Prediction of seizures using canine EEG data and deep learning techniques: a comparison between convolutional, and recurrent neural networks

Roberto Vega
Department of Computing Science
University of Alberta
rvega@ualberta.ca

Abstract

Epilepsy is a neurological disorder that diminishes the quality of life of patients who have it. It most devastating characteristic is the sudden appearance of seizures. Being able to predict a seizure before it happens has the potential to allow patients to take preventive measures to avoid sever damage. This project analyzes the use of deep learning techniques for the prediction of seizures. Using data extracted from dogs, the task is to classify a given EEG segment as being part of interictal (normal activity of the brain) and preictal (before a seizure) stages. We evaluate the use of convolutional, recurrent, and feed-forward neural networks for this purpose. Recurrent neural networks achieved the best result with an F-score above 0.7 in 3 out of 4 cases.

1 Introduction

Epilepsy is defined by the World Health Organization as a chronic neurological disorder characterized by recurrent seizures[1]. It is estimated that it affects over 50 million people worldwide, and it is one of the most debilitating neurological conditions [1]. A particularly disabling aspect of epilepsy, and that has a strong impact in the quality of life of patients, is that seizures can appear suddenly, without any apparent warning [2]. This has motivated researches to look for a seizure warning system that could alert patients of a seizure before it happens, so they can modify their activities accordingly; however, this is still considered an open problem [3]

Generally speaking, the seizures can be divided into 4 stages[2]: 1) precital, which is the time before the seizure, 2) ictal, the actual seizure, 3) postictal, the time immediately after the seizure, and 4)interictal, the time between seizures. The challenge of seizure prediction consists in distinguishing between the interictal and the preictal stages (where the brain behaves "normally").

The primary tool for the detection and prediction of seizures is electroencephalography (EEG). EEG estimates the brain activity by measuring the electric potentials generated by the brain using electrodes located either on the scalp, or directly on the brain [4]. Currently, most of the EEG analysis is done offline –i.e. EEG data is recorded from several patients and then it is analyzed to create the algorithms that can distinguish between the interictal state and the preictal state. One of the biggest challenges for seizure prediction is that seizure manifestations in EEG can change drastically from one patient to another [4], therefore usually patient-specific predictors are created. A typical pipeline for the classification of EEG signals follows three steps: preprocessing, feature extraction, and learning a model. The objective of the preprocessing stage is to decrease the noise in the raw EEG signals.

[1] http://www.who.int/topics/epilepsy/en/

[2] https://www.epilepsycolorado.org/wp-content/uploads/2016/01/2-Types-of-Seizures.pdf

Feature extraction consists in the computation of hand-crafter features that are believed to distinguish between the different stages of the seizure. Finally, a learning algorithm is used to learn a model that allows for further classification of new, previously unseen EEG segments.

A second challenge for creating a system that can predict seizures is the availability of data. Usually the electrodes are placed in patients for a few days, and they rarely contain enough number of seizures to perform an adequate analysis. In order to overcome this obstacle Brinkmann et. al collected EEG records from 5 dogs suffering from naturally occurring epilepsy [1]. They implanted the dogs a 16-channel, intracranial, EEG recording device and monitored their brain activity during an average of 380 days, capturing between 24 - 97 seizures per dog. Interestingly, canine epilepsy show many similarities with human epilepsy, such as similar interictal and ictal patterns, a similar rate of occurrence, and similar rate of drug resistance [5]. This makes canine epilepsy a good alternative for studying the differences between preictal and interictal EEG patterns in epilepsy [1].

In recent years, deep neural networks have achieved state of the art results in classification of images, video, speech and text; however, they have not been widely adopted by the neuroimaging community [6]. The objective of this project is to use some of the deep learning techniques to analyze the canine EEG data collected by Brinkmann et. al [1] in order to learn a model that can distinguish between the interictal and preictal stages in epilepsy. Since we expect to have high variability between the seizure manifestations in different dogs, we will train a model tailored to every dog. The rest of the report is structured as follows: section 2 describes some approaches that have been used for seizure prediction using EEG data. It also provides an overview of the deep learning techniques used in this project. Section 3 describes the exact problem that we are solving, the dataset used, and the questions that this project attempts to solve. Section 4 describes the pipeline that we followed in our experiments: data preprocessing, neural network architecture selection, and evaluation metrics. Section 5 describes the best models found and their performance. Finally, section 6 mention the conclusions and lessons learned.

2 Literature review

Several attempts have been made in the literature for solving the problem of prediction and detection of seizures using EEG data. Alotaiby et al. [7] provide a survey of these techniques, which they classify in 6 approaches: Time domain, wavelet domain, frequency domain, PCA and ICA domain, singular value decomposition, and empirical mode decomposition. Depending on the exact task to solve, and the characteristics of the dataset, the reported accuracies range from sensitivities of 60% to 100%. Despite these promising results, the specific challenge of detecting a seizure before it happens (seizure prediction) still remains elusive [8].

One of the challenges to successfully perform seizure predictions is the scarcity of data. Even if several patients with epilepsy are recruited, the amount of EEG recordings that show seizure activity is relatively low [1]. Having an unbalanced dataset might prevent machine learning algorithms to learn a good classifier. In an effort to overcome that problem Park et al. proposed the use of cost-sensitive support vector machines. A combination of SVM and Kalman filter to post-process the output of the machine learning algorithm achieved a sensitivity of 97.5% with a total of 80 seizure events, with an alarm rate of 0.27 per hour [8]. Brinkmann et. al tried to solve the scarcity of the data using a completely different approach: they used dogs with naturally occurring epilepsy to overcome the problem of data scarcity. After extracting some features in the frequency domain, they used a SVM to achieve better than chance predictions in 5/5 dogs.

The recent success of deep learning has attracted the attention of researchers in the neuroimaging community. For instance, Mirowski et al. compared the performance of SVM, logistic regression, and convolutional neural networks for the task of seizure prediction in intercranial EEG [3]. Their results show that convolutional neural networks achieved the best performance with a sensitivity of 71%, with the advantage of having no false alarms (no false positives) in 15 patients. Their strategy consisted in creating a 2-D matrix that has time in one of its axis. The other axis are just features that can be considered as independent of each other. Then, they applied 1-D convolution across every row in the matrix in order to create the classifier. Although very creative, this approach does not deal explicitly with the temporal sequence of the data.

On a different field, Ng et al. proposed the combination of convolutional neural networks (CNN) and recurrent neural networks (RNN) to perform video classification [9]. Their idea was to treat video

classification as a sequence of individual images. They extracted features from the images using a CNN, and then used their outputs as inputs to a recurrent neural network. Using Long Short-Term Memory (LSTM) cells they achieved accuracies up to 88.6% in sports video classification. Borrowing ideas from this paper, Bashivan et al. proposed to use a similar approach (combining CNN with RNN) to perform classification in a working memory experiment [6]. Unlike video classification, EEG data does not naturally encode spatial information. In order to take full advantage of CNN, they proposed to map the 3-D location of the electrodes position to a 2-D image. Then, using frequency domain features and cubic interpolation they created the equivalent to an RGB image, where each channel encodes information of some frequency bands. With this approach, their new image now encodes the position of electrodes, creating what they call a *spatially invariant detector*. Thodoroff et al. took this idea and applied to the task of seizure detection using EEG data [4]. They reported an average sensitivity of 85%, with a false positive rate 0.8 per hour. Following this line of thought, I want to continue exploring the performance of deep learning algorithms on EEG data for seizure detection. I want to empirically measure the effect that methodologies for dealing with unbalanced dataset and regularization have on the final prediction results. At the same time, I want to compare the performance of convolutional,recurrent, and feed-forward neural networks for the prediction of seizures in canine epilepsy.

2.1 Quick deep learning overview

As described by Mesinl et al., *the objective of machine learning is to discover statistical structure in the data*. In order to discover this statistical patterns it is necessary to transform the raw data into features that make easier the supervised learning task [10]. One of the biggest challenges is to determine these set of features that will allow learning. Traditionally, these features are extracted by introducing expert knowledge in the preprocessing of the data. Given the challenge that s to discover relevant features, an appealing idea is to have algorithms that can use the framework of *self-taught learning* [11]. Under this paradigm, it is possible to use unlabeled data to learn higher-level representation of the inputs. These representations can be then used in conjunction with supervised learning to (hopefully) achieve good classification results.

One of the first ideas was the use of autoencoders to learn the high level representation of the raw data [12]. An autoencoder is a learning circuit that tries to transform it input into a different representation, and then reconstruct the output from that new representation with the least amount of distortion. As described by Baldi [12], the general idea was to stack several of this autoencoders together, creating a deep structure, that could learn useful features. Finally, the last layer of the autoencoder could be connected to any learning algorithm, such as a feed forward neural network, or SVM to create a classifier.

Two of the most important model in deep learning are convolutional networks and recurrent neural networks. A full description of this structures is outside of the scope of this report. Here I just provide a shallow overview of both structures. Section 4 describes in detail the network architecture that I used in this project.

- **Convolutional neural networks** are neural networks specialized in processing data with a grid-like topology, such as time series, or images [13]. Their main characteristic is that they use convolution instead of the traditional matrix multiplication used by the more commonly known feed-forward neural networks. The three key elements of CNN, according to Goodfellow et al [13] are: sparse interactions, parameter sharing, and equivariant representations. This allow for a relatively efficient training of the networks (compared with not using sparse interactions not parameter sharing). Also, they cause an interesting effect know as equivariance to translation. The second key element of CNN is the pooling layer, which usually takes the form of a sum, an average, or a max [14]. According to Boreau et al., the pooling layers allow "invariance to small image transformations, more compact representations, and better robustness to noise" [14]. The combination of these two elements turned out to be a game changer in the task of image recognition, as shown by the results of Krizhevsky et. al [15], which improved state of the art result in the task of image classification.
- **Recurrent neural networks** are a family of architectures focused on the processing of sequential data [13]. Similar to the CNN, the sharing of parameters is a key element of this structures. Although CNN allows the use of 1-D kernels to perform convolution across time, it has the "disadvantage" that is is shallow. In RNN is possible to create deep structure

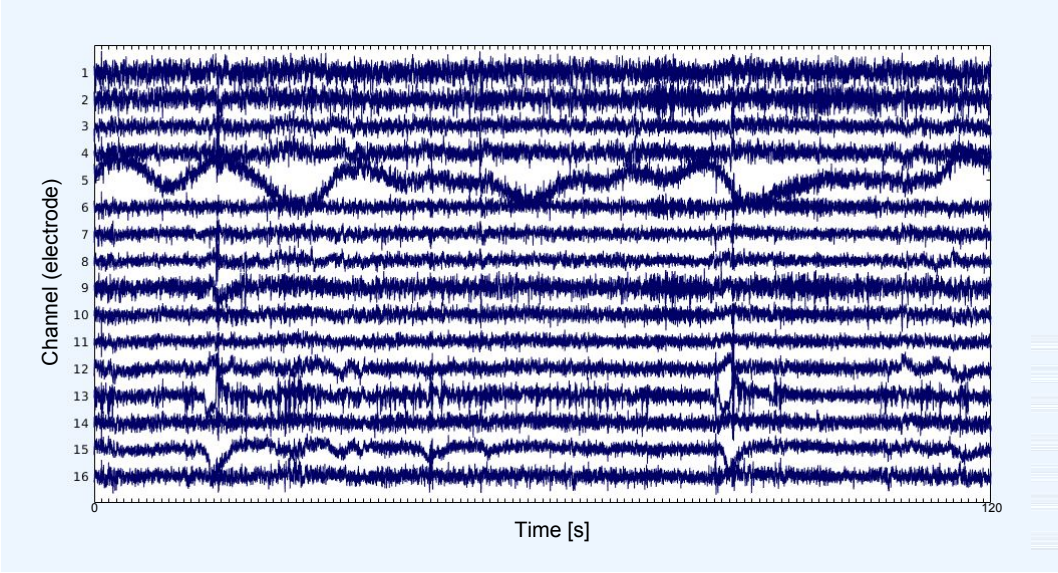

Figure 1: EEG segment of 120 seconds. The data was extracted using 16 electrodes placed directly in the brain of the animals

in which each member of the output function is a function of the previous member of the output. The main idea behind these networks is that the hidden units can be seen as "a lossy summary of the task-relevant aspects of the past sequence of inputs up to the current time point".

3 Problem statement

The learning task that this project addresses is: Given a set of EEG recordings extracted from 4 dogs[3], learn a model for each dog that can distinguish between: a) EEG segments that are labeled as being part of the interictal stage, and b) EEG segments labeled as being part of the preictal stage. For purposes of this project, the preictal stage was defined as the recordings that span the 60 minutes previous to a seizure, while the interical stage was defined as the period of time elapsed from 4 hours after the last seizure until the start of the preictal stage. Table 1 shows the number of EEG segments available for every dog. Every segment consist of 120 seconds of EEG activity recorded through 16 channels at a frequency of 400 Hz. Since most of the data belongs to the interictal state, we used a random sample of the EEG signal in this state. The random samples were released in a public competition hosted by Kaggle [4]. Figure 1 shows how an EEG segment looks like.

The four questions that this project attempts to answer are:

- Given the relatively short duration of the signal, it is possible to encode the temporal information as part of the feature matrix (see section 4 for details). What is the difference in performance of using this approach and a convolutional neural network and using a recurrent neural network that explicitly deals with time series? Do these approach represent any advantage over using a traditional feed-forward neural network to solve this problem?
- As shown in Table 1, the number of instances in the preictal stage is much lower than the ones in the interictal stage. An intuitive solution is to use an oversampling technique to deal with this situation; however, by doing this we would be learning a model on balanced data that would be then used on in an unbalanced scenario. Are the deep learning models constructed robust enough to learn from an unbalanced dataset? Can they learn if we artificially balanced the dataset? If so, is the performance kept when using the model in the original unbalanced scenario?

[3]Dataset collected by Brinkman et al. [1], publicly available at https://www.ieeg.org/

[4]https://www.kaggle.com/c/seizure-prediction

Table 1: Number of samples per dog

Dog id	# Interictal	# Preictal
1	2,400	120
2	2,500	210
3	7,200	360
4	4,020	485

- Deep learning models usually have to learn many parameters, which might lead to overfitting. This is particularly important in this project due to the relatively small datasets used. Dropout regularization has been proposed to help to mitigate the risk of overfitting [16]. What is the effect of adding dropout regularization to our model? Can it compensate for the small dataset size?
- The intuition behind deep learning models is that deeper layers learn higher level features. The hope is that these higher level features will be shared by different classes, and they would be more robust to the differences found among datasets [10]. Our original experiments were designed to create a model per every dog. Are the features learned by deep learning models robust enough to be able to generate a general classifier for the 4 dogs used in this project?

4 Experimental design

4.1 Data preprocessing

Since every segment spans 120 seconds of EEG signal, and it has 16 electrodes (channels) sampling at a frequency of 400 Hz, we have $120 \times 16 \times 400 = 768,000$ timepoints per segment. This is a high dimensionality compared to the number of available instances for training. At the same time, brain signals are non-stationary (their statistics change over time) and important information is found in the frequency domain, rather than in the time domain [6]. These three factors motivate the use of preprocessing to reduce the dimensionality of the data and to extract meaningful features. The preprocessing approach that I applied consist in the following steps:

1. Divide every segment of 120 seconds in 12 segments $s_i, i \in \{1, 2, \ldots, 12\}$ of 10 seconds each, as shown in Figure 2.
2. Every segment s_i consists of 16 channels $c_j^i, j \in \{1, 2, \ldots, 16\}$. For all pairs (i, j), extract the signal of the channel c_j^i and compute the spectrogram in the range of frequencies from 0 - 50 Hz [8] (See Figure 3 a and b)
3. Compute the average power on the bands of interest: delta (0-4 Hz), theta (4-8 Hz), alpha (8-12 Hz), beta (12-38 Hz) and gamma (> 38 Hz). This will produce a vector of length 5 for every c_j^i. (See Figure 3 c)
4. Repeat this procedure for all the channels in s_i, and all the 12 segments s_i (See Figure 3 d and e)

The output of this entire process is a 3-D matrix of dimensions $c \times b \times s$, where *c* is the number of channels (16), *b* is the number of bands of interest (5), and *s* is the number of subsegments (12). This matrix will be the input to the different learning algorithms.

4.2 Model architecture

In order to compare the performance of feed forward neural networks, convolutional neural networks, and recurrent neural networks we first need to define the model architecture of every one of these networks.

- **Feed-forward neural network:** As a baseline for assessing the gain that we get for using more complex models, we will use a simple feed-forward neural network with two hidden layers, as shown in Figure 4. To feed this network we simply expand the 3-D matrix created

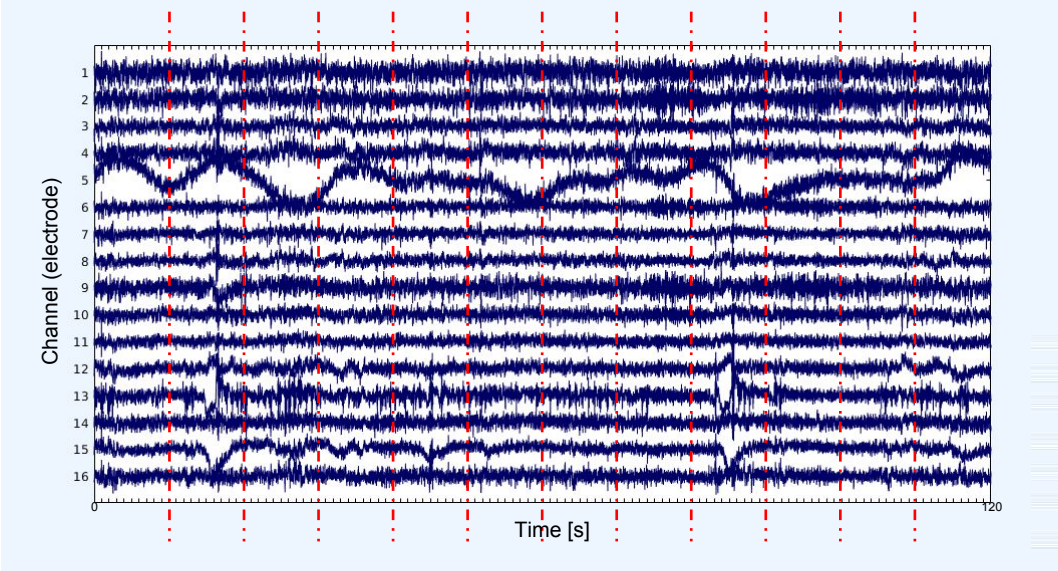

Figure 2: Every segment of 120 seconds is divided into 12 subsegments of 10 seconds each.

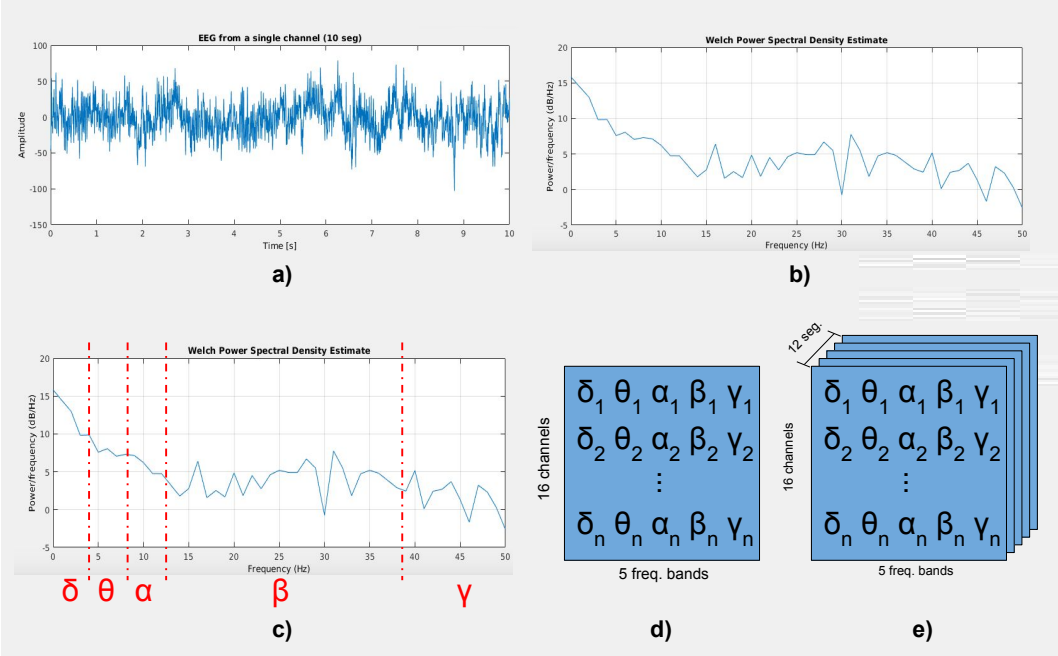

Figure 3: Preprocessing of the EEG signal. a) 10 seconds of EEG signal. b) Spectrogram of the signal. c) Computation of the mean power in the bands of interest. d) Compute the power in the bands of interest for every channel. e) Repeat this procedure to all the 12 segments of the EEG signal.

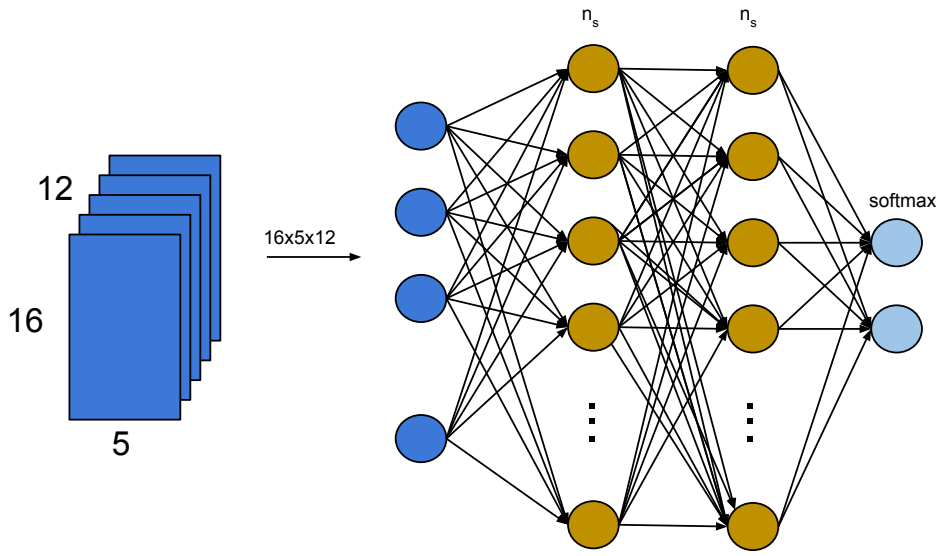

Figure 4: Feed forward neural network used as a baseline. It contains 2 hidden layers with *n* neurons each. The last layer applies *softmax*.

in the preprocessing stage into a feature vector of length $16 \times 5 \times 12$. The number of neurons in the hidden layer is a tunable parameter that we selected using a validation set. We made experiments with $n \in \{1000, 5000, 10000\}$ neurons. Both layers are fully connected and the output are two neurons (one for each class). We convert the output to a probability vector via the softmax function. There are several configuration parameters that can be used for this network, since we will use it as baseline we used the Rectifier Linear Unit (ReLU) as activation function, stochastic gradient descent as optimizer, and softmax cross entropy as loss function.

- **Convolutional neural network:** We used the architecture shown in Figure 5. It consists in two convolutional-pooling layers. In the first layer we used n_1 filter of size $1 \times 3 \times 5$, and for the second one we used n_2 filters of size $1 \times 3 \times n_1$. Since the channels axis does not represent spatial information the size of the filter along this axis is 1, –i.e. the channel 2 is not necessarily located next to channel 1 and 3, so I did not attempt to find "local information" along this axis. For the pooling step, we used max-pooling filters of size $1 \times 1 \times 2$ for the first layer and $1 \times 1 \times 3$ for the second layer. Finally, the last layer is a fully connected layer with a softmax function at the end. I determined the size of the convolutional filters $(n_1, n_2) \in \{(64, 32), (128.64), (256, 128)\}$ using a validation set. Based on the results of previous experiments with the MNIST dataset, I selected the ReLU gate as the activation function; however, I experimented with three different loss functions: softmax cross entropy, expected reward with maximum entropy regularizer, and reward-augmented maximum likelihood [17].

- **Recurrent neural network:** The architecture for this network is shown on Figure 6. It consists in two hidden layers with 1,200 neurons on each. Since these networks take considerable more time to train, I did not explore different layer sizes. My selection was based on previous experiments done with these network in character level language models that have an input size of similar dimensionality. Based on the result of those experiments I also decided to use ReLU as the activation function of the neurons. Another alternative was to use the Long Short Term Memory units [18]; however, the performance of the simple RNN with ReLU gates was better than LSTM in our experiments with language models, so I did not pursue that direction. Similar to the case of convolutional neural networks, I experimented with three loss functions:softmax cross entropy, expected reward with maximum entropy regularizer, and reward-augmented maximum likelihood.

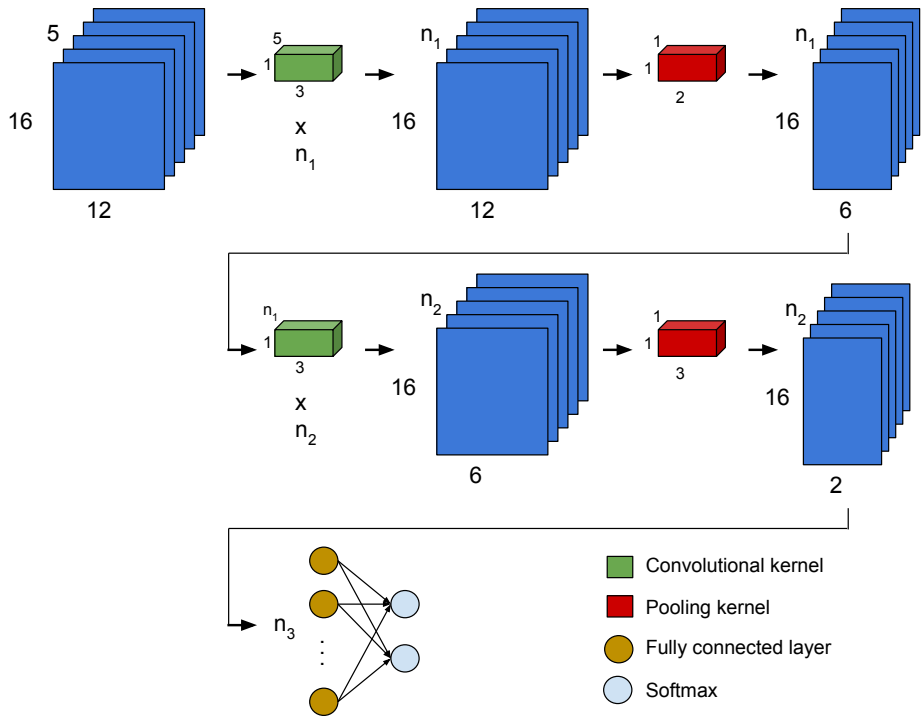

Figure 5: Architecture of the convolutional neural network. It consists of two convolutional-pooling layers. The number of filter in every convolutional layer is a tunable parameter. Note that in both layers the convolution is performed across the time (subsegment) axis. The number of filter and the number of neurons in the last layer is a tunable parameter.

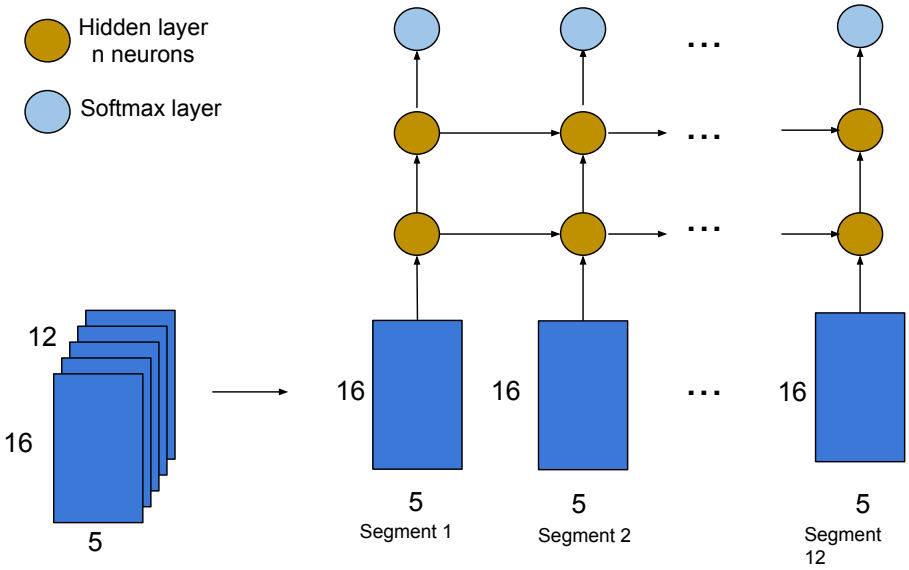

Figure 6: Architecture of the recurrent neural network. It consists of two layers with n neurons each and a softmax layer at the output.

4.3 Experiments

In order to determine the generalization capability of the architecture selected, I decided to select the tunable parameters of the different architectures by experimenting in data extracted from only one (out of four) dogs. Besides determining the size of the filters and the loss function to use, I used the dataset of the same dog to analyze the influence of oversampling and dropout regularization. After analyzing the results and choosing the best combination of parameters for this dog, I fixed the architecture of the networks and trained them with the rest of the dogs in order to compare the performance between convolutional, recurrent and feed-forward neural networks. Note that although the architecture will be the same for all dogs, the weights learned will be different for each dog, since we train a model for each one of them. For all the experiments, we divided the dataset into *Train set*: 64%, *validation set*: 16%, and *test set*: 20%.

Specifically, in order to answer the questions posed in Section 3 we will perform the following experiments implemented in TensorFlow 0.12 [5].

- **Experiment 1. Imbalanced dataset:** If the imbalance in the dataset prevents the models for learning, the choice of the architecture will make no difference. Therefore, solving this problem is the first step in the experiments. I separated the data from Dog 1 in two datasets: *Test set* (20%) and *TempTrain* (80%). I then used the algorithm SMOTE (synthetic minority over-sampling technique) [19]. This algorithm artificially creates data from the minority class until a certain ratio of minority/majority class is achieved. We compared the performance of the convolutional neural network and the recurrent neural network with: 1) No class imbalance correction, 2) A ratio minority/majority class of 0.5, 3) a ratio minority/majority of 1. Finally, I divided *TempTrain* into the *train set* and *development set* as previously described. We used the publicly available library imbalanced-learn [20].
- **Experiment 2. Architecture:** After deciding the best way to deal with the unbalanced dataset I used data from the Dog 1 to determine the number of units in the hidden layer for the feed-forward neural network, the number of kernels and loss function to use in the convolutional neural network, and the loss function for the recurrent neural network. I selected the best parameters based on the final performance on the *test set* in Dog 1.
- **Experiment 3. Dropout regularization:** I used the best architecture to analyze the effects of using regularization in the convolutional and recurrent neural networks. Using again data from Dog 1 and the best architecture found, I used dropout after the pooling layers in case of the CNN, and after the second hidden layer in the case of the RNN. Dropout has one tunable parameter that controls the probability that a certain unit will be set to zero. I experimented with three values: No dropout, 50%, and 90%. The suggested value is usually 50% [16, 6].
- **Experiment 4. Networks comparison:** After the first 3 experiments, I fixed the architecture of the three networks. I then used the selected architecture and methodology for dealing with the imbalanced dataset and learned a model for every dog. I then compared the performance of the three networks on the 4 dogs.
- **Experiment 5. Single model:** Finally, I joined the data of the 4 dogs into a single dataset and used the best architecture found to train a new model (one for every network) and analyzed the performance of this new classifier.

4.4 Performance evaluation

Due to the imbalance of the dataset, accuracy is not a good metric to use. A dummy algorithm that predicts *interictal* all the time would achieve an accuracy > 90% without learning anything. A better metric for this case would be the F-score, which is the harmonic mean of precision and recall and can be computed as:

$$F_{score} = \frac{2tp}{2tp + fp + fn}$$

where *tp* are true positives (in this case a true positive is a preictal state correctly identified), *fp* are the false positives, and *fn*. Note in Figure 6 that the recurrent neural network produces an output in

[5] https://www.tensorflow.org/

every step. For evaluation purposes at testing time we only considered the output at the end of the network (after seeing the 12 segments).

5 Results

5.1 Dataset Imbalance

Figure 7 shows the results of applying SMOTE to the dataset extracted from Dog 1 in both: convolutional neural networks and recurrent neural networks. The most relevant effect observed in the graphs is the low performance obtained when the dataset imbalance is not addressed at all. Both, the RNN and the CNN quickly decrease the training loss; however, this increase in performance is meaningless, since it can be easily achieved by learning a set of weights that always predict the majority class. This effect is immediately perceived in the validation set and the test set, where the F-score is very low.

An interesting phenomena occurs in the convolutional neural network. Note that as the number of batches start to increase, the F score on the test set start to increase too, despite having almost a constant training loss. This might be an indication that as the training time increases, the networks makes an effort to correct the misclassification made on the train set. Despite this effect, oversampling the minority class has an evident payoff in this experiment. For the case of the convolutional neural network the train loss is very similar independently of the oversampling ratio selected (and even of not doing oversampling a all); however, there is a dramatical improvement on the validation set and on the test set. For the case of the recurrent neural network, the training loss increases when we use oversampling, but it also achieves a F-score > 0.9 in the validation set and around 0.5 in the test set.

It is important to note the difference in performance between the validation and test set. The validation set has an almost perfect F-score, but the performance decreases by half on the test. Two possible explanations for this phenomena are: The validation set contains oversampled data. The method used by SMOTE for oversampling is to create a linear combination of a given point and its nearest neighbors. This means that it is likely that the new data created is very similar to the original points in the dataset. Since I divided the dataset after oversampling it is highly likely that very similar points are in both: training and validation, giving a false impression of good performance. The data on the test set, on the contrary, is not generated artificially, so it gives a better estimation of the real performance of the method. A second explanation of the diminished performance on the test set is that we are learning a classifier on a balanced test set, but we are testing it on a dataset with a different class distribution. This effect is known as prior probability shift and, if left unattended, might hurt the performance of the classifier [21]. To understand better why this is the case visualize a classifier that tries to estimate $P(y|x)$. From the Bayesian point of view we can classify to the $argmax_y P(x|y)P(y)$, but $P(y_{train}) \neq P(y_{test})$, which might generate a decrease in performance.

We can also observe that the performance on the test set is slightly better when we oversample to make the ratio of minority/majority class 0.5. This is the parameter that I chose for the rest of the experiments. Although the model is still unbalanced, it is able to learn better and faster than not oversampling at all. One possible explanation for this improvement is that by oversampling we are indirectly penalizing more mistakes made in the minority class –i.e. the artificial data will (presumably) be very similar to the original datapoints, so if the network missclassifies one it will presumably misclassify the artificially generated datapoints similar to the original. This will in turn increase the loss, (ideally) "forcing" the network to look for a better set of parameters.

5.2 Architecture

The objective of this experiments were only to select the best parameters for the three different models. Table 2 shows the final parameters selected for every architecture. The parameters were selected based on the F-score achieved in the test set.

As described in the previous section, I implemented three different models for the feed-forward neural network: one with 1,000 neurons, one with 5,000, and the last with 10,000. It is interesting that the smallest network achieved the best results. Note that we have two hidden layers, so the number of parameters largely exceeds the number of training instances. Since no regularization technique was used here it is possible that the other two more complex network were overfitting to the training set. A similar rationale follows the selection of parameters for the convolutional neural network. The most surprising result came from the selection of the loss function to train the model. It is remarkable

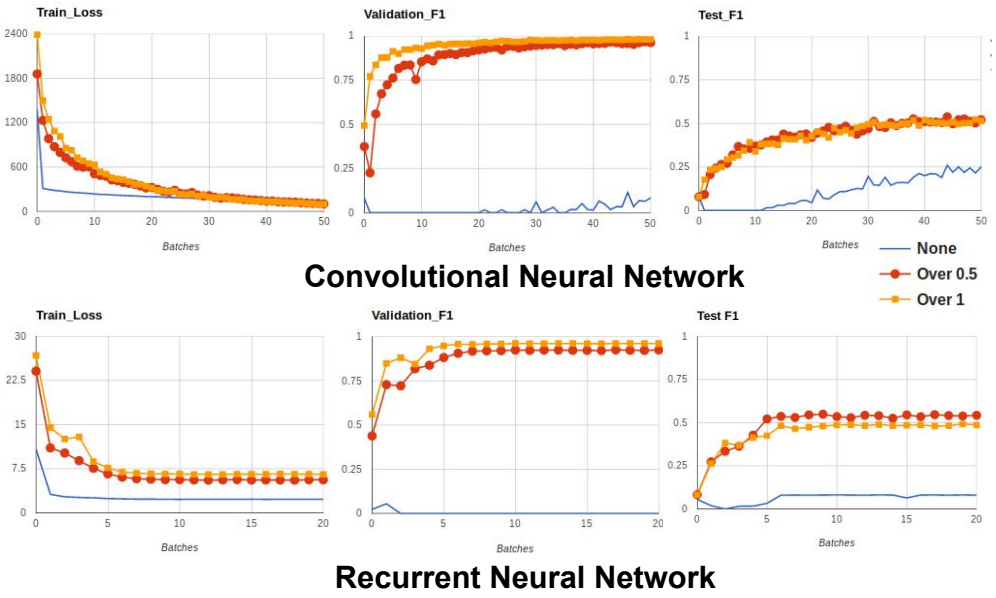

Figure 7: Results of applying oversampling in the CNN and RNN. The graphs on the left show the evolution of the train loss with the batches, the middle ones show the results in the validation set, the ones on the right show the performance on the test set.

Table 2: Parameters selected for every architecture

Model	# Parameters	Loss function
Feed-forward NN	$n = 1000$	Softmax cross entropy
Convolutional NN	$n_1 = 128$	Softmax cross entropy
	$n_2 = 64$	
Recurrent NN	$n = 1,200$	Reward augmented ML

that the recurrent neural network performed better with the reward augmented maximum likelihood, rather than with the more commonly used softmax cross entropy. Figure 8 shows the performance of the different losses in the convolutional and recurrent neural network. Note for the RNN the entropy regularized expected reward is missing. The reason is that, despite several efforts to tune the model, the loss did not converge.

For the recurrent neural network, the reward augmented maximum likelihood clearly outperforms the softmax cross entropy in the training set. It would be tempting to think that this is because of overfitting, like in the previous experiments; however, the performance on the test set is practically the same than the softmax cross entropy. It is important to mention that these results are the average over 7 repetitions of the experiments, so it is unlikely that it was just because of chance. It is also interesting than the same effect is not present in the convolutional neural network, where it achieves also the lowest train loss, but is slightly outperformed by the softmax cross entropy.

5.3 Dropout regularization

This experiment also yielded surprising results. According to the current literature in deep learning, the use of Dropout generalization improves the performance of the networks [16, 6, 4, 13]; however that was no the case in this study. Figure 9 shows the results of applying different degrees of dropout to the networks. In this regularization approach, every neuron might be set to zero with a probability λ.

A good heuristic is to set $\lambda = 0.5$ [16], but in this experiment performing no regularization at all is better than using the recommended value of 0.5. Increasing λ to 0.9 only damages the performance even more. These results are counter intuitive. A possible explanation for this behavior might be that

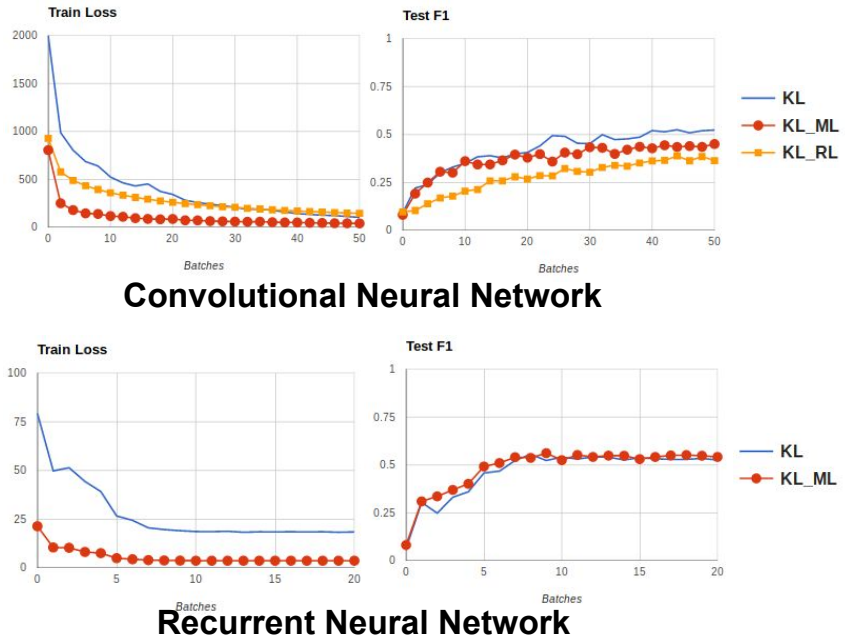

Figure 8: Results of training the CNN and RNN with different loss functions: softmax cross entropy (KL), reward-augmented maximum likelihood (KL_ML), and entropy regularized expected reward (KL_RL).

the dataset is relatively small. The idea of dropout regularization is to emulate the training of several networks with different architectures at training time with the objective of simulating an average response of these networks at test time. It might be possible that the dataset is too small to allow the network to learn those different models, leading to a lower accuracy than training exclusively one single model. For the rest of the experiments we decided not to use dropout regularization.

5.4 Classification results

The last experiment consisted in using the networks shown in Figures 4, 5 and 6 with the parameters shown in Table 2 to classify between the interictal and preictal stages in the four dogs of the dataset. We decided to not use dropout and use oversampling in the training set with a minority/majority class ratio of 0.5. It is important to remind that all the decision with respect to the network parameters where done by analyzing the F-score achieved on the test set of the Dog 1.

Figure 10 shows the results of applying the three networks to the datasets extracted from the different dogs. It is interesting that all the decisions regarding the structure of the network were made using data from Dog 1, which turned out to be the one with the lowest performance. Also, an interesting effect can be observed here: the convolutional neural network achieved the best results in Dog 1, followed by the RNN, and finally the feed-forward neural network. Nevertheless, this pattern is very different in the other dogs, which were not used for making any decision. Besides, the pattern in the other 3 dogs looks very consistent: the recurrent neural network clearly outperforms the other two methods, while the convolutional neural network and the feed-forward neural network have practically the same performance.

During this project, I was doing several experiments with the different networks. I did not put too much effort in tunning the feed-forward neural network, since it was meant to be just a baseline against which to compare the deep learning approaches. The recurrent neural network, on the other side, is more computationally expensive to train, so the amount of tunning possible was limited. The convolutional neural network was the opposite. The training was very efficient, which allowed me to perform more experiments and tune the network better; however, all these extra tunning might have lead to overfitting to the dataset from Dog 1. It is interesting than the results observed in this dog are proportional to the amount of time spent tunning the network. The remaining three datasets, on

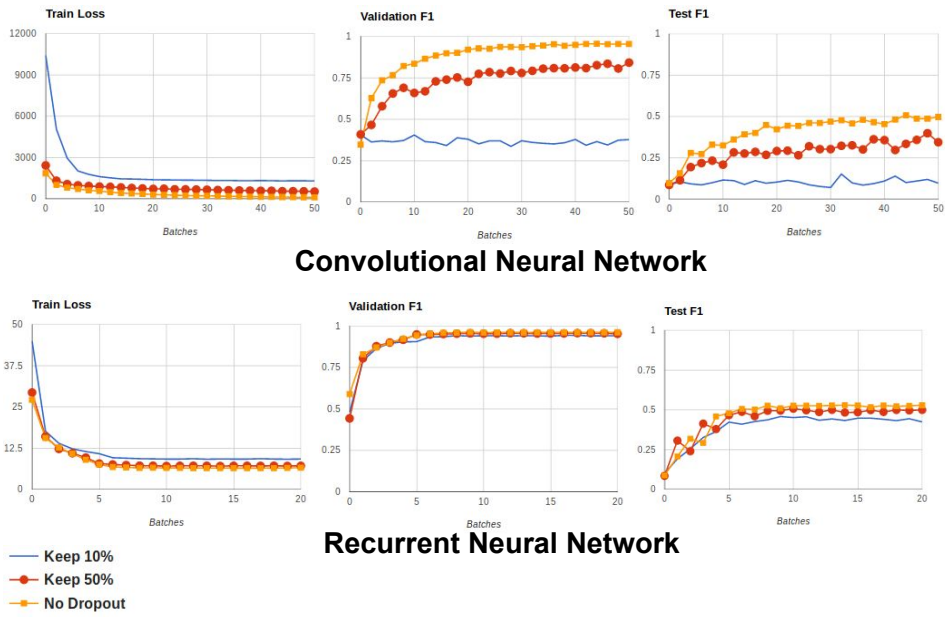

Figure 9: Results of applying dropout regularization to the CNN and RNN. Note that no applying regularization at all achieves the best results.

the other side, were used only once in this project (technically, they were used 10 times, since the F-score shown on Figure 10 is the average over 10 repetitions), so they are a better estimator of the performance that we expect to achieve on other datasets.

It is also remarkable that the dataset from Dog 2 achieved he best results, with an average F-score above 0.85. Note in Table 1 that the size of this dataset is very similar to the one of the Dog 1. Dogs 3 and 4 contain considerably more data than the other two datasets; however, their performance is worse. This is a strong indication that the amount of data is not the only limiting factor to achieve success in this task. True, having less data only makes the task more complicated, but simply increasing the dataset is not guaranteed to improve the performance. Additional research is needed to better understand what are the features that might be relevant for distinguishing between preictal and interictal stages in EEG data.

The final experiment consisted in merging the 4 datasets together, and then learning a new convolutional neural network model (although the architecture of the network was the same than in the previous experiments). Unfortunately this experiment yielded negative results, since the average F-score over 10 repetitions of the experiment was of only 0.54. This result is consistent with previous experiments that show that learning a single model to classify different subjects is very challenging tasks because of the differences in the EEG signals between them [6].

6 Lessons learned

There are several important lessons to learn from this project:

- **Complexity of EEG signals.** Distinguishing between the interictal and preictal stage is a difficult task. The non-stationarity of the EEG signals and the difference between the seizure manifestation in different subjects complicate the task of creating a classifier. Nevertheless, analyzing small EEG segments, and extracting features from the frequency domain can lead to the training of classifiers that perform well above chance level.
- **Importance of dealing with the unbalance in the data.** This project displays a problem that is common among many biological datasets: high-dimensional data, with scarce instances, and low ratio of minority/majority class. As it is shown in the experiments, simply ignoring this problem might prevent the network for learning anything, no matter how

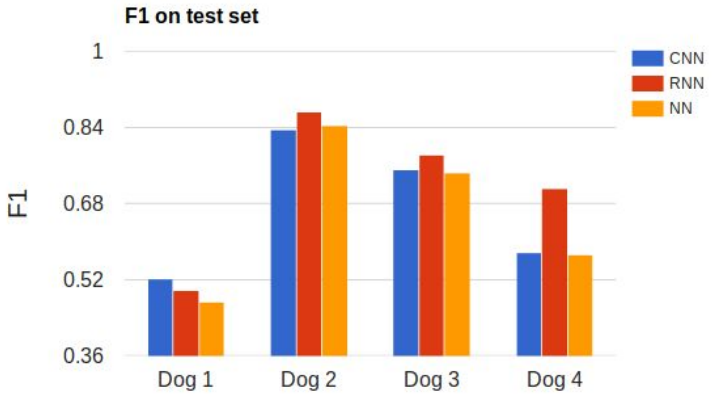

Figure 10: Classification results of the three networks on the different dogs. Note how RNN outperforms the other two approaches in 3/4 dogs.

complex the model might be. Simple technique like oversampling alleviate the problem, but do not solve it completely. It is important to realize that when learning a model with a target distribution different from the target distribution where it will be used the performance might decrease significantly.

- **Importance of having a hold-out set.** As shown by the experiments in this report, it is easy to tune a model to achieve an apparent good results; however, all that tunning might quickly lead to overfitting to that particular dataset, even if it is divided intro train, validation and test. Having a hold-out set might help to get a better estimate of the real performance of the system.
- **Importance of measuring models against a simple baseline.** Deep learning is proposing models with increased level of complexity. It is important to measure those complex models against simple baselines to check that progress is being made. In this project the CNN did not show any improvement with respect to simple feed-forward neural network, despite the increased complexity of the latter. In experiments outside the scope of this project we have seen similar behaviors, in which simple RNN can outperform LSTM for some tasks.

References

[1] Brinkmann, B. H. *et al.* Forecasting seizures using intracranial eeg measures and svm in naturally occurring canine epilepsy. *PloS one* **10**, e0133900 (2015).

[2] Mormann, F. *et al.* On the predictability of epileptic seizures. *Clinical neurophysiology* **116**, 569–587 (2005).

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

[4] Thodoroff, P., Pineau, J. & Lim, A. Learning robust features using deep learning for automatic seizure detection. *arXiv preprint arXiv:1608.00220* (2016).

[5] Patterson, E. E. Canine epilepsy: an underutilized model. *ILAR Journal* **55**, 182–186 (2014).

[6] Bashivan, P., Rish, I., Yeasin, M. & Codella, N. Learning representations from eeg with deep recurrent-convolutional neural networks. *arXiv preprint arXiv:1511.06448* (2015).

[7] Alotaiby, T. N., Alshebeili, S. A., Alshawi, T., Ahmad, I. & El-Samie, F. E. A. Eeg seizure detection and prediction algorithms: a survey. *EURASIP Journal on Advances in Signal Processing* **2014**, 183 (2014).

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

[9] Yue-Hei Ng, J. *et al.* Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4694–4702 (2015).

[10] Mesnil, G. *et al.* Unsupervised and transfer learning challenge: a deep learning approach. *ICML Unsupervised and Transfer Learning* **27**, 97–110 (2012).

[11] Raina, R., Battle, A., Lee, H., Packer, B. & Ng, A. Y. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, 759–766 (ACM, 2007).

[12] Baldi, P. Autoencoders, unsupervised learning, and deep architectures. *ICML unsupervised and transfer learning* **27**, 1 (2012).

[13] Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016). `http://www.deeplearningbook.org`.

[14] Boureau, Y.-L., Ponce, J. & LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 111–118 (2010).

[15] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105 (2012).

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

[17] Norouzi, M. *et al.* Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, 1723–1731 (2016).

[18] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

[19] Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research* **16**, 321–357 (2002).

[20] Lemaître, G., Nogueira, F. & Aridas, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* **18**, 1–5 (2017). URL `http://jmlr.org/papers/v18/16-365.html`.

[21] Storkey, A. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning* 3–28 (2009).