BCI: Classifying motor task EEG signals using a combination of deep CNN and LSTM architectures

Marvin Martin Etoga Mouroum[1-3 *], **Jonathan Smyth**[1-3 *], **Cesar Gonzalez Cruz**[1-3 *], **and Elisa Ricci**[**]

[1]EIT Digital Masterschool, Autonomous Systems Track
[2]Department for Industrial Engineering. University of Trento, Trento, Italy
[3]Course on Deep Learning, University of Trento, Trento, Italy
[*]These authors contributed equally.
[**]Professor of the course.

EEG data is light weight, portable and cost efficient. It's ability to read brain activity gives high potential for BCI applications. Classifying EEG data using traditional machine learning methods has shown good results, however requires expensive preprocessing of the data. Deep Learning techniques outperform traditional ML methods when using an end-to-end approach. Tan et al. (1) converted EEG signals to 2D images classifying them with regular image recognition techniques. In medical image classification 3D convolution showed to be more reliable than 2D convolution (2). We reconstructed 3D images of the brain activity, passing them to a 3D CNN and as a mini movie to a LSTM. Results showed that this method is computational heavy to execute and does not receive better results than a simple CNN architecture. However this technique could perform better when using 64 electrodes rather than 22, because the 3D image quality would be higher. The CNN-RNN combination received 50 - 60% accuracy, whereas the CNN showed up to 76%.

BCI | CNN | LSTM | RNN | EEG | Deep Learning | Motor Tasks | Course Project | 3D Convolution | EIT Digital

Correspondence: ***marvin.mouroum@studenti.unitn.it, jonathan.smyth@studenti.unitn.it, cesar.gonzalez@studenti.unitn.it***

Introduction

Brain Computer Interfaces (BCI) have the potential to change how humans interact with machines. They provide a more efficient and intuitive interface by using language, gestures, eye-movement and even thoughts. The latter is based on the activity level of the brain. By reading the potential differences in the brain using dry electroencephalography (EEG), motor behaviour imagery can be classified. Classifying EEG data using traditional machine learning (ML) methods has shown promising results (3). However these techniques require expensive pre-processing of the data. Deep Learning (DL) techniques are suitable for creating end-to-end solutions. The lack of free accessible EEG data hinders the creation of powerful DL networks. However because EEG data can be seen as a video stream of activity levels of the brain, a mixture of CNN and RNN seams to be applicable (4). Tan et al. (1) projected the activity of the electrodes into 2D images, resulting in a video of the brain activity. Historically for high resolution image classification 3D convolution layers have been shown to yield better results (2). We explore the possibility to classify a CNN-LSTM artificial neural network on reconstructed 3D images based on the brain activity levels measured with EEG. This way the time dependency, as well as the spatial characteristics of the brain activity can be taken into account while classifying the the thoughts. With the emergence of additional EEG datasets, the importance of DL classifiers for BCI applications rises. This paper explores a unique way of classifying EEG data.

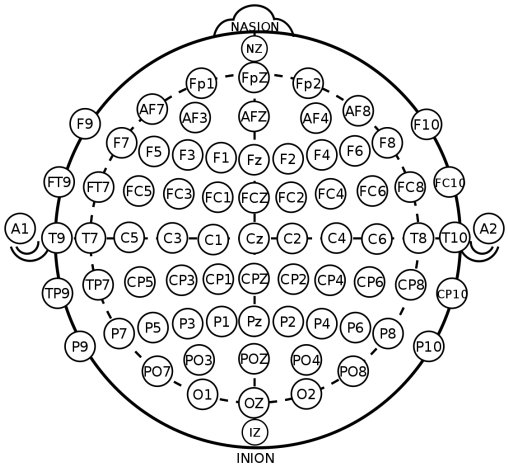

Fig. 1. Brain Computer Interface. BCI can be used to controll machines more efficient then ever before.

Methodology

Dataset. Datasets for DL applications are rare for EEG measurements. This is because experiments are expensive. Usually a EEG session with a single subject may contain up to a million different labeled data points. However the variation between subjects is quite high. Different brains seem to operate in different manners. Since most datasets only contain a few subjects, the overall generalization is not ideal for real life applications and a calibration of the classifier would be necessary. Aznan et al. (5) showed that shallow CNN architectures could outperform deep architectures because of the lack of data in EEG applications. A new database appeared in 2018 (6), which contains 60 hours of different EEG experiments across 13 participants. The experiments can be classified in 4 groups:

1. CLA - Classic

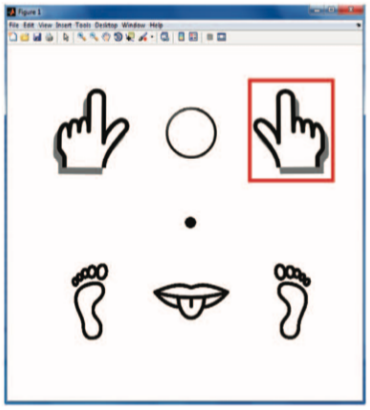

Fig. 2. Experimental set up for CLA data

2. HaLT - extended CLA
3. 5F - finder movements
4. FreeForm - arbitrary movements
5. NoMT - similar to HaLT

For this paper the simplest experimental design - the CLA - was chosen. The subjects stare at a screen, waiting for visual queues to act on. Possible states are to move the left hand, right hand, nothing or no impulse at all Fig. 2.
The dataset comes in the form of raw EEG data waves. During pre-processing the data was centered and zero meaned. Additionally, experimental breaks, the time before the official start and end phases of the experiment were erased. The positions of the 22 electrodes were reconstructed assuming the head is a 3-dimensional ellipsoid Fig. 1 with dimensions: a = 72,5 cm, b = 100 cm, c = 56 cm.

$$\begin{aligned} x &= a \cdot \cos\theta \cdot \sin\phi \\ y &= b \cdot \sin\theta \cdot \sin\phi \\ z &= c \cdot \cos\theta \end{aligned} \tag{1}$$

Assigning a voxel to every cm^2 a 3D-image for every time step, a 3D video was reconstructed. The activation of the electrode at it's position in space was converted into a gray-scale image and the neighbouring parts were interpolated to have a weaker activation - radiating from the the position of the electrode Fig. 3.
There were 15 CLA experiments with roughly 700.000 data points at a sampling rate of 200. With over 10 million 3D images with 400 Gbyte of memory after the 3D video creation. Half of the data was used for testing and the other half for validation.

Architecture. EEG signals contain spatial as well as time dependent information since the activity of a brain region is a constant flow of alternating electrical potentials. At a sufficient sampling rate the time dependency can be evaluated. To leverage both, spatial and time dependent characteristics, a CNN-LSTM architecture is used. A larger batch of images is convoluted and formed into a smaller batch of feature

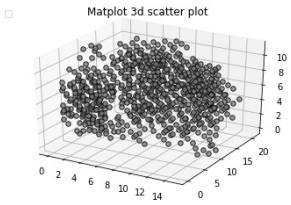

Fig. 3. Example plot for brain activation. 22 electrodes and their positions have been reconstructued and their activation value has been plotted in a gray scale between 0 and 1. A voltage value of 0 is equivalent to a 0.5 gray scale value. Around the 22 positions, interpolated values have been added, which decrease over time by 1/r from the heat point. A total of 708 activation points, which contribute 16% to the overall 12x16x22 image, are plotted.

representation sequences. This batch of sequences is then classified by the LSTM network.

Convolutional Neural Network. Different CNN architectures have been developed to classify imagery motor tasks, such as Conv1D (7), Conv2D-LSTM (4), Conv2D-GRU (1) and LSTM.
This paper examines a new approach using a Conv3D-LSTM architecture. Prior to connecting the Conv3D net to the LSTM a classifier using only the CNN was tested with a result of 72 % accuracy across all subjects. These results are comparable with the literature (7),(1). Relu activation function is used across all layers with batch normalization. 1x1x1 convolution is used repeatably with max pooling layers. Weights were initialized with Xavier normalization. The 12x16x22 input image is convoluted into a 1x1x22 tensor.

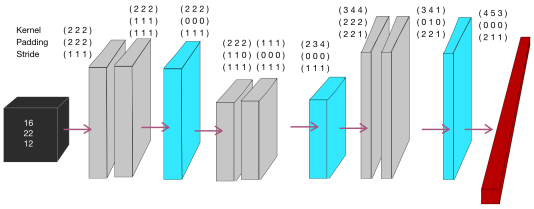

Fig. 4. CNN encoder architecture A 9 Layer 3D-CNN with max pooling (Blue) and output representation of the form 1x1x22

The asymmetric input shape of the 3D-image requires asymmetric patting and step patterns, as well as asymmetric kernel sizes. Avoiding cubic inputs minimizes storage volume of the data and enables higher batch sizes. The presented architecture is convoluting to a 1x1x22 tensor, to directly compare the novel method to a regular LSTM connected directly to the sensor output data of the EEG apparatus. Periodic pooling used to reduce dimensionality. The kernel size is expected to be less important than in regular image classification architectures, because the electrodes are not closely arranged. The raw CNN architectures were compared with more and less intermediate channels, 22 and 9, respectively, however 22 proved to be more efficient.

Recurrent Neural Network. Approaching the problem from another angle resulted in the looking into recurrent neural networks as an architectural design idea. It was decided that based on the sequential manner of the data and that the clas-

sification of a motor movement was not based solely on one explicit reading of EEG signal at one given instant in time, rather a series of readings equated to a thought about motor movement, for this reason an RNN style network was decided upon. The idea was to extract the temporal features from the data set and use this to try and predict when a subject is about to think about doing some motor movement.

Initially the idea of a RNN was to be adopted, but given the nature of the size of the data sets and taking into consideration the idea to move forward with the project in a real world scenario, a simple RNN was thrown out due to such problems as vanishing/exploding gradient which is a common problems in RNNs. This coupled with the fact that to train the large data sets we need to back propagate through time could also result in very expensive computational needs (even truncated back propagation through time).

Moving on from these downsides, the LSTM approach was adopted to mitigate the issue of exploding/vanishing gradients and issues surrounding BPTT. A Simple LSTM approach was adopted, with input size of 22(as there were 22 channels from the electrodes) and hidden size of 6 (due to the limit amount of data). The hidden and cell states are both initialized with random numbers to begin with. Using the data loader, each entire subject was split up into mini batches, these mini batches were then broken down further into tensors of shape [time sequence, batch time sequence, channels] this is then what is fed into the LSTM network, then this is reconstructed sequentially upon output to match up with the labels.

Training. The CNN is pretrained on the dataset with a test accuracy of 71%. After training the fully connected layer is removed and the LSTM is appended to the 1x22 dimensional feature vector output from the CNN. The CNN-LSTM network is trained with a cyclic learning rate $\epsilon = |\cos(i)e^{-4}|$ and $ADAM$ optimizer 1. The size of the entire dataset is roughly 400 Gbyte. Mini Batches of 4*200 samples are chosen, which is equivalent to 4 seconds of recording. The CNN will receive the entire batch for processing, whilst the LSTM will receive a batch of 4 sequences of length 200, which is equivalent to 4 sequences of 1 second recording. Loading the files is slow and the learning process therefor expensive. Additionally the sequences were selected in a manner that they start with a zero and end with a classifier value from 0 to 3. This way every sequence can be attributed to exactly one value - the last value of the sequence.

The LSTM was trained by itself with a wide range of different hyper parameters and variable sizes of time sequences. Two approaches were conducted, matching all labels to all individually data points and a contrasting matching of labels to an entire time sequence. In the end matching a single label to a group of organized time sequences was opted for as this is more real world scenario.

The model was trained in Google Colab with the PyTorch framework.

LR	Optim.	Mom.	WD	BS	Eps
$\|\cos(i)e^{-4}\|$	$ADAM$	0.9	e^{-9}	$4 \cdot 256$	15

Table 1. Hyperparemeters for training of the CNN-LSTM architecture. LR: Learning Rate, Optim: Optimizer, Mom: Momentum, WD:Weight Decay, BS: Batch Size, Eps: Epochs

Results

The CNN and LSTM classified every time step, whereas the combination classified the result of a sequence containing 200 time steps. This reduces the amount of targets labelled with 0 = 'no task given'. In general the label 0 dominates the dataset. Each subject is suspect to another score by the network. The tables below give an overview of the results.
In general training time was the fastest with the LSTM and the longest with the CNN-LSTM combination. The long training time is due to the fact that the CNN uses the 3D reconstructed data, which is massive in size and needs to be loaded from the drive in small chunks while training. Otherwise RAM would explode and training would not be possible on Google Colab. The LSTM was trained directly on the raw data.

	Accuracy.	Loss
Subject0	71.04	0.00409
Subject1	73.81	0.00391
Subject2	73.49	0.00394
Subject3	72.39	0.00374
Subject4	73.79	0.00393
Subject5	73.87	0.00378
Subject6	72.40	0.00311
Subject8	71.01	0.00412
Subject9	74.02	0.00391
Subject10	73.91	0.00388
Subject11	72.41	0.00378
Subject12	73.93	0.00390
Subject13	74.91	0.00388
Subject14	76.49	0.00324
Subject15	74.59	0.00389
Subject16	74.19	0.00391

Table 2. Accuracy for each subject for the single 3D CNN

Discussion

Our method uses a great amount of pre processing CPU time of the raw data in order to convert it into a 3D images. Additionally the RAM usage is 10 GBytes for a batch of 800 images, which effectively is a batch of 4 sequences. The overall dataset size exploded to be 400/,GBytes in size and therefore could not be loaded into the ram, but needed to be dynamically loaded from the cloud storage. Storage is expensive and the run time is extremely diminished. Testing and training therefor is extremely slow and takes about 1 hour per epoch. Although theoretically we were hoping for better results, the results do not differ much from a regular LSTM trained on the raw data, or an CNN trained on the converted data. Both options are faster, although the LSTM on raw data is the fastest

	Accuracy.	Loss
Subject0	71.44	0.000637198
Subject1	71.31	0.000637198
Subject2	70.31	0.000668321
Subject3	71.95	0.000644332
Subject4	70.84	0.000666331
Subject5	71.02	0.000666341
Subject6	72.37	0.006632123
Subject8	72.37	0.006632123
Subject9	70.14	0.000723402
Subject10	70.14	0.000621102
Subject11	70.84	0.000637130
Subject12	70.11	0.000737171
Subject13	71.46	0.000658722
Subject14	69.87	0.005666654
Subject15	70.01	0.006586622
Subject16	70.17	0.006371292

Table 3. Accuracy for each subject for the single LSTM The results for the LSTM only network all converge around the same range, between 69-72, this was running all tests on 20 epochs. In some cases, the network would converge to the shown results much quicker than others. With an average time of 3 minutes per epoch to run through 666700 data time steps.

	Accuracy.	Loss
Subject0	46.51	0.31965
Subject1	45.14	0.32307
Subject2	43.06	0.32827
Subject3	44.47	0.32473
Subject4	43.14	0.32806
Subject5	54.02	0.30086
Subject6	54.18	0.30058
Subject8	54.69	0.29919
Subject9	54.09	0.30070
Subject10	52.89	0.30369
Subject11	53.91	0.30114
Subject12	63.79	0.27643
Subject13	54.75	0.29904
Subject14	58.90	0.28867
Subject15	58.47	0.28974
Subject16	58.83	0.28884

Table 4. Accuracy for each subject for the CNN-LSTM. The accuracy has been computed by counting up the times the classification was correct and then divide by the sequences used for evaluation. This is different to the pure LSTM and CNN comparison because sequences were used for evaluation instead of single values in time. The target vector originally containing 800 entries was reshaped to 4*200 entries representing the sequences fed to the network. Only the last values of the targets per sequence was used to compute the loss. Equally the network output was reshaped in the same manner to compute loss and accuracy. This has been done to use PyTorch's LSTM module in a 'Many to One' relationship. Because of time constrains the training epochs for the this network was smaller than the ones before.

option.

Comparing the 3D CNN trained on the pre processed data with the LSTM trained on the raw data shows an improvement of 2,6 % on average across all subjects. The contribution of the spatial character to the classification seems to be a better classifier than the time dependency. The difference however is not big enough to make up for the cost in GPU time used to compute the results.

The resolution of the images might not be high enough since the data set only contains 22 sensors instead of 64 in other data sets (1). Unless the correct hardware is locally available, we would not promote using out method on cloud services like Google's Colab. We encountered several small errors when analyzing the dataset. It has no citations yet so we cannot be sure that the data is correctly labeled. Perhaps it might be interesting redoing this project on new upcoming data sets. The goal is to classify a movement before it happened by reading the brain activity. Time dependency taken into account does not provide a better classifier than a simple CNN. The combination of both is more difficult to train and provides no upgrade in performance.

Bibliography

1. Chuanqi Tan, Fuchun Sun, Wenchang Zhang, Jianhua Chen, and Chunfang Liu. Multimodal classification with deep convolutional-recurrent neural networks for electroencephalography. *Lecture Notes in Computer Science*, page 767–776, 2017. ISSN 1611-3349. doi: 10.1007/978-3-319-70096-0_78.
2. Jens Kleesiek, Gregor Urban, Alexander Hubert, Daniel Schwarz, Klaus Maier-Hein, Martin Bendszus, and Armin Biller. Deep mri brain extraction: A 3d convolutional neural network for skull stripping. *NeuroImage*, 129:460 – 469, 2016. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2016.01.024.
3. Hafeez Ullah Amin, Wajid Mumtaz, Ahmad Rauf Subhani, Mohamad Naufal Mohamad Saad, and Aamir Saeed Malik. Classification of eeg signals based on pattern recognition approach. *Frontiers in Computational Neuroscience*, 11:103, 2017. ISSN 1662-5188. doi: 10.3389/fncom.2017.00103.
4. P. Xia, J. Hu, and Y. Peng. EMG-Based Estimation of Limb Movement Using Deep Learning With Recurrent Convolutional Neural Networks. *Artif Organs*, 42(5):E67–E77, May 2018.
5. Nik Khadijah Nik Aznan, Stephen Bonner, Jason D. Connolly, Noura Al Moubayed, and Toby P. Breckon. On the classification of ssvep-based dry-eeg signals via convolutional neural networks. *CoRR*, abs/1805.04157, 2018.
6. Murat Kaya, Mustafa Kemal Binli, Erkan Ozbay, Hilmi Yanar, and Yuriy Mishchenko. A large electroencephalographic motor imagery dataset for electroencephalographic brain computer interfaces. *Scientific Data*, 5:180211, 10 2018. doi: 10.1038/sdata.2018.211.
7. Nik Khadijah Nik Aznan, Stephen Bonner, Jason D. Connolly, Noura Al Moubayed, and Toby P. Breckon. On the classification of ssvep-based dry-eeg signals via convolutional neural networks. *CoRR*, abs/1805.04157, 2018.