. Graser, "Convolutional neural networks for p300 detection with application to brain-computer interfaces," in *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, 2011, pp. 433–445.

[6] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[7] F. Cincotti, A. Scipione, A. Timperi, D. Mattia, A. Marciani, J. Millan, S. Salinari, L. Bianchi, and F. Bablioni, "Comparison of different feature classifiers for brain computer interfaces," in *Neural Engineering*,

2003. Conference Proceedings. First International IEEE EMBS Conference on. IEEE, 2003, pp. 645–647.

[8] P. Davidson, R. Jones, and M. Peiris, "Detecting behavioural microsleeps using EEG and lstm recurrent neural networks," in *Proc. Int. Conf. IEEE Eng. Med. Biol. Soc*, vol. 27, 2005, pp. 5754–5757.

[9] L. A. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," in *Neural Computation*, vol. 70, 1988, pp. 510–523.

[10] A. Graves, "Supervised sequence labelling," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, pp. 5–13.

[11] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández, "Unconstrained on-line handwriting recognition with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2008, pp. 577–584.

[12] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[13] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," in *IEEE transactions on neural networks and learning systems*. IEEE, 2017.

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," in *IEEE Signal Processing Magazine*, vol. 29, no. 6. IEEE, 2012, pp. 82–97.

[15] S. Hochreiter and J. Schmidhuber., "Long short-term memory," *Neural computation*, vol. 9.8, pp. 1735–1780, 1997.

[16] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," vol. 86, no. 11. IEEE, 1998, pp. 2278–2324.

[19] X. D. Li, J. K. Ho, and T. W. Chow, "Approximation of dynamical time-variant systems by continuous-time recurrent neural networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 10, pp. 656–660, 2005.

[20] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain–computer interfaces," *Journal of neural engineering*, vol. 4, no. 2, p. R1, 2007.

[21] R. Manor and A. B. Geva, "Convolutional neural network for multi-category rapid serial visual presentationBCI," *Frontiers in computational neuroscience*, vol. 9, 2015.

[22] B. Obermaier, C. Guger, C. Neuper, and G. Pfurtscheller, "Hidden markov models for online classification of single trial EEG data," *Pattern recognition letters*, vol. 22, no. 12, pp. 1299–1309, 2001.

[23] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[24] A. Rakotomamonjy and V. Guigue, "BCI competition iii: Dataset ii-ensemble of svms for BCI p300 speller," *IEEE transactions on biomedical engineering*, vol. 55, no. 3, pp. 1147–1154, 2008.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, Tech. Rep., 1985.

[26] M. Soleymani, S. Asghari-Esfeden, M. Pantic, and Y. Fu, "Continuous emotion detection using EEG signals and facial expressions," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2014, pp. 1–6.

[27] S. Solhjoo, A. M. Nasrabadi, and M. R. H. Golpayegani, "Classification of chaotic signals using hmm classifiers: EEG-based mental task classification," in *Signal Processing Conference, 2005 13th European*. IEEE, 2005, pp. 1–4.

[28] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems*, 2009, pp. 1601–1608.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[30] T. Tieleman and G. Hinton, "Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning," Technical report, 2012. 31, Tech. Rep.

[31] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[32] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention." in *ICML*, vol. 14, 2015, pp. 77–81.

[33] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.