Faculty of sciences

Epileptic seizure prediction using deep leaning

Iryna Korshunova

Master dissertation submitted to
obtain the degree of
Master of Statistical Data Analysis

Promoter: Dr. Pieter Buteneers
Co-promoter: Prof. Dr. Olivier Thas
Department of Electronics and Information Systems

Academic year 2014 - 2015

Foreword

This diploma thesis is entirely the result of my own work, I mentioned all the sources of information I used.

Data was obtained under the terms of American Epilepsy Society Seizure Prediction Challenge [1], which does not prohibit the use of the data for education. At the time of writing the thesis, test labels were kept confidential, and we received them only for thesis purposes. In the future, competition organizers will make all the data and source code of the top 10 models available on the *ieeg.org* portal.

First of all, I would like to express my immense gratitude to my promoters, Dr. Pieter Buteneers and Dr. Olivier Thas, for their support, guidance and encouragement. Without Dr. Buteneers I would not have been able to start and finish this dissertation. I am very grateful for the meetings I had with Dr. Thas: each of them used to give me a burst of energy for doing great statistical things. I would also like to thank Benjamin Brinkmann, one of the competition organizers, who kindly provided additional data for the experiments. A special acknowledgement goes to my friend Juan Pablo Carbajal, who used to admonish me from doing "mindless number-crunching". I would like to give my deepest thanks to Sander Dieleman; because of him I am fond of convolutional neural networks and Theano. Thank you to Thibault Verhoeven, Jeroen Burms and Juan Pablo Carbajal (once again) for proofreading and valuable remarks. Last but not least, I am grateful to Matthijs Vynck and Alain Visscher for making an attempt to understand my fuzzy explanations about spectrograms and to help me, when I got stuck with exploratory data analysis.

[1] http://www.kaggle.com/c/seizure-prediction/rules

Contents

1 Summary

Epilepsy is currently one of the most commonly diagnosed neurological disorders, which is characterized by the occurrence of spontaneous seizures. About 1 out of 3 patients has a drug-resistant epilepsy, which means that seizures cannot be controlled by medication. In such cases, seizure forecasting systems are vitally important, since they allow patients to avoid dangerous activities and bring themselves in a safe environment before a seizure starts (Buteneers, 2012).

Electroencephalogram is a signal regularly used for diagnosing epilepsy and predicting or detecting seizures. Despite a significant hardware progress, which enables to get high-quality EEG signals, practitioners still lack reliable algorithms for identifying periods of increased probability of seizure occurrence. In an attempt to boost the development of better models, leading epilepsy research institutions organized an international competition: "American Epilepsy Society Seizure Prediction Challenge". The goal was to develop a method, able to classify ten minute long data clips of a pre-seizure brain activity covering an hour prior to a seizure onset and ten minute clips of inter-seizure activity.

Our approach to the seizure prediction problem was based on convolutional neural networks – a special class of multi-layer neural networks. We constructed several architectures in a way the network could learn various types of time-localized features on several levels of representation and afterwards combine the information across time to make a prediction for a data clip. Our design decisions regarding the architecture were crucial for networks to perform well. This allowed us to finish in the 10th place out of 504 teams.

We also performed the analysis of other solutions; with special attention we explored linear classifiers, which appeared to beat more complex methods in terms of the evaluation metric used in the competition. At the moment, we cannot draw any firm conclusion about the superiority of some methods over others, because many factors have to be considered when applying these models in practice.

There was a significant gap between the competition and real life settings. However, its results are promising and can serve as a solid ground for further research.

2 Introduction

2.1 Epileptic seizure prediction

Epilepsy is one of the most common neurological disorders, which affects nearly 1% of the world's population, and it is characterized by the occurrence of spontaneous *seizures*. A form of the seizures may vary from a brief lap of attention to a severe convulsion. In 60-80% cases seizures can be prevented by using anticonvulsant drugs, however for the remaining patients medications are not effective and additionally may have unwanted side effects (Buteneers, 2012). Therefore, people with epilepsy have significantly lower quality of life due to anxiety associated with a seizure possibility. They have higher probability of injures and sudden death, limited independence, driving restrictions and difficulties with finding and keeping a job. Seizure forecasting systems aim to help patients with epilepsy lead better lives: they would be able to avoid potentially dangerous activities e.g. swimming, driving and could intake medications only when needed. Obviously these systems need highly reliable algorithms to detect periods of increased probability of a seizure to come.

Because seizures are caused by electrical discharge of the brain cells, electroencephalogram (EEG) is a common signal to analyze in systems for seizure detection or prediction. EEG is a multichannel recording of the brain's electrical activity, where EEG electrodes are placed on the scalp or invasively in the brain (intracranial EEG, iEEG). As a result of greater proximity to neural activity, iEEG has higher spatial resolution and signal-to-noise ratio than scalp EEG, so it is more valuable for epilepsy research.

Multiple studies support the hypothesis of an existing *preictal* state (prior to seizure), which is associated with stereotypic EEG waveforms and spectral patterns (Howbert et al., 2014). Nevertheless, possibility of seizure forecasting remains a very difficult problem and to some extent controversial due to statistical flaws in many studies (Mormann et al., 2007). In the pursuit of an objective evaluation of various algorithms, International Epilepsy Electrophysiology Portal was initiated. It will enable researchers and paper reviewers to run competing algorithms on the same data sets and directly compare the results. As a part of this effort, National Institutes of Health, the Epilepsy Foundation, and the American Epilepsy Society organized an international competition "American Epilepsy Society Seizure Prediction Challenge". Its goal was to identify the best model for discriminating *preictal* vs. *interictal* (between seizures) iEEG clips. The competition had a prize fund of 25000$ and was hosted on Kaggle platform (Kaggle, 2014). It was running from August 25th, 2014 to November 17th, 2014. The model presented in this thesis finished in the 10th place out of 504 teams.

2.2 Data

iEEG data for the competition was recorded from 5 dogs with the naturally occurring epilepsy and 2 humans undergoing iEEG monitoring. For canine subjects, iEEG signal was

sampled from 16 electrodes at 400 Hz and voltages were referenced to the group average. These recordings span multiple months up to a year and contain up to a hundred seizures in some dogs. A seizure advisory system as on Fig.1 is based on the implantable device (A). It acquires 16 channels of iEEG (B) and wirelessly transmits the data to the personal advisory device, which stores it on a flash drive and uploads weekly via the internet to a central data storage site. (C) shows a temporal profile of a seizure probability and threshold defining a period of increased seizure probability (preictal state). When the forecasted probability exceeds the defined threshold a warning is triggered (Howbert et al., 2014).

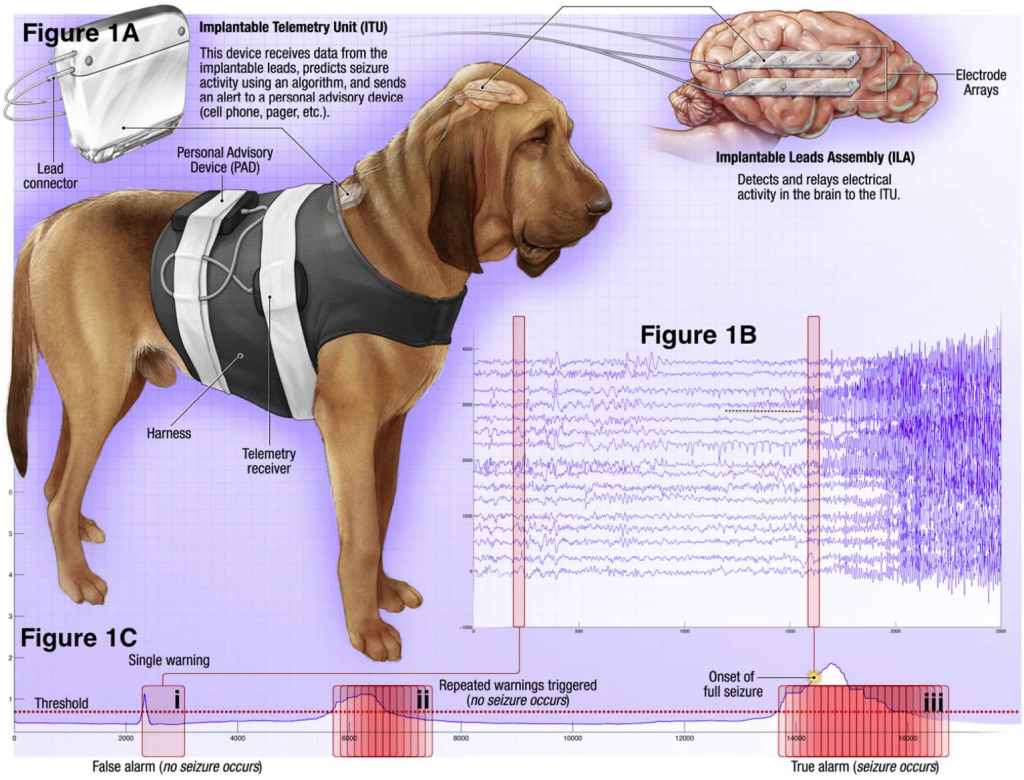

Figure 1: Seizure advisory system in canines with epilepsy (Howbert et al., 2014)

The sample rate for human patients was 5000 Hz, the recorded voltages were referenced to an electrode outside the skull; monitoring period for was up to a week.

Seizures are known to occur in groups, which implies little benefit from forecasting follow-on seizures, so data included only leading seizures, defined as seizures occurring four hours or more after another seizure. Interictal data segments were chosen at random from the whole iEEG recording with the following restriction: segments for dogs had to be at least one week before or after any seizure and for humans – at least four hours (Kaggle, 2014).

Data is organized into ten minute long clips of preictal and interictal activity. Training data is grouped into one hour sequences, for the test clips their timing is unknown. Preictal period starts one hour prior to seizure onset with a five minute horizon: from 1:05 to 0:05 before seizure as shown on Fig. 2. This ensures that seizures could be predicted with enough time

ahead to allow administration of fast-acting medication.

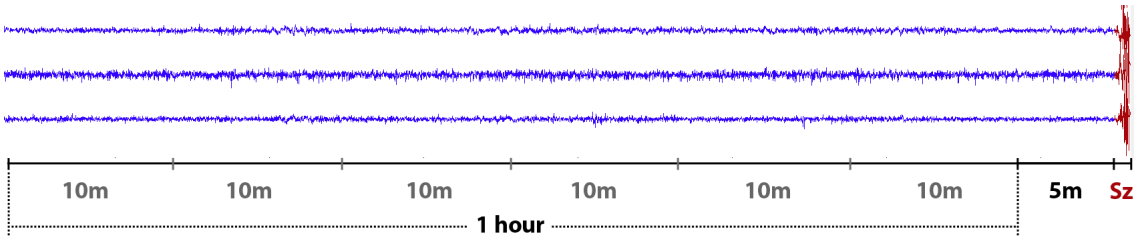

Figure 2: Example of iEEG signal (only 3 channels), showing 1 hour and 5 minutes before the seizure onset (Kaggle,2014)

The number of 10 minute interictal/preictal clips for each subject in train and test sets is given in Table 1 (during the competition, this split for the test set was unknown).

For canine subjects 16 channels were recorded, except Dog_5, who has 15 channels. Number of electrodes for humans was dictated by the clinical problem, so Patient_1 and Patient_2 had 15 and 24 channels respectively.

Set \ Subject	Dog_1	Dog_2	Dog_3	Dog_4	Dog_5	Patient_1	Patient_2
train	480/24	500/42	1440/72	804/97	450/40	50/18	42/18
test	478/24	910/90	865/42	933/57	179/12	183/12	136/14

Table 1: Number of data clips (interictal/preictal) in train and test sets

2.3 Model evaluation

The required output was a probability of the preictal state for a given data clip. To evaluate the quality of the models organizers used area under ROC curve (AUC), using a single curve across all 7 subjects.

ROC curve (Fawcett, 2004) plots true positive rate (proportion of actual positives which are correctly classified) against the false positive rate (proportion of falsely classified negatives) at various thresholds used in a decision rule. The decision rule assigns a positive label (one) if the classifier's output is above the theroshold and negative label (zero) otherwise. For instance, classifier gives the following probabilities: $P(Y = 1|x_1) = 0.4$ and $P(Y = 1|x_2) = 0.6$. If we take the threshold of 0.2, both observations will be labelled as positives; with 0.5 threshold x_1 will be classified as negative and x_2 as positive.

The area under the curve is the probability that a classifier ranks a randomly chosen positive instance higher than a randomly chosen negative one. A random classifier has an AUC of 0.5, whereas the maximum is 1.0.

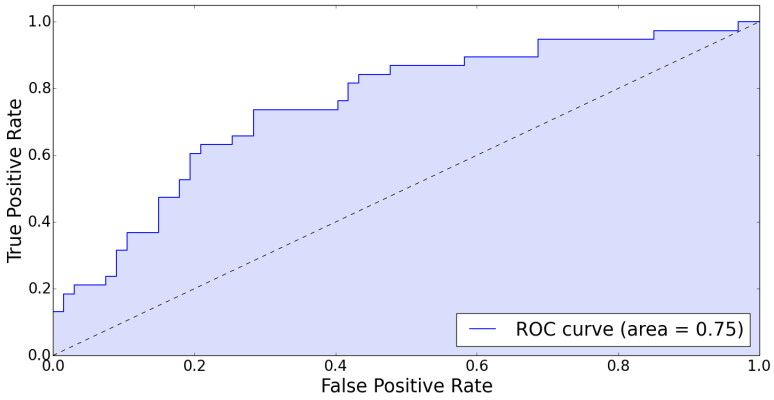

Figure 3: An example of a ROC curve

Seizure activity is highly individual, so models are mostly tuned for each patient separately. Therefore, AUC across combined per subject predictions is a tricky evaluation metric. Even if per subject models have maximal AUC, the combined score may be worse since the predictions from per subject classifiers have different optimal (based on certain criterion) thresholds. For example, for the first subject AUC=1.0 if we have the following pairs of predictions and class labels: (0.1, 0), (0.2, 0), (0.3,1), (0.4, 1). For the second patient with (0.6,0), (0.7,0), (0.8, 1), (0.9, 1) AUC is also 1.0. However, if we combine the predictions, AUC becomes 0.75. So in order to have high leaderboard score, calibration of predictions becomes as important as building good models. However, models, that need less calibration, are more robust: their performance is more independent of the threshold. It is valuable for practical usage, because by choosing a threshold on a validation set, we cannot guarantee its optimality for a new data.

Organizers allowed the use of the test data to calibrate the predictions from per subject classifiers. This gave rise to many dirty tricks, which are unsuitable in real world problems as seizure prediction devices do not have access to the future data during training.

When one does the submission with test predictions, the Kaggle platform computes two scores: a public score, calculated on approximately 40% of the evaluation data, and a private score based on the other 60%. While public scores are revealed immediately, the private score determines the final rankings and is disclosed after the competition finishes. Participants do not know to which of two sets each test example belongs to, so direct optimization of the private score is impossible. While the competition is active, participants can choose 2 submissions; the best of them will contribute to the final leaderboard standings.

3 Methods and Results

In this chapter we are going to describe several machine learning techniques we used to tackle the problem of seizure prediction. Machine learning is a set of automated methods of data analysis for uncovering patterns in data. From a probabilistic perspective, machine learning is closely related to statistics, however they are somewhat different in terminology and emphasis (Murphy, 2012). According to a well-known statistician Robert Tibshirani, there is a following amusing correspondence between machine learning and statistical terms: network, graphs – model; weights – parameters; learning – fitting; generalization – test set performance; supervised learning – regression/classification; unsupervised learning – density estimation, clustering; large grant=1M$ – large grant=50K$ [2].

We will mostly concentrate on deep learning techniques and present a novel architecture of a convolutional neural network we used for the competition. Moreover, several linear models will be also explored in order to gain better understanding about the problem.

3.1 Linear classifiers

3.1.1 Linear discriminant analysis

In our problem we are dealing with two types of data clips: preictal (positive class) and interictal (negative class), so further we will describe linear discriminant analysis (LDA) for two groups. LDA assumes that $\boldsymbol{X}$ [3], a multivariate random variable, within each group has a multivariate normal distribution with different means but equal variance-covariance matrices:

$$\boldsymbol{X}|Y=1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

$$\boldsymbol{X}|Y=0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}).$$

We shall denote probability density functions in both groups as $f_1(\boldsymbol{x})$ and $f_0(\boldsymbol{x})$, so for a given observation $\boldsymbol{x}$, log-likelihood of being from a positive class vs. negative is:

$$\log\frac{f_1(\mathbf{x})}{f_0(\mathbf{x})} = \log\frac{\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1))}{\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0))} = \mathbf{x}^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0) - \frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_0),$$

which means that decision boundary is linear in the space of x. An observation x is classified as a positive when $\log\frac{f_1(x)}{f_0(x)} > \log\frac{\pi_0}{\pi_1}$, however we are interested in probability that x is a preictal clip:

$$P(Y=1|\mathbf{x}) = \frac{\pi_1 \cdot f_1(\mathbf{x})}{\pi_1 \cdot f_1(\mathbf{x}) + \pi_0 \cdot f_0(\mathbf{x})},$$

[2] http://statweb.stanford.edu/ tibs/stat315a/

[3] Hereafter, we will use bold upper-case letters for multivariate random variables or matrices, which is clear from the context. Bold and lower-case letters will denote column vectors.

where π_1 and π_0 are the prior probabilities of preictal and interictal classes ($\pi_0 + \pi_1 = 1$, so we can use $\pi_1 = \pi$ and $\pi_0 = 1 - \pi$).

In LDA we aim to find the joint distribution of $\mathbf{X}$ and Y, which is parameterized by $\boldsymbol{\theta} = \{\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}\}$. To find the parameter estimates one needs to maximize the loglikelihood function $\ell(\boldsymbol{\theta}; D) = \sum_{i=0}^{N-1} \log(P(y_i, \mathbf{x}_i|\boldsymbol{\theta}))$ with respect to $\boldsymbol{\theta}$, where $D = \{(\mathbf{x}_i, y_i)\}_{i=0..N-1}$ is a set of training pairs of input feature vectors $\mathbf{x}_i$ and target labels y_i, containing N_1 preictal examples and N_0 interictal examples. The process of maximum likelihood estimation (MLE) yields the following estimates:

$$\hat{\pi} = \frac{N_1}{N_1 + N_0},$$

$$\hat{\boldsymbol{\mu}}_1 = \frac{1}{N_1} \sum_{y_i=1} \mathbf{x}_i, \; \hat{\boldsymbol{\mu}}_0 = \frac{1}{N_0} \sum_{y_i=0} \mathbf{x}_i,$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N-2} [\sum_{y_i=1} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_1)^t \cdot (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_1) + \sum_{y_i=0} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_0)^t \cdot (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_0)].$$

The term $\mathbf{x}^t \boldsymbol{\delta} = \mathbf{x}^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ is called a discriminant function. We can get a similar $\boldsymbol{\delta}$ (up to a scaling factor) by explicitely searching for a projection vector that maximizes the separation between classes. Thus, by analysing $\boldsymbol{\delta}$ coefficients we can gain an insight how preictal and interictal clips differ.

3.1.2 Logistic regression

Logistic regeression is a generalized linear model with the following components: the distributional assumption is $Y|\mathbf{x} \sim Bernoulli(\pi(\mathbf{x}))$, where $\pi(\mathbf{x}) = P(Y = 1|\mathbf{x}) = E(Y|\mathbf{x})$; and linear predictor $\alpha + \boldsymbol{\beta}^t \mathbf{x}$, which is linked to the expected value $E(Y|\mathbf{x})$ with a logistic function, so $P(Y = 1|\mathbf{x}) = \frac{\exp(\alpha + \boldsymbol{\beta}^t \mathbf{x})}{1 + \exp(\alpha + \boldsymbol{\beta}^t \mathbf{x})}$. Parameters $\boldsymbol{\theta} = \{\alpha, \boldsymbol{\beta}\}$ can be estimated with maximum likelihood estimation based on Bernoulli distribution for Y, so the likelihood function is $\ell(\boldsymbol{\theta}; D) = \sum_{i=0}^{N-1} \log(P(y_i|\mathbf{x}_i, \boldsymbol{\theta}))$. In general there is no analytical solution for $\hat{\theta}$, so one needs to use numerical optimization techniques. Normally, any reasonable algorithm can find the maximum of $\ell(\boldsymbol{\theta}; D)$, however, in limited number of cases, e.g. complete separation, MLE does not exist (solution is infinite).

Gradient descent is an iterative optimization method, which attempts to find a minimum by following the steepest descent direction given by a negative gradient of the objective function. Equivalently, to maximize the likelihood, it takes steps proportional to the gradient at a current point $\boldsymbol{\theta}_n = \{\alpha_n, \boldsymbol{\beta}_n\}$:

$$\alpha_{n+1} \leftarrow \alpha_n + \eta \frac{\partial \ell}{\partial \alpha_n},$$

$$\boldsymbol{\beta}_{n+1} \leftarrow \boldsymbol{\beta}_n + \eta \frac{\partial \ell}{\partial \boldsymbol{\beta}_n},$$

where initial point $\boldsymbol{\theta}_0 = \{\alpha_0, \boldsymbol{\beta}_0\}$ and step size (learning rate) η should be specified. With particular choices for η this procedure converges to the global maximum of $\ell(\boldsymbol{\theta}; D)$.

3.2 Deep learning models

Deep learning is an area of machine learning, which specializes on learning data representation at different levels of abstraction using models with multiple non-linear transformations. Nowadays, it provides state-of-the-art solutions for many problems in computer vision, speech recognition, natural language processing, etc. Deep learning can be seen as a rebranding of neural networks, some types of these models we are going to discuss further.

3.2.1 Artificial neural networks

Artificial neural networks (ANN) are biologically-inspired models, which have been a subject of active research for many decades. A feedforward ANN consists of multiple layers, whose purpose is to extract various levels of data representation, which are relevant for a particular problem. An example of a feedforward ANN with N layers is given on Fig.4.

Each ith unit (neuron) in nth hidden layer computes a weighted sum of its inputs, followed by a nonlinearity f (activation function) as shown on Fig.5. The output of the nth layer is the following vector: $\mathbf{x}^n = f(\mathbf{W}^n \cdot \mathbf{x}^{n-1} + \mathbf{b}^n)$, where $\mathbf{x}^{n-1}$ is the input to the nth layer, $\mathbf{W}^n$ is a matrix of weights and $\mathbf{b}^n$ is a vector of biases. The network's input and output are denoted as $\mathbf{x}^0$ and $\mathbf{x}^N$ respectively.

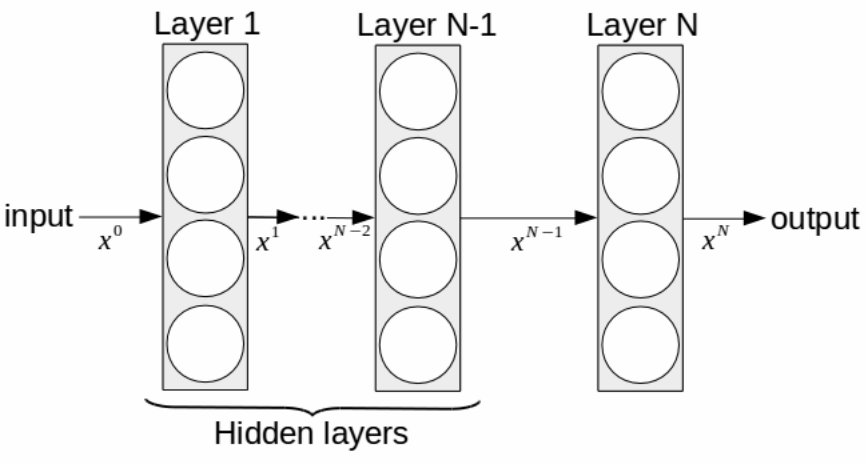

Figure 4: A schematic of a feed-forward ANN

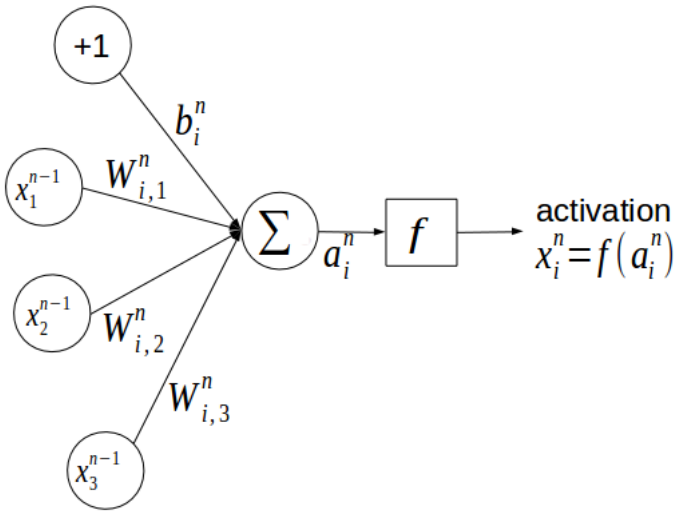

Figure 5: A schematic of an artificial neuron

In multiclass classification problems, when we are interested in class membership probabilities, softmax is commonly used as an output layer:

$$\mathbf{x}_i^N = P(Y = i|\mathbf{x}^0;\boldsymbol{\theta}) = \frac{\exp(\mathbf{W}_i^N \cdot \mathbf{x}^{N-1} + \mathbf{b}_i^N)}{\sum_j \exp(\mathbf{W}_j^N \cdot \mathbf{x}^{N-1} + \mathbf{b}_j^N)},$$

where Y is a stochastic variable for a class label, $\mathbf{x}_i^N$ is a unit in the output layer associated with ith class, $\mathbf{W}_i^N$ and $\mathbf{b}_i^N$ are weights and bias corresponding to $\mathbf{x}_i^N$, $\boldsymbol{\theta}$ is a model parameterization, When we have 2 classes, softmax reduces to logistic regression with one output:

$$\mathbf{x}^N = P(Y = 1|\mathbf{x}^0;\boldsymbol{\theta}) = \frac{\exp(\mathbf{W}^N \cdot \mathbf{x}^{N-1} + \mathbf{b}^N)}{1 + \exp(\mathbf{W}^N \cdot \mathbf{x}^{N-1} + \mathbf{b}^N)},$$

Training of a neural network aims to modify the parameters of the model, so that the output layer can produce the desired probabilities. Learning the parameters involves minimization of a cost function, which in case of softmax output is usually negative log-likelihood (NLL): $\ell(\boldsymbol{\theta};D) = -\sum_{i=0}^{|D|} \log(P(Y = y^{(i)}|x^{(i)},\boldsymbol{\theta}))$

In order to minimize $\ell(\boldsymbol{\theta};D)$ we can again use gradient descend:

$$\mathbf{W}^n \leftarrow \mathbf{W}^n - \eta \frac{\partial \ell}{\partial \mathbf{W}^n}, \qquad \mathbf{b}^n \leftarrow \mathbf{b}^n - \eta \frac{\partial \ell}{\partial \mathbf{b}^n} \tag{1}$$

Unlike for logistic regression, $\ell(\theta;D)$ is a non-convex function of its parameters with numerous local minima and saddle points (Pascanu et al., 2014), so at least we would like to find a point with sufficiently low value of NLL.

Normally, to estimate the direction of steepest descend an average gradient over subset (minibatch) of training examples is used instead of the whole training set. Minibatch size is determined empirically; it regulates the tradeoff between the variance of gradient estimates and computational time. "Noisy" gradient also helps to escape certain local minima (Bengio, 2012).

Determining a good learning rate η makes the optimization tricky: if η is too high, the system diverges in terms of the objective function; if η is too low, learning proceeds very slowly. To alleviate this task we used ADADELTA (Zeiler, 2012) – one of the several methods, which adapts the learning rate during training. At iteration t ADADELTA updates some parameter ω as follows:

$$\triangle\omega_t \leftarrow -\frac{RMS[\triangle\omega]_{t-1}}{RMS[g]_t} \cdot g_t,$$

$$\omega_t \leftarrow \omega_t + \triangle\omega_t.$$

In the above formula $RMS[g]_t = \sqrt{E[g^2]_t + \varepsilon}$, where $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2$ is an exponentially decaying average of squared gradients g^2 with a decay rate ρ . Similarly, $RMS[\triangle\omega]_{t-1} = \sqrt{E[\triangle\omega^2]_{t-1} + \varepsilon}$ – is a decaying average of previous updates. ε is a small constant to ensure the conditioning of the denominator and a start from the first iteration, where $E[\triangle\omega^2]_0 = 0$. ADADELTA assigns each dimension in the space of parameters its own dynamic learning rate with some desired properties. Its numerator $RMS[\triangle\omega]_{t-1}$ implies an acceleration in directions, where gradient signs are the same, and it slows down the progress along the dimensions, where gradients change their sign. Such behavior is beneficial when optimizing long and narrow valleys of the error surface in parameter space. The denominator $RMS[g]_t$ aims to give large learning rates to small gradients and vice versa, thus balancing the progress along different dimensions.

Backpropagation algorithm (Bengio et al., 2014) is used to calculate gradients in Eq. 1. It applies a chain rule to iteratively compute gradient for each layer.

Let us remind the notation: $\mathbf{x}^n = f(\mathbf{W}^n \cdot \mathbf{x}^{n-1} + \mathbf{b}^n)$ – is an ouput of the nth layer ($n = 1 \ldots N-1$), where a single neuron i has an activation: $\mathbf{x}_i^n = f(\sum_{\mathrm{j}} \mathbf{W}_{i,j}^n \cdot \mathbf{x}_j^{n-1} + \mathbf{b}_i^n)$. We will denote pre-activation of the nth layer as $\mathbf{a}^n = \mathbf{W}^n \cdot \mathbf{x}^{n-1} + \mathbf{b}^n$. The network's output vector, which gives probability distribution over classes is $\mathbf{x}^N = softmax(\mathbf{W}^N \cdot \mathbf{x}^{N-1} + \mathbf{b}^N)$. The cost function on one example we will denote as $C = -\log \mathbf{p}_y$, where $\mathbf{p}_y$ stands for the probability assigned to the target class y , which we want to maximize.

Now we can apply a recursive procedure to compute gradients with respect to weights and biases on each layer n from N down to 1:

1. biases:
$$\frac{\partial C}{\partial \mathbf{b}_i^n} = \frac{\partial C}{\partial \mathbf{a}_i^n} \cdot \frac{\partial \mathbf{a}_i^n}{\partial \mathbf{b}_i^n} = \frac{\partial C}{\partial \mathbf{a}_i^n} \tag{2}$$

2. weights:
$$\frac{\partial C}{\partial \mathbf{W}_{i,j}^n} = \frac{\partial C}{\partial \mathbf{a}_i^n} \cdot \frac{\partial \mathbf{a}_i^n}{\partial \mathbf{W}_{i,j}} = \frac{\partial C}{\partial \mathbf{a}_i^n} \cdot \mathbf{x}_j^{n-1} \tag{3}$$

3. back-propagate gradients to layer $n-1$:

$$\frac{\partial C}{\partial \mathbf{x}_j^{n-1}} = \sum_i \frac{\partial C}{\partial \mathbf{a}_i^n} \cdot \frac{\partial \mathbf{a}_i^n}{\partial \mathbf{x}_j^{n-1}} = \sum_i \frac{\partial C}{\partial \mathbf{a}_i^n} \cdot \mathbf{W}_{i,j}^n$$

$$\frac{\partial C}{\partial \mathbf{a}_j^{n-1}} = \frac{\partial C}{\partial \mathbf{x}_j^{n-1}} \cdot \frac{\partial \mathbf{x}_j^{n-1}}{\partial \mathbf{a}_j^{n-1}} = \frac{\partial C}{\partial \mathbf{x}_j^{n-1}} \cdot \frac{d}{d\mathbf{a}_j^{n-1}} f(\mathbf{a}_j^{n-1}) \tag{4}$$

and repeat from step 1.

To start the computation we still need $\frac{\partial C}{\partial \mathbf{a}_i^N}$ – gradient with respect to pre-softmax sum. A simple derivation yields: $\frac{\partial C}{\partial \mathbf{a}_i^N} = \mathbf{p}_i - I\{y = i\}$.

Activation functions commonly used in neural networks are: logistic function $(1+\exp(-x))^{-1}$, hyperbolic tangent $tanh(x)$ (Fig. 6) and linear rectification $\max(0,x)$ (Fig.7). The latter gives rise to rectified linear units (ReLU) (Nair and Hinton, 2010; Glorot et al., 2011).

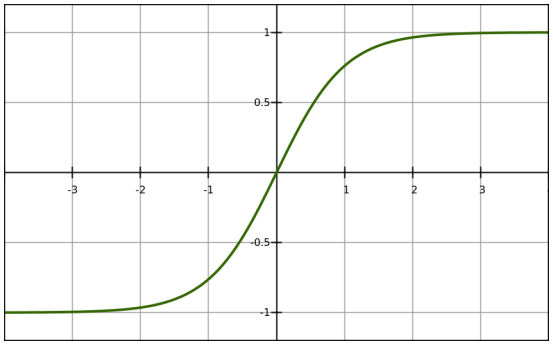

Figure 6: Hyperbolic tangent

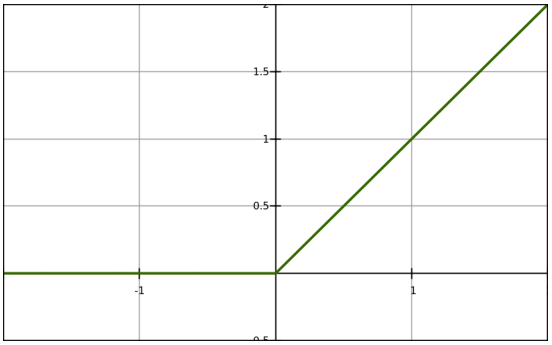

Figure 7: Linear rectification

To ensure the information easily propagates forward and backward through the network from the very start of training, weights should be properly initialized. For *tanh* units it is recommended to uniformly sample the weights for the i-th layer from $\pm\sqrt{\frac{6}{fan_{in}+fan_{out}}}$ interval, where fan_{in} is the number of units in the $(i-1)$-th layer and fan_{out} is the number of units in the layer i (Glorot and Bengio, 2010). For ReLUs a common practice is to initialize weights from $\mathcal{N}(0, 0.01)$. Initial biases for *tanh* can be zeros; for ReLUs positive values (we used 0.1) are preferred: units will mostly operate in their linear regime, which implies non-zero gradients.

As we could see from Eq. 4 in backpropagation algorithm, the derivative of the cost function with respect to pre-activations involves the derivative of the activation function. Fig.8 plots $\frac{d}{dx}tanh(x)$, which indicates that units may saturate when pre-activations are large in absolute value (similarly, for logistic units). As the result gradients approach zero and the error does not propagate to the lower layers. That is why in deep neural networks ReLUs have advantages over other activations: when a unit is activated its derivative is one and zero otherwise. It is fine for

ReLUs to saturate on some examples, however we have to make sure by choosing good initial parameters, that units do not always produce zeros. To conclude, ReLUs allow to effectively train deep architectures because of efficient gradient propagation, outputs sparsity and cheap computation.

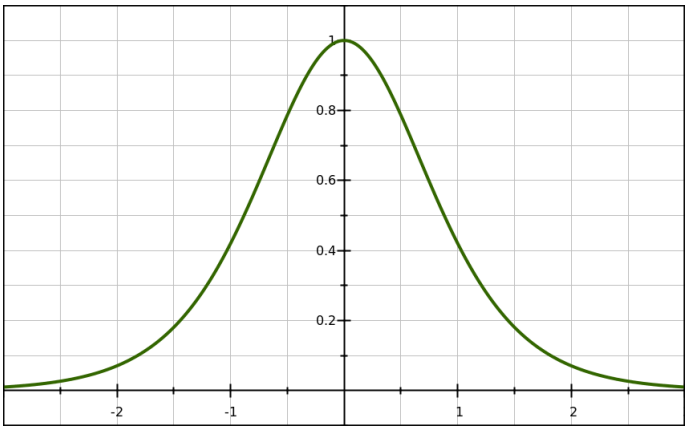

Figure 8: Derivative of the hyperbolic tangent

3.2.2 Convolutional neural networks

Convolutional Neural Networks (convnets)(LeCun et al., 1998) is a special class of neural networks, which aims to emulate the behaviour of a visual cortex by exploiting locally-sensitive neurons, tiled together to cover the complete visual field. Convnet normally consists of three types of layers: convolutional, sub-sampling and dense (fully-connected) layers.

Convolutional layers are built upon the ideas of sparse connectivity and weights sharing. Sparse (local) connetivity enforces units to connect only with a subset of units from the previous layer, called a *receptive field*, as in the primary visual cortex. Such units are then replicated across the entire input to form a *feature map* – a set of units with identical weights and biases, which act like filters for detecting the same feature in different locations, while traversing the input. Fig. 9 illustrates a feature map $\mathbf{h}$, formed by convolving an input with a linear filter $\mathbf{W}$ (3 parameters), adding a bias b (1 parameter) and applying some non-linearity f:

$$\mathbf{h}_i = f((\mathbf{W} * \mathbf{x})_i + b),$$

where $i = 1..4$ is an index of a unit within a feature map and $*$ denotes a convolution operation. Connections with equal weights are indicated with the same type of lines. Because of weights sharing convnets have fewer parameters than traditional fully-connected networks, what makes them particularly useful for large scale problems. We should note that in general, convolution can be done in 2 or 3 dimensions, e.g. for images and video, however in our models we need one-dimensional convolution, so further we will only show 1D examples.

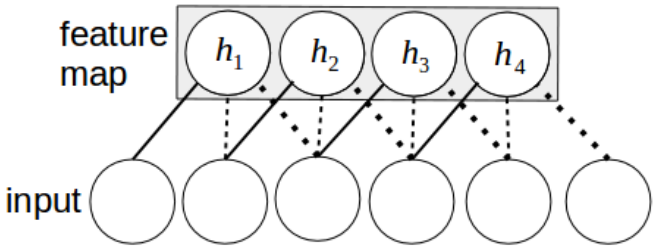

Figure 9: A schematic of a 1D convolution with one feature map (bias is ommited)

Strided convolution is oftenly used to reduce the resolution of the output feature maps. In this type of convolution, the distance between receptive fields centers of neighboring units is greater than 1. An example of 1D convolution with a stride of 2 is shown on Fig. 10.

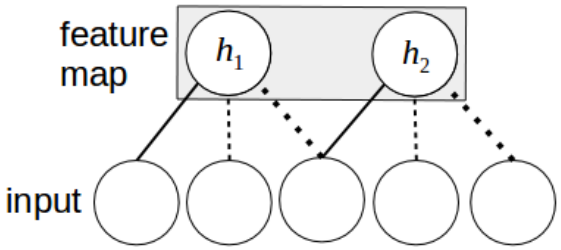

Figure 10: Convolution with a stride of 2

To form a rich representation of the data in each layer multiple feature maps are used. By stacking convolutional layers, units in higher layers can see an increasingly larger part of an input. Fig. 11 illustarates the way two convolutional layers can be connected. Here we have three feature maps $\mathbf{h}^{n-1,i}$, $(i = 1 \dots 3)$ in layer $n-1$ and two feature maps $\mathbf{h}^{n,1}$, $\mathbf{h}^{n,2}$ from layer n with weights $\mathbf{W}^{n,1}$, $\mathbf{W}^{n,2}$ and biases $b^{n,1}$, $b^{n,2}$ respectively. Each kth unit $(k = 1 \dots 3)$ from $\mathbf{h}^{n,1}$ and $\mathbf{h}^{n,2}$ have a receptive width of 2. We can write their activations as follows:

$$\mathbf{h}_k^{n,1} = f(\sum_{i=1}^{3} \sum_{j=k}^{k+1} \mathbf{W}_{i,j}^{n,1} h_j^{n-1,i} + b^{n,1}),\ \mathbf{h}_k^{n,2} = f(\sum_{i=1}^{3} \sum_{j=k}^{k+1} \mathbf{W}_{i,j}^{n,2} h_j^{n-1,i} + b^{n,2})$$

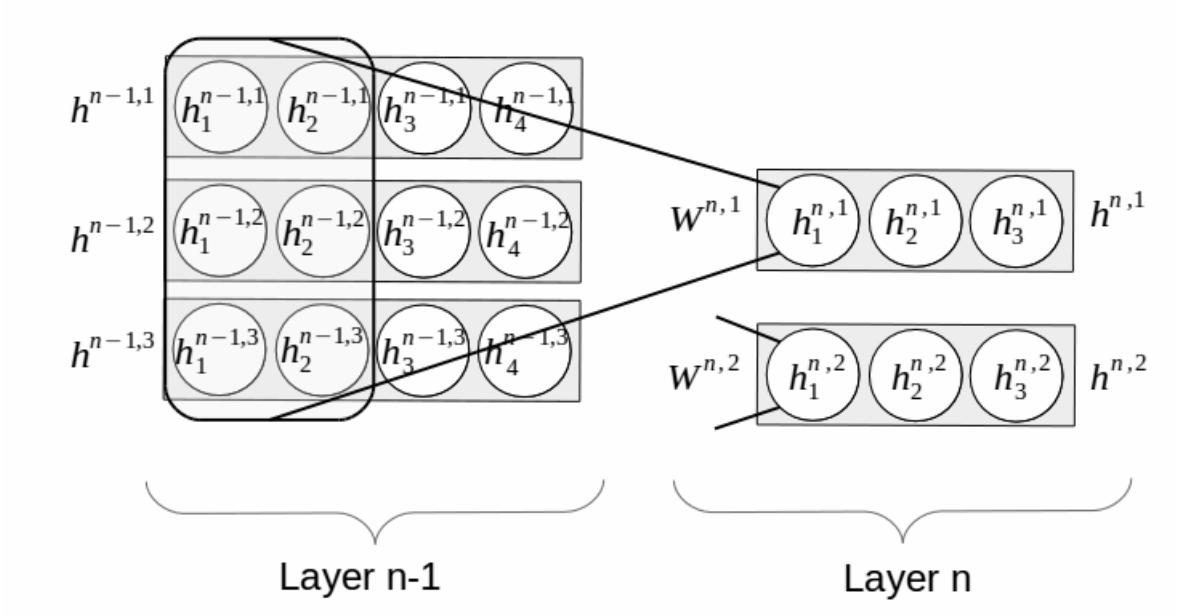

Figure 11: Connection between convolutional layers

Sub-sampling (pooling) layers usually compute the maximum or the average of non-overlapping sub-regions of the input and are included between convolutional layers. They reduce the spatial resolution of feature maps, thus removing potentially harmful information about the exact position of the detected features. Pooling layers provide additional translation invariance and reduce computation for higher layers. A schematic of a max-pooling operation is shown on Fig. 12.

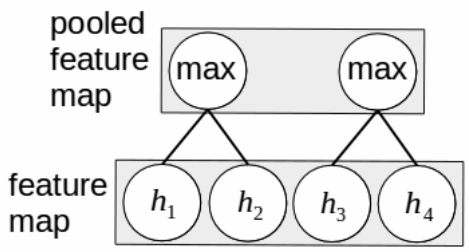

Figure 12: A schematic of a max-pooling operation

Dense layers are normally the topmost layers in convnets. Their units are connected with all units from the previous (convolutional, pooling or dense) layer, so they incorporate the largest part of convnet's parameters. These layers contain no spatial information anymore, so convolutional layers cannot follow dense layers.

Training of convnets can be done with some changes to the backpropagation algorithm due to the shared weights and overlapping receptive fields in convolutional layers. Returning to the example on Fig. 11, we see that a particular weight $\mathbf{W}_{i,j}^{n,1}$ and bias $\mathrm{b}^{n,1}$ is present in activations

of all the units from $\mathbf{h}^{n,1}$, so using $\mathbf{a}_k^{n,1}$ as a pre-activation of kth unit in $\mathbf{h}^{n,1}$ we get the following gradients:

$$\frac{\partial C}{\partial \mathrm{b}^{\mathrm{n},1}} = \sum_{k=1}^{3} \frac{\partial C}{\partial \mathbf{a}_k^{n,1}} \cdot \frac{\partial \mathbf{a}_k^{n,1}}{\partial b^{n,1}} = \sum_{k=1}^{3} \frac{\partial C}{\partial \mathbf{a}_k^{n,1}}$$

$$\frac{\partial C}{\partial \mathbf{W}_{i,j}^{n,1}} = \sum_{k=1}^{3} \frac{\partial C}{\partial \mathbf{a}_k^{n,1}} \cdot \frac{\partial \mathbf{a}_k^{n,1}}{\partial \mathbf{W}_{i,j}^{n,1}} = \sum_{k=1}^{3} \frac{\partial C}{\partial \mathbf{a}_k^{n,1}} \cdot \mathbf{h}_j^{n-1,i}$$

To propagate the error to lower layers, we need gradients $\frac{\partial C}{\partial \mathbf{h}_k^{n-1,i}}$, $(k = 1 \dots 4, i = 1 \dots 3)$. One can notice that $\mathbf{h}_2^{n-1,i}$, for instance, is present in activations of 4 units from nth layer: $\mathbf{h}_1^{n,1}$, $\mathbf{h}_2^{n,1}$, $\mathbf{h}_1^{n,2}$, $\mathbf{h}_2^{n,3}$. Considering this fact, we can write the gradients as:

$$\frac{\partial C}{\partial \mathbf{h}_k^{n-1,i}} = \sum_{j=1}^{2} \sum_{i=k}^{k-1} \frac{\partial C}{\partial \mathbf{a}_i^{n,j}} \cdot \frac{\partial \mathbf{a}_i^{n,j}}{\partial \mathbf{h}_k^{n-1,i}} = \sum_{j=1}^{2} \sum_{i=k}^{k-1} \frac{\partial C}{\partial \mathbf{a}_i^{n,j}} \cdot \mathbf{W}_{i,k}^{n,j}$$

The rest of the backpropagation procedure is straightforward.

3.3 Overfitting

A model is said to overfit when it learns a sampling noise in a particular training set instead of a true underlying relationship between inputs and outputs, thus failing to generalize well to new examples. Obviously, we would like our model to have enough capacity to represent the true structure in the data and to be insensitive to minor fluctuations in a training data sample. However, there is always a tradeoff between these 'bias' and 'variance' criteria. In our case, when training set ranges from 60 to 1512 examples and model may have thousands of parameters, we need regularization methods to prevent models from having high variance.

The most common technique is to penalize the parameters with large magnitude and thus bound the ability of the model to fit any given function. It is done by adding L_1or/and L_2 norm of the weights to the loss function with some λ coefficient – a hyperparameter, which controls the amount of the penalty.

Additionally, for any model one can use the following methods to improve generalization:

1. *data augmentation* increases the train set size by using transformations, which preserve labels.

2. *model averaging* is a good way to reduce overfitting, and it is most effective when models disagree in their predictions. This implies individual models to be better on different test cases. So if we average individual predictors with roughly the same performance, typically it gives better predictions. In classification problems model averaging can be done either by taking geometric mean of the predicted probabilities followed by normalization (product of models) or by taking arithmetic mean of their output probabilities (mixture of models), (Hinton and Tieleman, 2012).

For neural networks there are several specific techniques:

1. *early stopping* (Bengio, 2012) keeps track of the generalization error on a validation set as the learning proceeds, and stops training as validation performance gets worse. As a result, it provides a value for another hyperparameter – a number of training iterations. Early stopping compensates large capacity by smaller training time.

2. *dropout* (Srivastava et al., 2014) is by far the most efficient technique for neural networks to reduce overfitting. Dropout approximately combines 2^n possible neural network architectures by training a network with n units, thus dropout can be considered as a form of model averaging. The key idea of dropout is to drop units with probability p from the network each time a training case is presented, so a new "thinned" network is trained. Because the weights are shared between different thinned networks, every such model is strongly regularized. Feed-forward operation becomes: $\mathbf{x}_n = f(\mathbf{W}_n \cdot (\mathbf{r}_n \circ \mathbf{x}_{n-1}) + \mathbf{b}_n)$, where $\mathbf{r}_n$ is a vector of Bernoulli random variables, each variable is 0 with probability p; $\circ$ denotes element-wise multiplication. At the test time a single network without dropout is used, but its weights are scaled: $\mathbf{W}_n^{test} = (1-p) \cdot \mathbf{W}_n$, so the output for any hidden unit is the same as the expected output during training.

3.4 Model validation

To estimate how accurately the predictive model will perform on unseen data and/or to tune the hyperparameters, we can train the model on one part of available data and test on the other, hold-out part (rule of thumb: 20% of the data). A better estimate of a model's performance can be done via cross-validation (CV). In the basic CV approach with k-folds the training set is partitioned into k smaller sets, for each of the folds a model is trained using $k-1$ of the folds and the resulting model is validated on the remaining k-th part. Performance over k folds is then averaged (Bishop, 2006).

For epilepsy research reliable cross-validation procedure would be beneficial as long as the amount of EEG data is limited, and it may be a luxury to have a separate validation set. Our type of data requires a special procedure of splitting into train/validation sets because of the following reasons:

1. Data clips within one hour sequence are more similar to each other than to clips from other sequences. Completely random split will place the clips from one sequence into train and validation sets, therefore, validation results will be too optimistic. The solution is to keep sequences intact: each 1 hour of data goes either to train or to the validation set.

2. Number of preictal and interictal instances is highly imbalanced (see Table 1), so with our earlier CV split in an extreme case the model may be trained or validated on interictal data only. To prevent this, one can use stratified k-fold CV, which preserves the percentage of samples for each class.

Following this reasoning we used the following CV procedure[4]: we pick the number of folds equal to the number of preictal sequences, then each validation part will contain one preictal sequence and $\frac{n_{interictal}}{n_{preictal}}$ interictal sequences (some parts will be smaller if the division has a remainder) . On each fold we save our validation predictions and in the end we compute AUC across all subjects (note that we do not calibrate the predictions). We expected these cross-validation scores to be used for model selection and for estimating the leaderboard AUC. The efficiency of our CV procedure will be discussed in section 3.4.

3.5 Calibrating the predictions

To compensate for the problematic evaluation metric as discussed in section 2.3, calibration of the prediction based on the test data was allowed. Therefore we used unity-based normalization (hereafter "min-max"): per subject predictions were restricted to have the same range from 0 to 1 with the following formula: $p_{scaled} = \frac{p-\min(\mathbf{p})}{\max(\mathbf{p})-\min(\mathbf{p})}$, where $\mathbf{p}$ is a vector of subject's predictions.

Several methods were proposed by other participants:

1. For each subject standardize test predictions and post scale them through a logistic function to (0,1) interval (we will refer to as "softmax scaling").

2. For each subject take the median of predictions, subtract it from the scores, and then divide the scores by 2 and add 0.5 (hereafter is called "median scaling")

These transformations have no effect on per subject AUC, so in practical terms it does not change the performance of individual models. However, it is critical for the combined AUC; this fact may obfuscate the results: bad models with good post-scaling may have good scores and vice versa.

3.6 Fourier transform

Let $a = (a_0, \ldots, a_{n-1})$ be a vector of samples from the signal, which are equally spaced in time. Its Discrete Fourier Transform (DFT) converts the signal in a time domain to the frequency domain by representing it as a combination of complex sinusoids with frequencies $f_k = \frac{k}{n}$ and corresponding coefficients $A_k = \sum_{m=0}^{n-1} a_m \exp(-2\pi i \frac{mk}{n})$, $k = 0, \ldots, n-1$. $|A_k|$ is the amplitude of a sinusoid with frequency f_k.

The following definitions will be used in the further explanations:

1. set of $|A_k|$ form an amplitude spectrum which consists of $n/2+1$ components for the real-valued signal.

[4]The similar approach was mentioned several times on the competition forum: http://www.kaggle.com/c/seizure-prediction/forums. It also resembles an idea used in (Park,2011).

2. set of $|A_k|^2$ form a power spectrumspectral energy: $E_k = |A_k|^2$
3. energy of a signal in time domain: $E = \sum_{m=0}^{n-1} |a_m|^2$
4. signal power in time domain: $P = \frac{1}{n} \sum_{m=0}^{n-1} |a_m|^2$

3.7 Convolutional neural networks approach

3.7.1 Data preprocessing

The first preprocessing steps were to resample the signal to 400 Hz and apply a band-pass filter between 0.1-180 Hz. If we denote the number of channels as N, then each 10 minute clip will have dimension of $N \times (600\,\text{sec} \cdot 400\frac{1}{\text{sec}}) = N \times 240000$.

Previous studies showed that features from the frequency domain are effective for seizure prediction (Howbert et al., 2014), so we followed the similar ideas. Each 10 minute time series was partitioned into nonoverlapping 1 min frames, and within each frame we calculated $\log_{10}$ of its amplitude spectrum. The result of this procedure is shown of Fig.13 (1st channel of an interictal data clip from Dog_1).

The spectrum was divided into 6 frequency bands: delta (0.1-4 Hz), theta (4-8 Hz), alpha (8-12 Hz), beta (12-30 Hz), lowgamma (30-70 Hz) and highgamma (70-180 Hz). In each frequency band we took an average of the corresponding $\log|A_k|$, which yields Fig.14. Thus, the dimension of the data clip is equal to $N \times 6 \times 10$ (channels $\times$ frequency bands $\times$time frames). We should note that the size of original data from 7 subjects is 106 Gb and after preprocessing less than 200 Mb is needed, so it may be advantageous for devices with limited memory capacity.

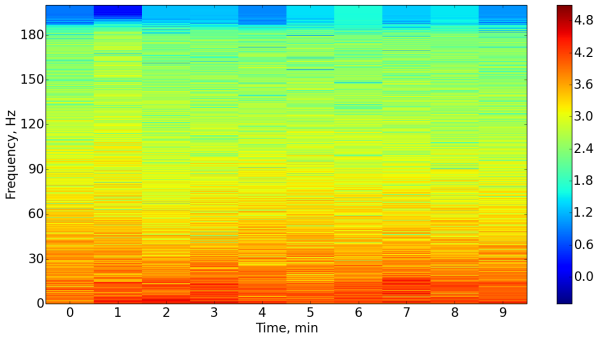

Figure 13: Log of the amplitude spectrogram

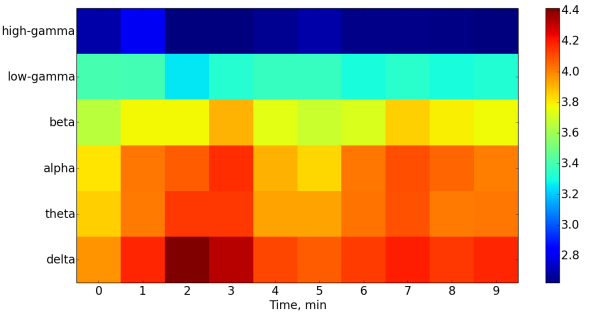

Figure 14: Binned spectrogram

Spectrograms from each channel, which formed an input to the neural network, were standardized with one of the following schemes:

1. flatten the spectrogram into a vector of 60 values, and from each value substract the mean and divide by standard deviation calculated over the complete train dataset. By doing so, we account for the position of a feature both in time and frequency.

2. decompose each 6×10 spectrogram into 10 examples each of 6 features, standardize these 6-dimensional vectors on per feature basis and combine them back into 6×10 matrix. In contrast to the previous scheme, we do not account for the time location.

Additionally, in some of our models we used standard deviation of a signal, calculated within the same time windows as DFT, resulting data clips with shape $N \times 7 \times 10$. Several models were also trained on frequency data partitioned into 8 bands by splitting wide gamma bands into 2 parts: low-gamma to 30-50Hz and 50-70Hz, high-gamma to 70-100Hz and 100-180Hz.

As a form of data augmentation, we took overlaps between consecutive 10 minute clips within one hour sequence. For instance, with 5 minutes overlap a new clip is formed as shown of Fig. 15. This can be done on the fly during model training, so we do not need to store additional data.

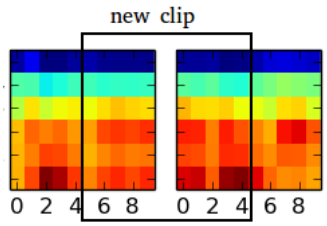

Figure 15: Overlap of 5 minutes between consecutive clips

3.7.2 Architectures

In the way data was collected and due to nonstationarity of EEG signal, signs of preictal activity may not be present during the whole 10 minute clip: they may appear at the beginning of the clip as well as in the end and for some clips from the preictal period epileptic symptoms may not be present at all. Therefore, we want our model to learn features localized in time and then combine information from different time slices. This idea was implemented with a convolutional neural network, which performs one-dimensional convolution through time, thus extracting the same types of features from each time frame separately, and then combines the information across the time axis in higher layers.

Moreover, it is important to consider the relationships between pairs of EEG channels (Mirowski et al., 2009). One option is to extend a feature set with pairwise features e.g. cross-correlations. However, our idea was to let the filters in the first convolutional layer to see all frequency bands of all the channels at a particular time frame, so they could learn relevant relationships by itself.

In order to check the aforementioned prior beliefs, we explored several architectures and further we will describe two of them in details.

An example of the first type of model's architecture is shown on Fig.16. Its first layer (C1) performs convolution in a time dimension over all N channels and all 6 frequency bands, so the shape of its filters is $(6 \cdot N) \times 1$. C1 has 16 feature maps each of shape 1×10; number

of parameters is $16 \cdot 6 \cdot N + 16$ (additional 16 are biases). The second layer (C2) performs convolution with filters 16×2, so the width of the resulting feature map is $10 - 2 + 1 = 9$. Second layer has 32 feature maps and $16 \times 2 \times 32 + 32 = 1056$ parameters. C2 is fully connected with F3, which contains 128 units, so there are $32 \times 9 \times 128 + 128 = 36992$ parameters. Last layer is logistic regression with 129 weights (128 inputs from F3 plus 1 trainable bias). In total, this network has ≈40K weights (exact number differs across subjects as the number of channels varies)

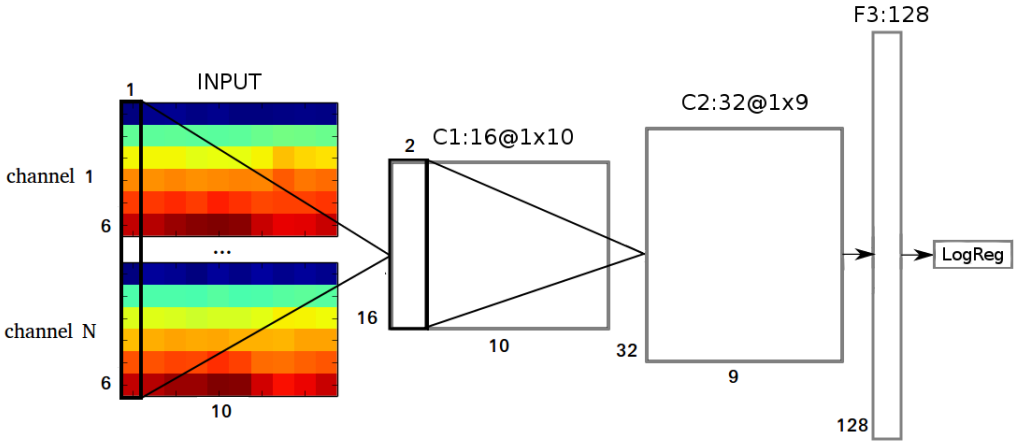

Figure 16: First type of architecture (see main text for explanation)

If we include *global temporal pooling* layer (Dieleman, 2014) to the previous model it will look like on Fig. 17 (our second type of models). Pooling layer GP3 computes the following statistics: mean, maximum, minimum, variance, geometrical mean, L_2 norm over 9 values in each feature map from C2. Therefore, GP3 contains $6 \times 32 = 192$ units, which are connected with 128 units in F4 layer, so there are 24704 parameters between GP3 and F4. Due to global pooling layer, which has no trainable weights, this network has $\approx 12K$ less parameters than the previous model.

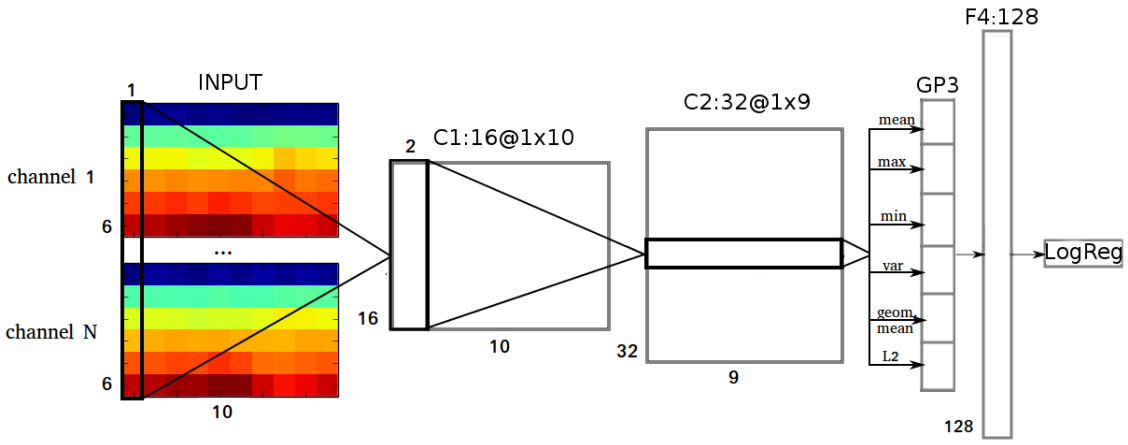

Figure 17: Second type of architecture(see main text for explanation)

We would like to note that in early experiments we performed convolution in the first 2 layers on each EEG channel separately, thus combining information from different channels only in hidden layers. With this approach we could not get higer than ≈ 0.6 on a public test set, which also justifies a good choice of the abovementioned architectures.

3.7.3 Model training

All models were trained with dropout in fully connected and logistic regression layers. Rectified linear units were used in all layers for type 1 models, however for models with global pooling we noticed *tanh* nonlinearity in the dense layer to work better. We used first standardization scheme, which accounts for the position in time and frequency, for type 1 models and the second scheme for type 2 models. We noticed that inserting max-pooling layers between convolutional layers harms the performance and convolution with a stride of 2 in C2 layer works well for type 2 models. All networks were trained on mini-batches of 10 examples per batch with ADADELTA method, where decay rate ρ was set to 0.95 and $\varepsilon = 10^{-6}$. However, we had to modify the original ADADELTA algorithm due to extremely unstable behavior, for which we do not know the reason (also it was not reported before). The updates in ADADELTA can be rewritten as $\triangle\omega_t \leftarrow -\eta \cdot \frac{RMS[\triangle\omega]_{t-1}}{RMS[g]_t} \cdot g_t$ with a default learning rate $\eta = 1.0$. For one of the convnet models, learning curve, which plots negative log-likelihood on the train set of Dog_1 after each 10 gradient updates is shown on Fig.18 in grey. Clearly, this method does not converge, so after sufficient number of iterations we may eventually end in a point with very high cost. Therefore, we tried to decrease η to 0.01, the result given by a black curve on Fig. 18 is satisfactory.

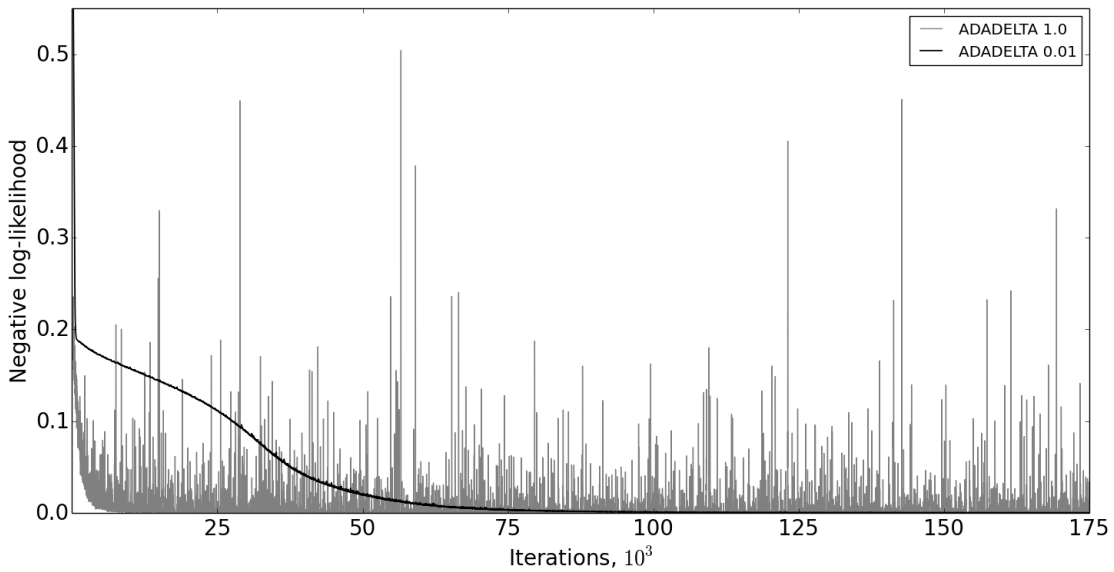

Figure 18: Training cost over time, when using ADADELTA with different learning rates

For convnets CV is computationally expensive, so initially we tried doing validation (1-

fold CV) with preserving the sequences as described in section 3.4. Early stopping was also used in this case. However, splits for some subjects produced abnormal results, like AUC of ≈ 0.2, which obviously was not a representative estimate for the score. Therefore, for convents we abandoned to perform validation and trained the networks for a fixed number of gradient updates equal to 175K iterations. Afterward, we were guided only with a public test score, which was potentially dangerous as one might overfit to the public test set, obtaining drastically worse results on the private set.

3.7.4 Implementation

We used Python and Theano (Bergstra et al., 2010; Bastien et al., 2012) to implement our models. Theano can automatically perform symbolic differentiation, which greatly simplifies the code. Moreover, the program can be run on CPU as well as on GPU without additional efforts. On CPU (Intel Core i7-3632QM) it takes about 2-6 hours, depending on the number of parameters, to train all per subject models.

3.7.5 Results

The result, which got the 10th place in the final ranking with **0.81292** public and **0.78513** private score, was obtained by taking geometric mean of min-max calibrated predictions from 11 models. All models had a global pooling layer, but different settings for number of feature maps, shape of filters, amount of dropout etc. (exact values of parameters along with individual model scores are given in Appendix A).

In Table 2 several results are listed. The table contains data description, architectures and public/private scores with and without min-max calibration. For instance, for the second model DFT and standard deviation of the time-domain signal was calculated in 1 minute windows; for each channel frequency features were taken within 6 bands; 9 minute overlaps between clips were used during training. Network has 2 convolutional layers with rectified linear units, global pooling layer and fully- connected layer with tanh activations, logistic regression unit is the network's output. C2 layer performs convolution with a stride of 2. Dropout of 0.3 and 0.6 is used in the last two layers.

The best calibrated model on a public test set appeared to be type 1 model – with full connections between convolutional C2 and hidden F3 layers, which implies that the model accounts for features location in time. As expected, this model cannot generalize well and its performance significantly dropped on the private set. We tried many type 1 models with various hyperparameters, and usually they scored better than models of type 2 (with global pooling layer). One should remember that during the competition we could not see the private score, so public results were misleading. However, relying on our intuition, we were mostly exploring models with global pooling layer. As we see, models having better public score does not necessarily perform better on the private set as in case of the 2nd and 3rd models. Choosing

a single model even with the highest public score is risky, so model averaging is vital to reduce this uncertainty. Model 4 was the lowest-scoring model in the final ensemble.

We used min-max normalization of per subject predictions, which gives relative improvement of $\approx 1\%$.

	Data	Architecture	Public/Private (min-max)	Public/Private (no calibration)
1	6 bands; 1 min windows	C1: 16@1x10, ReLU C2: 32@1x9, ReLU F3: 128, ReLU, dropout 0.2 LogReg: dropout 0.5 (40K parameters)	**0.81448/** 0.76256	0.79899/ 0.75201
2	6 bands + std; 1 min windows; 9 min overlap between clips	C1: 32@1x10, ReLU C2: 64@1x5, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6 (200K parameters)	0.80872/ 0.77182	**0.80276/** 0.76432
3	8 bands + std; 2 min windows; 4 min overlap between clips	C1: 16@1x5, ReLU C2: 32@1x5, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6 (85K parameters)	0.80192/ **0.78114**	0.79414/ **0.77249**
4	8 bands + std; 30 sec windows; 5 min overlap	C1: 16@1x20, ReLU C2: 32@1x10, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6 (101K parameters)	0.78612/ 0.76133	0.77949/ 0.75465

Table 2: Performance of convnets models

3.8 Linear models approach

This approach, although without giving enough details for reproducing the results, was described after the competition deadline by Jonathan Tapson [5], who initially finished in the 2nd place. The idea is:

1. for each subject use penalized least squares to fit a linear model based on 1 minute chunks from the original 10 min clips. Dependent variable is a class label 0 or 1.

2. take the mean of 1 minute predictions to form a prediction for 10 minuite clips

[5] An outline of his solution was assembled from several posts on a competition forum: http://www.kaggle.com/c/seizure-prediction/forums/t/10945/congratulations-to-the-winners

3. rescale using the softmax normalization scheme.

This approach is sensible, however it does not have a theoretical justification, so we are going to use logistic regression and LDA instead. Further we will discuss this method in details.

3.8.1 Data preprocessing

The preprocessing steps on converting time series into spectrograms are analogous to section 3.7.1 with the following differences:

1. windows, in which DFT was calculated, had overlaps of 30 seconds. For instance, if the first window spans 0-60 seconds of the original EEG signal, the second window covers 30-90 seconds, etc. Therefore, from 10 minute clip we get 19 time frames.

2. frequency bands were: 1Hz bins in 1-50 Hz interval, 5Hz bands in 50-100Hz and 10 Hz bands in 100-180Hz, so there are 67 frequency features per time frame.

3.8.2 Model fitting and validation

Unlike for convnets, we will split the spectrogram into 1 min time frames, so from one original clip we get 19 examples with $(N \times 67)$ features each (N is a number of channels: 15, 16 or 24). After standardizing the features, we fit penalized logistic regression or LDA and get a prediction for each 1 minute clip in the test set. By taking their average we get a single prediction for an original clip: $p = \sum_{i=0}^{18} p_{1min}^{(i)}$, which is already a valid probability. In addition, we can calibrate the probabilities with e.g. softmax normalization: standardize the predictions within each subject by using mean and standard deviation of the test set predictions and pass them through the logistic function $P(Y = 1|x)_{scaled} = (1 + exp(\frac{p-\mu}{\sigma})))^{-1}$.

Before providing cross-validation results, we would like to verify the claim we made in section 3.4 about the similarity of clips within one sequence, which was the main obstacle for using a common CV procedure. Fig. 19 illustrates t-SNE mapping(van der Maaten and Hinton, 2008) of preictal EEG clips from Patient_2 (it was chosen due to a small amount of data for clear visualization). For this plot we used 6 frequency bands data from train and test sets, where the dimensionality of each clip was reduced to 50 by the means of principal component analysis; all data clips were then mapped with t-SNE into 2D space. The plot shows preictal clips from the train set with explicitly outlined groups, which correspond to particular sequences.

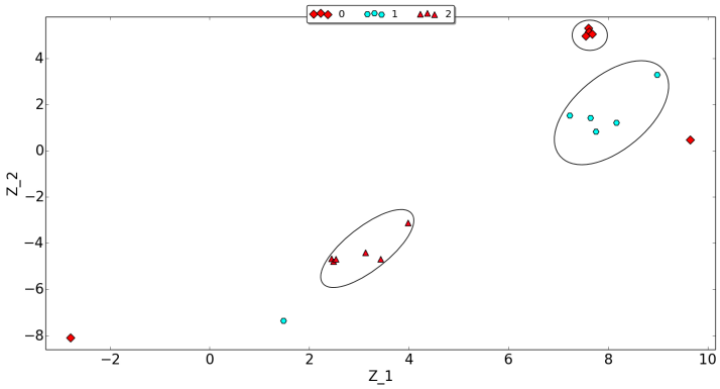

Figure 19: Visualization of preictal clips from Patient_2 by t-SNE

We performed cross-validation to compare logistic regression with several values of L_2 penalty (λ parameter) and LDA, fitted to the data with 6 frequency bands; the results are given in Table 3. Within the logistic regression framework results are consistent: higher CV score leads to higher public score, though the difference between CV scores is of 10^{-3} order so we would not rely on its conclusions. Evidently, cross-validation cannot be used to select models among various types: logistic regression with $\lambda = 10$ appears to be the best on CV, however on a public test set LDA outperforms it with a large margin. This phenomenon may arise due to disagreement between objective functions we are trying to optimize in logistic regression or LDA and AUC. The problem of CV is not trivial, so deeper analysis is required. For comparison purposes we provided the scores for 10-fold CV (denoted as Random CV in the table), when using a random stratified split.

Score \ Model	LR $\lambda = 0.0$	LR $\lambda = 0.01$	LR $\lambda = 0.1$	LR $\lambda = 1.0$	LR $\lambda = 10$	LR $\lambda = 100$	LDA
CV	0.85298	0.85391	0.85433	0.85503	**0.85767**	0.85296	0.83749
Public score	0.68562	0.69026	0.69712	0.70881	0.70820	0.67735	**0.74287**
Random CV	**0.93356**	0.93278	0.93028	0.92372	0.91430	0.89769	0.91130

Table 3: Results of cross-validation and public test scores

Despite our best efforts, CV score is still very optimistic estimate of the leaderboard score. Figure 20 provides the evidence of why this happens. In its highest extreme, train and test data sets are totally dissimilar, which we noticed from a t-SNE plot for Dog_3 when using the same 6-frequency bands data. For other subjects this effect is less expressed, but yet present.

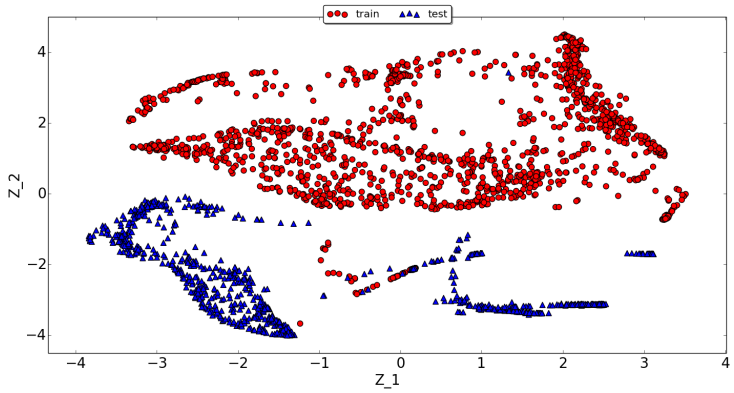

Figure 20: Visualization of train and test data clips from Dog_3 by t-SNE

3.8.3 Model analysis

According to the competition rules it was not allowed to manually pick features and models for each subject, e.g. for Dog_1 use only high-gamma features or for Patient_1 select fewer EEG channels. Though, it can be automatized using cross-validation, which is computationally expensive and as we could see earlier, it may not be reliable. Therefore, we needed a universal set of features for all the subjects; to gain some intuition on which features are useful for the task at hand, we will explore LDA model.

In our experiments we started from analyzing models based on 6-frequency-bands data. For each subject we plotted the absolute values of $\boldsymbol{\delta}$ coefficients. Figure 21 shows $|\boldsymbol{\delta}|$ from LDA model for Dog_1: each of 16 subplots corresponds to one EEG channel, and 6 coefficients (from 0 to 5) are related to delta, theta, alpha, beta, lowgamma and highgamma bands. We can notice that for several channels gamma bands have the largest coefficients and thus discriminating power. The similar behavior can be observed for other subjects as well. Therefore, useful information might be present in higher frequencies, and splitting relatively wide bands of 30-70 Hz and 70-180 Hz can reveal more relevant features.

For that reason we tried to split low- and high-gamma bands into 2 parts – 30-50Hz and 50-70Hz, 70-100Hz and 100-180Hz, which results 8 bands in total; and 4 parts with cuts at 30, 40, 50, 60, 70, 85, 100, 140 and 180 Hz, which yields 12 frequency features per channel.

Results of LDA models based on data with different frequency splits are given in Table 4. We can see that splitting gamma bands, as in case of 8 bands, improves both public and private scores, however using too narrow bands worsens the generalization. It is clear that public and private scores are not well correlated, so when choosing a model one should not faithfully entrust the public score (as we have seen for convnets in Table 2). Softmax calibration appeared to have a great impact on linear models: the relative AUC change is $\approx 7\%$.

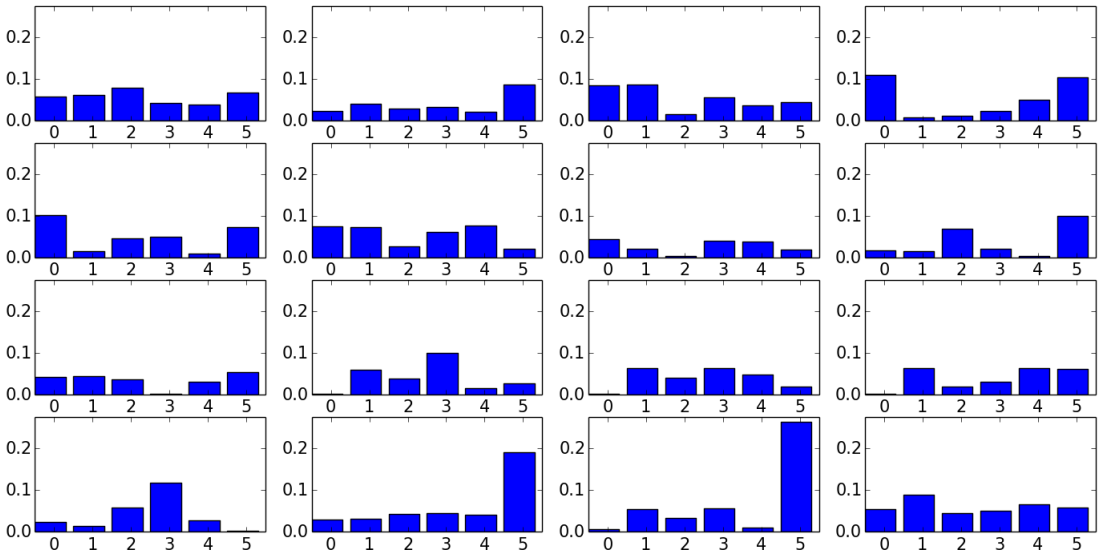

Figure 21: Magnitudes of the coefficients corresponding to 6 frequency bands and 16 EEG channels. Each subplot corresponds to 1 channel, each x-axis reflects frequency bands coded from 0 to 5

Score / Freq. bands	Public (softmax)	Private (softmax)	Public (no calibration)	Private (no calibration)
6 bands	0.79939	0.78787	0.74287	0.72723
8 bands	0.79894	0.79481	**0.75268**	**0.74224**
12 bands	0.80044	**0.79539**	0.74528	0.72823
67 bands	**0.81405**	0.79211	0.74693	0.69885

Table 4: Public and private performance of LDA models

3.9 Per subject analysis

Up to now we were looking at the AUC across all the subjects – the only accessible score while the competition was active. Since the test labels became available, in this section we will analyze per subject performance of the models and try to link it with a global AUC metric.

We will consider the final convnet ensemble, the worst on the public set single convnet (number 4 from Table 2) and LDA model fitted to 8 frequency bands data. Table 5 provides per subject results from these models based on the whole test set (public and private together). The difference is not significant: a weighted two-sided t-test (Bland and Kerry, 1998) between convnet ensemble and LDA, where scores were weighted accordingly to the number of test examples, yields a p-value of 0.7106016 and a 95% confidence interval $(-0.1114034, 0.1535390)$. Despite getting a low leaderboard score, a single convnet model performs comparably well on per subject basis, but the difference with LDA is still insignificant: p-value=0.573051 and 95% confidence interval is $(-0.08248713, 0.13559163)$.

We anticipated convnets to perform poorly for human subjects because of too few training examples relative to the amount of parameters. Only 9% of the test examples were from hu-

mans, so we focused mainly on dogs, however in the future it is reasonable to develop different architectures depending on the available data. We cannot explain why LDA is much better for Dog_2.

Model \ Subject	Dog_1	Dog_2	Dog_3	Dog_4	Dog_5	Patient_1	Patient_2
ensemble	**0.79489**	0.78937	0.80512	**0.90656**	**0.78864**	0.53734	0.58981
single convnet	0.76456	0.81243	**0.81547**	0.89970	0.73138	0.67304	0.56066
LDA 8 bands	0.74294	**0.86694**	0.74338	0.87601	0.29609	**0.73452**	**0.63971**

Table 5: Per subject AUC of convnets ensemble, single network and LDA

Figures 22 and 23 illustrate the probabilities, which single convnet model and LDA assigns to the test examples (true labels are denoted with different markers and we added some jitter to reduce overplotting). Without regarding the true labels, we see that convnets probabilities are densely compressed around the extremes, so this model is more confident in its predictions whenever it is right or wrong. It also implies that any classification threshold in a central part e.g. around 0.5 will produce approximately the same global AUC score. In contrast, LDA's probabilities are more spread, and their range differs across subjects as in case of Dog_1 vs. others, so the effect of changing the threshold is larger.

Greater divergence in per subject predictions seems to be a logical explanation on why LDA model benefits more from the calibration as we saw in Tables 4 and 2. We also noticed that min-max, softmax and median scaling schemes in combination with LDA and convnets have a different effect on the global AUC (see Appendix B). We cannot explain the reasons behind it, however it seems to matter only for the competition scores and have little use for the problem in general, so we did not delve into the details.

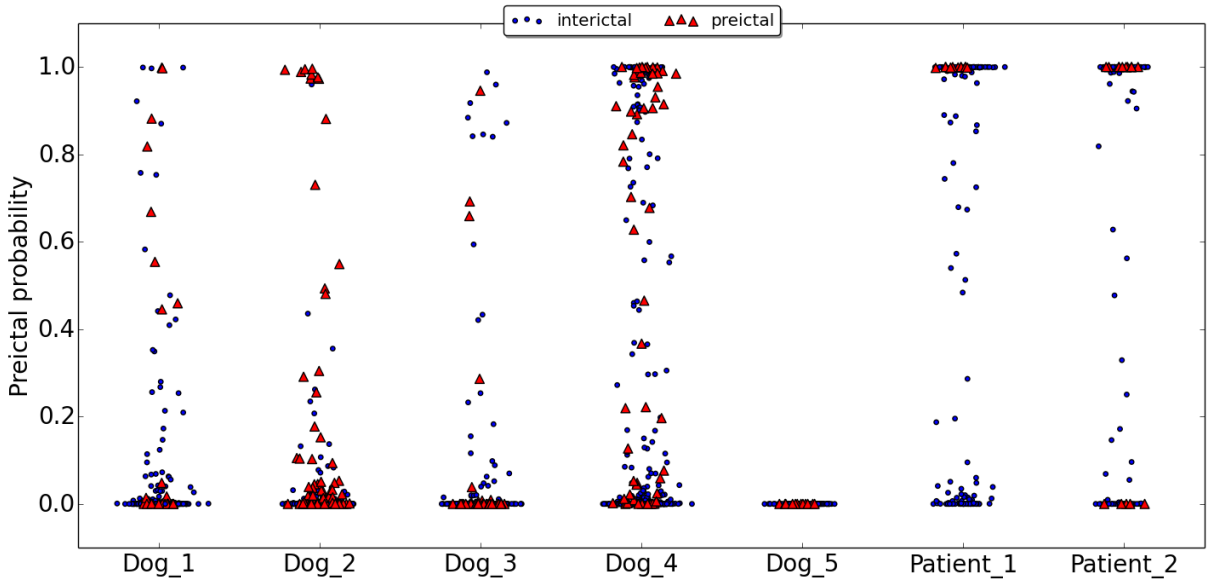

Figure 22: Per subject test probabilities from a single convnet.

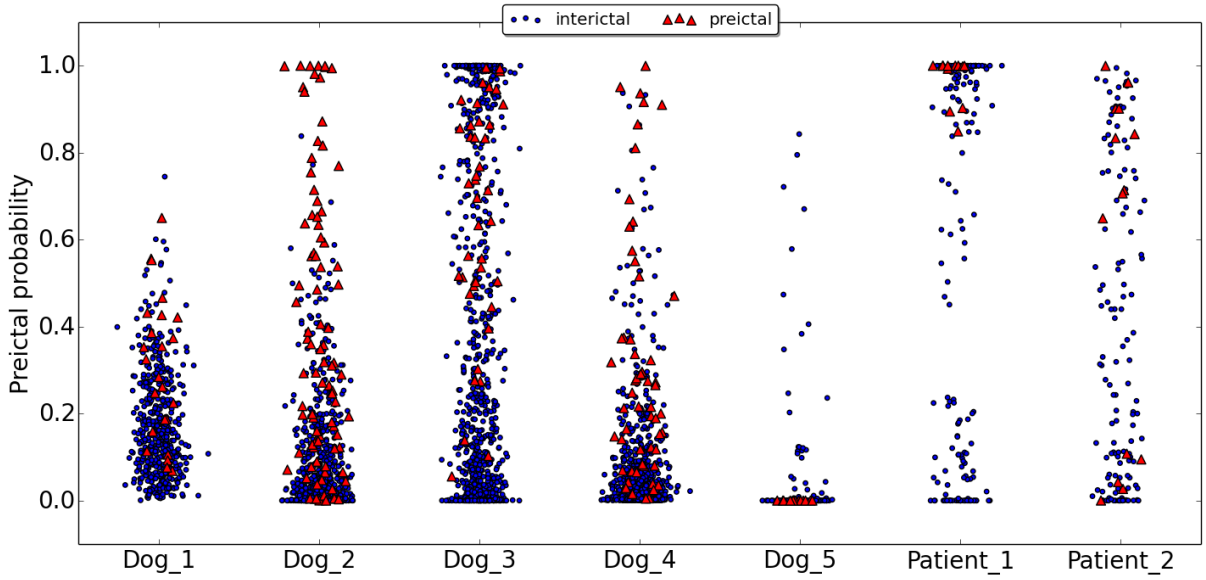

Figure 23: Per subject test probabilities from LDA model

In practical terms LDA is less robust to the choice of the threshold: in real test conditions a medical device may produce more false alarms and false negatives than expected for a threshold tuned on a patient's individual validation set. Unfortunately, having information only on the test labels for individual data clips we cannot check this hypothesis formally. It would be possible if we had per subject validation and test sets. This can be achieved in two ways: either we know how the public/private split was done for the competition or we have the timing of each clip in the test set, so we can split the data ourselves while keeping the test sequences intact.

Nevetheless, we can develop some intuition by analysing plots for sensitivity (true positive rate) and specificity (true negative rate) while modifying the classification threshold. Figures 24 and 25 plot sensitivity of convnet and LDA classifiers when the threshold ranges from 0.0 to 1.0 with a step of 0.02. We can see that LDA for canine subjects except Dog_4 performs better for most of the thresholds, but its sensitivity changes more rapidly than for convnets. Picking the best model may be alike with a problem of bias-variance tradeoff: in some cases, we can sacrifice the performance to be sure it is similar for new data. Definitely, we need to know the limits to which the sensitivity can be dropped without making a seizure prediction device useless. In practice, however, to predict a seizure we need to correctly classify at least one preictal segment, so the sensitivity calculated over all segments is not the best measure. True positives per seizure, as proposed in Buteneers (2012), may be more appropriate, but as we stated earlier we need the timing of each segment in the test set to quantify that.

Figures 26 and 27 illustrate similar plots for the specificity, where we can see greater plateau behavior of the convnet's curves as opposed to LDA. Indeed, high specificity is desirable in seizure prediction: a large number of false alarms can lead to "alarm fatigue", when patients start to disregard the warnings (Dolgin, 2014).

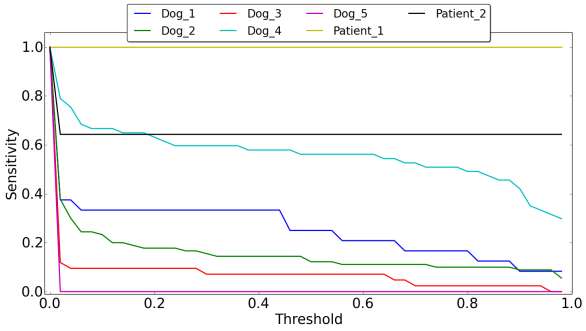

Figure 24: Convnet per subject sensitivity

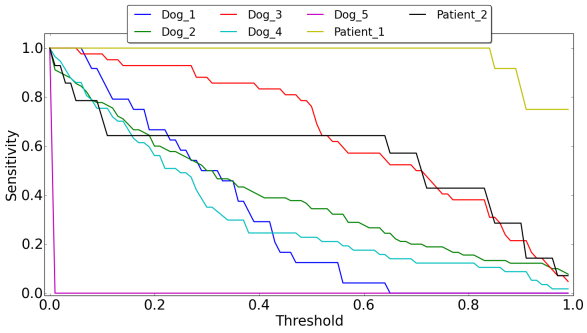

Figure 25: LDA per subject sensitivity

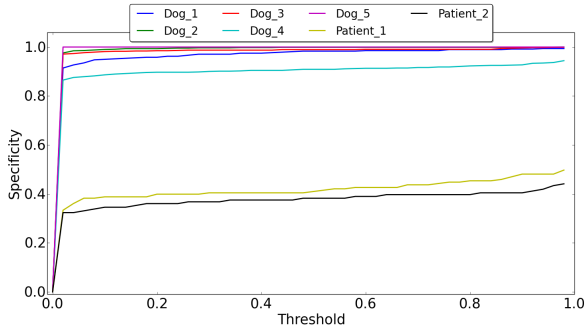

Figure 26: Convnet per subject specificity

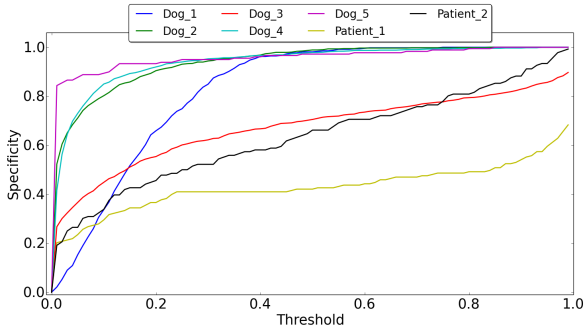

Figure 27: LDA per subject specificity

The downside of our convnet model is that we cannot find an optimal threshold on the train set as it is done in Buteneers (2012), because convnet's probabilities on train examples are approximately zeros or ones, and almost any threshold achieves a perfect classification. For LDA it is different, so we can optimize the threshold on the train set. A criterion, we used, was the minimum distance between the ROC curve and (0,1) point in the ROC space, which corresponds to zero false positive rate and maximal sensitivity. The results were compared to the convnet model with 0.5 threshold, Table 6 shows sensitivity and specificity on the whole test set for both models. Though, this comparison is somewhat prejudiced, it shows that a simple LDA model in some aspects can be superior to more advanced techniques.

	Dog_1	Dog_2	Dog_3	Dog_4	Dog_5	Patient_1	Patient_2
LDA Sensitivity	0.792	0.5	1.000	0.719	0.000	1.000	0.643
Covnet Sensitivity	0.250	0.122	0.071	0.561	0.000	1.000	0.643
LDA Specificity	0.423	0.945	0.345	0.869	0.950	0.464	0.721
Convnet Specificity	0.983	0.999	0.980	0.909	1.000	0.410	0.382

Table 6: Per subject sensitivity and specificity of LDA and convnet on the test set

3.10 Analysis of other solutions

We analyzed solutions of other participants (see Appendix C for an overview): a common approach was to train models on features from small time windows (15-75 seconds) and average the predictions to get a probability for a 10 minute clip, similarly as we described for linear classifiers. In addition to various frequency features, most of the models used pairwise correlations between EEG channels in time and frequency domain, eigenvalues of these correlation matrices, different fractal dimensions, statistical moments of time series, etc. Some features are memory consuming and slow to compute, which may not be appropriate when the speed is crucial and memory of the device is limited. Among popular classifiers were: support vector machines, penalized logistic regression, random forests, gradient boosted trees.

Deep neural networks (up to 6 hidden layers) in combination with k-nearest neighbors were used by a team from Universidad CEU Cardenal Herrera, which won the 3rd prize (Zamora-Martinez et al., 2014). It is worth mentioning that they did not apply calibration. Their best single model was an ANN with 5 hidden layers having a public and private scores of **0.7937** and **0.7644** respectively, which is comparable to our results (see Table 2). The key ingredient to their superiority was the use of Bayesian model combination.

We are aware of one more participant, who used convolutional neural networks and finished on 245th place. His approach was based on bivariate features as in (Mirowski et al., 2009). Presumably, the results were not so positive as described in the paper because of the larger number of channels leading to $\approx 10^2$ more bivariate features, which may be irrelevant and noisy [6].

Private and public leaderboard scores of top 10 models are given in Table 7. There was quite a large shake-up between public and private standings, e.g. on a public leaderboard we had the 16th place.

	Team name	Public score	Private score
1	Medrr	0.90316	0.83993
2	QMSDP	0.85951	0.81962
3	Birchwood	0.83869	0.80079
4	ESAI CEU-UCH	0.82488	0.79347
5	Michael Hills	0.86248	0.79251
6	KPZZ	0.82051	0.79136
7	Carlos Fernandez	0.84225	0.79063
8	Isaac	0.84197	0.78863
9	Wei Wu	0.81803	0.78724
10	**golondrina**	**0.82455**	**0.78513**

Table 7: Leaderboard scores of the top 10 finishers in American Epilepsy Society Seizure Prediction Challenge

[6] Personal communication

4 Discussion

In this thesis we proposed a method for epileptic seizure prediction based on convolutional neural networks. A novel architecture combined with simple frequency features extracted from EEG appeared to work well for American Epilepsy Society Seizure Prediction Challenge, where it finished in the 10th place out of 504 participants.

It was not the first time convnets were applied to analyze EEG or more generally, for multichannel time series analysis(Mirowski et al., 2009; Zheng et al., 2014), however based on our research and experience of other participants, the architecture we proposed suits best particularly for the dataset used in the competition. The idea is straightforward: extract simple DFT features from sliding windows over 10 minute EEG clips to form spectrograms for each EEG channel, learn several layers of representation by performing 1D convolution through time, calculate some temporal statistics over the learnt features and combine this information to make a single prediction.

Other solutions mostly exploited features from small time windows (e.g. 1 minute long) as inputs to the classifiers. Afterward, a prediction for a 10 minute clip was formed by averaging the predictions from each chunk. It may seem to be alike with the concepts we used for building convnets. However, convnets try to assign the probability for the whole clip while other models care only about classification of small EEG pieces. Therefore, our method is original which, unfortunately, it does not make it better. Interestingly, that a simple linear classifier in terms of AUC and on per subject basis can work no worse than a bunch of complicated techniques. However, analysis of sensitivity and specificity for LDA and convnets showed that these methods are quite different in their behavior, so ultimately it may be worthwhile to combine them or exploit separately depending on patient's (doctor's) preferences. We should mention that evaluation measure used in the competition, AUC across subjects, requires a good model to be robust against the choice of the classification threshold. By allowing the use of the test data to calibrate the predictions, organizers, in our opinion, decreased this effect.

Our analysis of models performance was not complete due to the several reasons. For the test data clips we did not know from which hour they came from, so it was not possible to say how many seizures we could actually foresee. Moreover, for each subject it would be interesting to evaluate the whole pipeline: train a model, find a threshold on a validation set, using a test set estimate the number of false positives and negatives along with an average time between seizures onsets and our alarms.

We analysed the difficulties one can confront when doing cross-validation in order to tune the hyperparameters or estimate the holdout error of the models. Despite our CV strategy was not reliable, we hope our findings will enable epilepsy researchers to develop better schemes. We demostrated several intrinsic pitfalls with EEG data, which prevent us to use classical CV: similarity of data clips close in time, class imbalance, different data distribution in train and test sets. In the future, it may be fruitful to explore importance weighted cross-validation (Sugiyama

et al., 2007), which operates under the covariate shift assumption – the case when train and test examples follow different distributions while the conditional distribution of class labels given the inputs remains unchanged.

Our exploration of LDA model shed a light on which features could be important for discrimination of preictal versus interictal clips. We showed that better classification results can be achieved by considering more high frequency bands features, however we did not conduct any formal tests to quantify the importance of each feature, so it will be a direction for our future research (similar study can be found in Park et al. (2011)).

It would be interesting to evaluate our methods on a different dataset, though only a few iEEG datasets are available, and some other problems exist. For instance, iEEG database Freiburg [7] is ubiquitously used in seizure detection and prediction research papers, however many authors use it in a different manner, so the results are not readily comparable. Therefore, we hope that American Epilepsy Society Seizure Prediction Challenge and International Epilepsy Electrophysiology Portal will facilitate the establishment of new standards of datasets exploitation, which in turn allows more accurate review of scientific papers.

Results of the competition are promising, however fair evaluation still has to be carried out. For this reason, the organizers are currently working on generating additional data clips to create previously unseen data. The top 10 models will be evaluated on this held-out set, the results will be published in an academic paper.

[7] http://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database

5 Reference list

Bastien, F., P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *CoRR abs/1206.5533*.

Bengio, Y., I. J. Goodfellow, and A. Courville (2014). Deep learning. Book in preparation for MIT Press.

Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010, June). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Bland, J. M. and S. M. Kerry (1998). Weighted comparison of means. *BMJ 316*(7125), 129.

Buteneers, P. (2012). *Detection of epileptic seizures: the reservoir computing approach*. Ph. D. thesis, Ghent University.

Dieleman, S. (2014). Recommending music on Spotify with deep learning. `http://benanne.github.io/2014/08/05/spotify-cnns.html`.

Dolgin, E. (2014). Technology: Dressed to detect. *Nature 511*(7508), 16–17.

Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. Technical report.

Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics*.

Glorot, X., A. Bordes, and Y. Bengio (2011, April). Deep sparse rectifier neural networks. In *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*.

Hinton, G. and T. Tieleman (2012). Lecture 10.1—Why it helps to combine models. COURSERA: Neural Networks for Machine Learning.

Howbert, J. J., E. E. Patterson, S. M. Stead, B. Brinkmann, V. Vasoli, D. Crepeau, C. H. Vite, B. Sturges, V. Ruedebusch, J. Mavoori, K. Leyde, W. D. Sheffield, B. Litt, and G. A. Worrell (2014). Forecasting seizures in dogs with naturally occurring epilepsy. *PLoS One 1*(e81920).

Kaggle (2014). American epilepsy society seizure prediction challenge. `http://www.kaggle.com/c/seizure-prediction`.

LeCun, Y., L. Bottou, B. Y., and P. Haffner (1998, November). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Mirowski, P., D. Madhavan, Y. LeCun, and R. Kuzniecky (2009). Classification of patterns of eeg synchronization for seizure prediction. *Clinical neurophysiology 120*(11), 1927–1940.

Mormann, F., R. G. Andrzejak, C. E. Elger, and K. Lehnertz (2007). Seizure prediction: the long and winding road. *Brain 130*(2), 314–333.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nair, V. and G. E. Hinton (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 807–814.

Park, Y., L. Luo, K. K. Parhi, and T. Netoff (2011). Seizure prediction with spectral power of eeg using cost-sensitive support vector machines. *Epilepsia 52*(10), 1761–1770.

Pascanu, R., Y. N. Dauphin, S. Ganguli, and Y. Bengio (2014). On the saddle point problem for non-convex optimization. *CoRR abs/1405.4604*.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1929–1958.

Sugiyama, M., M. Krauledat, and K.-R. Müller (2007, December). Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res. 8*, 985–1005.

van der Maaten, L. and G. E. Hinton (2008). Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9*, 2579–2605.

Zamora-Martinez, F., F. J. M. Almaraz, and J. Pardo (2014). Kaggle-epilepsy. `https://github.com/ESAI-CEU-UCH/kaggle-epilepsy`.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *CoRR*, –1–1.

Zheng, Y., Q. Liu, E. Chen, Y. Ge, and J. Zhao (2014). Time series classification using multi-channels deep convolutional neural networks. In *Web-Age Information Management*, Volume 8485 of *Lecture Notes in Computer Science*, pp. 298–310. Springer International Publishing.

Appendix A: models from the final ensemble

	Data	Architecture	Public (min-max)	Private (min-max)
1	6 bands + std; 1 min windows; 9 min overlap	C1: 32@1x10, ReLU C2: 64@1x5, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.80872	0.77182
2	6 bands + std; 1 min windows	C1: 16@1x10, ReLU C2: 32@1x5, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.80614	0.76014
3	6 bands + std; 2 min windows; 4 min overlap	C1: 32@1x5, ReLU C2: 64@1x2, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.80533	0.79296
4	8 bands + std; 2 min windows; 4 min overlap	C1: 16@1x5, ReLU C2: 32@1x5, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.80192	0.78114
5	8 bands + std; 1 min windows; 8 min overlap	C1: 16@1x10, ReLU C2: 32@1x5, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.79964	0.77905
6	6 bands + std; 1 min windows	C1: 16@1x10, ReLU C2: 32@1x5, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.5 LogReg: dropout 0.5	0.79886	0.76305

	Data	Architecture	Public (min-max)	Private (min-max)
7	6 bands + std; 1 min windows; 8 min overlap	C1: 16@1x10, ReLU C2: 32@1x5, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.79882	0.76817
8	8 bands + std; 2 min windows; 4 min overlap	C1: 16@1x5, ReLU C2: 32@1x4, ReLU GP3 F4: 1024, tanh, dropout 0.3 LogReg: dropout 0.6	0.79865	0.78916
9	6 bands + std; 30 sec windows	C1: 16@1x20, ReLU C2: 32@1x10, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.79833	0.77090
10	6 bands + std; 2 min windows	C1: 16@1x5, ReLU C2: 32@1x5, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.78967	0.76209
11	8 bands + std; 30 sec windows; 5 min overlap	C1: 16@1x20, ReLU C2: 32@1x10, stride 2, ReLU GP3 F4: 512, tanh, dropout 0.3 LogReg: dropout 0.6	0.78612	0.76133

Table 1: Convnet models from the final ensemble

Appendix B: effect of the calibration schemes

Model \ Calibration	No calibration	Softmax	Min-max	Median
ensemble	0.78572	0.74418	**0.79560**	0.75959
single convnet	0.76611	0.74449	0.77291	**0.78146**
LDA 8 bands	0.74766	**0.79679**	0.74418	0.78590

Table 2: Global AUC calculated over the test set for model-calibration pairs.