# Epileptic Seizure Prediction in Intracranial EEG Recordings using Deep Learning

by

**Daniel Varnai**
1839132

Supervisor
**Hamid Dehghani**

A thesis submitted to the University of Birmingham for the degree of MASTER OF ADVANCED COMPUTER SCIENCE

School of Computer Science
University of Birmingham
September 2018

**Abstract**

Epilepsy is the most common serious neurological disorder affecting almost 40 million people worldwide. About 25% of these people still experience seizures despite multiple attempts at medication. For them, responsive neurostimulation, a system similar to heart pacemakers, could mean a possible therapy by predicting and aborting seizures before they happen. We explored how we can improve a commonly used preprocessing pipeline, how we can design and optimise a deep neural network to predict seizures, how we can create an ensemble of different models to improve prediction and how our approach compares to the best algorithms in the field. We propose a novel data augmentation approach, that is simple, yet effective for all models and propose a convolutional neural network model that achieves comparable performance to the state-of-the-art models, while being still simple enough to be deployed on implanted devices.

# Contents

# INTRODUCTION

Epilepsy is the most common severe neurological disorder affecting almost 700 million people worldwide. In developed countries, prevalence was estimated to be 700 per 100,000 population, while in developing countries this number is expected to be even higher (Abramovici and Bagić, 2016). About 20-40% of these people still experience seizures despite multiple attempts at medication and even for whom anti-epileptic drugs work it can have serious side effects.

Epilepsy can severely affect the quality of the patients' lives, and it is not uncommon to experience constant anxiety or the fear of an upcoming seizure. Seizures can make everyday tasks, such as driving or swimming, very dangerous. They can potentially be fatal not just for the patients but also for others nearby.

More invasive prevention methods, such as responsive neurostimulation, a system similar to heart pacemakers, could mean a possible therapy to these patients by aborting seizures before they happen, provided we can predict them in time, which could potentially improve the quality of life of millions of people worldwide.

Research shows that from EEG recordings, it is possible to predict epileptic seizures before they happen using statistical methods, such as different machine learning algorithms, which is still an ongoing research area, with several publications proposing different algorithms, but a perfect solution is yet to be found. In this project, we would like to explore seizure prediction using deep learning, a subset of machine learning that has been proven useful in several fields of study, including medicine.

As part of this project, we would like to explore the following research questions:

1. How can we preprocess the raw EEG signals to reduce their dimensionality while keeping essential features
2. How can we design and optimise a deep neural network to predict epileptic seizures
3. Can we create an ensemble of different models with superior predictive capabilities
4. How does the proposed model compare to the state-of-the-art algorithms

## 1.1 Structure of the report

In chapter 2, we discuss the latest research in the field and put our project in context.

In chapter 3, we provide the reader with the necessary background on the algorithms and techniques used during the project, to help with the understanding of the rest of the paper.

In chapter 4, we discuss the data used, how it was collected, and the various challenges we faced using it. We explore how the data can be preprocessed to extract useful features for prediction, and we propose a new data augmentation approach.

In chapter 5, we explain how our proposed deep neural network model was designed, and how its hyper-parameters were chosen.

In chapter 6, we experiment with two more machine learning techniques, and we create an ensemble model of the three proposed models.

In chapter 7, we discuss our results and evaluate how the different approaches affected the predictive capabilities of the models. We put our findings in context and compare them to the state-of-the-art algorithms.

In chapter 8, we summarise our achievements, and we propose future work possibilities based on the deficiencies of our approach.

In chapter 9, we discuss what we achieved regarding our goals.

# LITERATURE REVIEW

Machine learning in general and especially artificial neural networks are becoming more and more widespread in several fields of study, providing state-of-the-art classification and regression capabilities that can even surpass humans in tasks such as image recognition, playing games such as Go (Silver et al., 2017), and even in designing algorithms for such tasks (Real et al., 2018).

New deep learning algorithms are now also making medical diagnosis easier for medical experts, decreasing wait times, and bringing us closer to a future of wearable and implanted medical devices. They have already been proven clinically applicable in several areas, such as diagnosing retinal diseases with 94.5% accuracy (De Fauw et al., 2018) or detecting brain tumours on MRI images with 97% accuracy (Mohsen et al., 2018) and have shown great promise in several other fields.

One such area is epileptic seizure prediction that has been actively researched for several decades now and has been shown to be possible using intracranial EEG recordings. Numerous datasets exist containing brain activity between seizures and leading up to seizures; however, prediction algorithms show significantly different performance depending on the patients and the datasets used. In this section, we are going to discuss the state-of-the-art prediction algorithms for popular datasets.

On the dataset collected by the University Hospital of Freiburg, convolutional neural networks were already used successfully as far back as 2008 to provide perfect classification without false alarms for 20 out of the 21 patients and very high accuracy for the remaining patient as well, yet it only contained 88 seizures. Even more straightforward approaches, such as logistic regression and support vector machines proved to provide high accuracy predictions for all patients in this study (Mirowski et al., 2008). However, these EEG recordings are short-term, and due to the inpatient monitoring they are suspected to be affected by antiepileptic drug dose tapering, that can alter the normal brain activity, and therefore statistical analysis of such data might not be reliable (Kuhlmann et al., 2018). This dataset has now been superceded by the European Epilepsy Database and is not publicly available anymore.

The first long-term human study with an implanted ambulatory device was conducted in 2010, in which 15 patients were included, however, due to adverse events the equipment had to be removed from some of them. This dataset is now referred to as the NeuroVista trial. In this research, an undisclosed machine learning algorithm was trained for each patient with

varying success; in some cases, it had perfect predictive capabilties without false alarms, while for other patients it did not even meet the performance criteria to be disclosed as part of the study (Cook et al., 2013). Only parts of this dataset are publicly available; however, as part of the on-going seizure prediction competition, the best performing teams are annually invited to benchmark their algorithms on the full dataset. A research was done with neuromorphic chips on the patients that passed the criteria that found that neural networks can potentially perform even better than the undisclosed algorithm (Kiral-Kornek et al., 2018).

A similar study was conducted on dogs, in which high accuracy classification was achieved for all patients using logistic regression based on the spectral analysis of the EEG recordings (Howbert et al., 2014). Parts of this dataset are now available publicly, and a competition was held on Kaggle in 2014. The best performing algorithm achieved an AUCROC (section 3.16.2) score of 0.83993, where 1.00 means perfect classification. The details of the winning solution were undisclosed, but it was achieved using a least squares support vector machine. The second-ranking team used an ensemble of a regularised logistic regression, bagged support vector machines and a random forest. The third place was achieved by a support vector machine as well. Ensembles including convolution neural networks also managed to get amongst the top ten (Brinkmann et al., 2016).

In 2016, a Kaggle competition was held to find an algorithm that is capable of predicting seizures for the three patients in the NeuroVista trial who did not even meet the performance criteria. The winning solution in this competition achieved an AUCROC score of 0.80701 and used an ensemble of several algorithms: an extreme gradient boosting classifier, k-nearest neighbours, a generalised linear model and a linear support vector machine. Amongst the top teams who discussed their algorithms, ensembles consisting of SVM models and decision tree-based algorithms were highly popular. The only disclosed algorithm that used a convolutional neural network as part of their model achieved the 45th place; however, it proved to outperform the best algorithms on the full NeuroVista dataset achieving an overall AUCROC score of 0.76632, making it the only algorithm that improved with the additional data (Kuhlmann et al., 2018). The second team in their report discussed some experiments with neural networks, but they abandoned the model due to poor performance that they suspected was caused by the limited amount of data available (Lang et al., 2016).

In August 2018, the dataset from the latest Kaggle competition is available again, and a new competition has started with slightly different goals. In the earlier competition, the algorithms had no restrictions on the computational resources available; however, as any such algorithm would be deployed

on an implanted device with limited resources, most top algorithms would not be usable in practice (Kuhlmann et al., 2018).

As we can see, the state-of-the-art algorithms are still not good enough, and neural networks generally perform poorly, yet they proved themselves in several research areas.

# BACKGROUND

This section aims to provide a high-level overview of the algorithms used in this project to help with the understanding of the upcoming sections.

## 3.1 Machine Learning

A machine learning algorithm is a computer program that can learn according to the following common definition of learning:

"A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$." (Mitchell, 1997)

The main categories for learning algorithms are:

- Supervised Learning: the algorithm is provided with the input and the corresponding output and should model the relationship between them.
- Unsupervised Learning: the algorithm is only provided with the input and should find patterns in it.
- Reinforcement Learning: an algorithm should learn based on indirect feedback on his actions such as a reward or a penalty.

## 3.2 Overfitting

We talk about overfitting when a statistical model has a large gap between its training error and test error, causing it to generalise poorly with previously unseen data (Goodfellow et al., 2016). Overfitting can affect many machine learning tasks, but several methods exist to prevent it, such as L1 (section 3.3.1) and L2 (section 3.3.2) regularisation or drop-out layers (section 3.12).

## 3.3 Regularisation

Regularisation is any method that aims to reduce the generalisation error of a model without affecting training error (Goodfellow et al., 2016). Common regularisation techniques include L1 and L2 norm, which prevent the weights from becoming too high.

### 3.3.1 L1 norm

L1 regularisation is done by adding a penalty term to the loss function in the form of the sum of absolute weights:

$$\Omega = \lambda \sum_i |w_i| \tag{1}$$

where $\lambda$ is the penalty factor and $w$ is weight vector.

### 3.3.2 L2 norm

L2 regularisation, or weight decay as commonly called in neural networks, is done by adding a penalty term to the loss function in the form of the sum of squared weights:

$$\Omega = \lambda \sum_i w_i^2 \tag{2}$$

where $\lambda$ is the penalty factor and $w$ is weight vector.

## 3.4 Artificial Neural Networks

Artificial neural networks are machine learning algorithms inspired by the biology of the brain. They consist of interconnected nodes (neurons) of functions that can approximate other functions. The model-parameters, such as the weights and biases, are commonly found using gradient-based optimisation algorithms, but evolutionary algorithms could also be utilised given enough resources (Koehn, 1994).

## 3.5 Backpropagation

Backpropagation is a technique used in neural networks to compute the gradients for gradient-based optimisation algorithms, such as gradient descent or Adam. Using backpropagation, we can compute how changes to the weights and biases change a differentiable loss function by moving the error backwards in the network using the chain rule. It can be summarised with the following equations (Nielsen, 2018).

Weighted input:

$$z_k^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \tag{3}$$

The error in the output layer:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L) \tag{4}$$

The error in layer $l$ in terms of the error in layer $l+1$:

$$\delta_k^l = (\sum_m \delta_m^{l+1} w_{mk}^{l+1}) \sigma'(z_k^l) \tag{5}$$

The derivative of the cost function with respect to any bias in the network:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{6}$$

The derivative of the cost function with respect to any weight in the network:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{7}$$

where $C$ is the cost function, $a$ is the activation, $z$ is the weighted input and $\sigma$ is the activation function (section 3.6) used.

## 3.6 Activation Functions

Activation functions are used to introduce non-linearity to an otherwise linear network, and thus making the network capable of approximating non-linear functions. In this section, we discuss some common activation functions and their application.

### 3.6.1 Sigmoid

The sigmoid or inverse logistic function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

It is commonly used for binary classification problems. Due to its squashing property, it suffers from the vanishing gradients problem (section 3.7).

### 3.6.2 Hyperbolic Tangent

The *tanh* activation function is a sigmoid (section 3.6.1) function scaled to the $(-1, 1)$ interval. It can be beneficial to use *tanh* over sigmoid in some cases, as it can converge faster (LeCun et al., 2012). It also suffers from the vanishing gradients problem (section 3.7). It can typically be found in LSTM units (section 3.13.1).

### 3.6.3 Hard Sigmoid

Hard sigmoid is a linear approximation of the sigmoid function. In different frameworks, it might be defined differently. We use Keras with TensorFlow backend, in which case it is defined as (Keras, 2018):

$$f(x) = max(0, min(1, 0.2 * x + 0.5)) \tag{9}$$

### 3.6.4 Rectified Linear Unit

Rectified Linear Unit (ReLU) is a simple, yet efficient, activation function defined as:

$$f(x) = max(0, x) \tag{10}$$

A benefit of ReLU is its sparsity, the ability to return precisely zero values. Research suggests that ReLU provides as good as or even better performance than sigmoid (section 3.6.1) or tanh (section 3.6.2) (Glorot et al., 2011). It can suffer from the dead ReLU problem, where the output is always zero. This can be prevented by decreasing the learning rate.

## 3.7 Vanishing Gradients

Vanishing gradients are an issue with squashing activation functions such as the sigmoid when gradient-based learning algorithms are used. Due to this squashing property, near the boundaries of the function, even substantial changes to the input would only cause a small difference in the output. This can slow down or prevent the network from learning as the gradients become very small in the lower layers of the network. We can avoid this issue by using different activation functions, such as the Rectified Linear Unit (section 3.6.4).

## 3.8 Exploding Gradients

We talk about exploding gradients when the weights become too large and cause massive changes in the output, preventing the network from learning. This can be prevented by using batch normalisation (section 3.10) or lower learning rate.

## 3.9 Mini-Batch Learning

Mini-batch learning means that the gradients are only computed for a subset of the samples at a time, and thus reducing the resources needed to train the network. It was also found that using mini-batches can even improve the generalisation performance compared to batch learning, where all the samples are used for gradient computation. Research suggests that batch sizes between 2 and 32 provide the best generalisation performanc; however, the smaller the batches are, the more time that is required for training (Masters and Luschi, 2018).

## 3.10 Batch Normalisation

Batch normalisation is a technique proposed by Google engineers to speed up the training while providing the same network performance (Ioffe and Szegedy, 2015). They found that the change in the distributions of the internal nodes during the training was an issue that slowed down the training in most cases, similarly to data that is not normalised before training. Their proposed method eliminates these distribution changes by standardising the mini-batches, thus requiring less training iterations. They also found that their method provides regularisation as well, and can potentially eliminate the need for drop-out layers.

## 3.11 Gradient descent

Gradient descent is a gradient-based optimisation algorithm for finding the minimum of a function. It works by taking steps towards the gradient of a function as computed by backpropagation.

### 3.11.1 Adam

Adam is a state-of-the-art gradient-based optimiser. It is well suited for large amounts of data and parameters. All of its hyper-parameters have intuitive meaning and generally don't require tuning, while providing comparable performance to other optimisers, making it an attractive choice for any problem. It works by computing adaptive learning rates from the first and second moments of the gradients, combining the advantages of other popular optimisers such as RMSProp (Tieleman and Hinton, 2012) and AdaGrad (C. Duchi et al., 2011). The pseudocode of Adam can be seen in Algorithm 1 (Kingma and Ba, 2014).

## 3.12 Drop-out

Drop-out was introduced as a simple, yet very effective, way to prevent overfitting in networks by randomly dropping nodes and their connections during training. During testing, the predictions of these thinned networks are simply averaged. This also proved to increase the performance of several learning tasks, leading to state-of-the-art results in several fields (Srivastava et al., 2014).

**Input:** $\alpha$: Step size
**Input:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Input:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Input:** $\theta_0$: Initial parameter vector
$m_0 \longleftarrow 0$ (Initialize first moment vector)
$v_0 \longleftarrow 0$ (Initialize second moment vector)
$t_0 \longleftarrow 0$ (Initialize time step)
**while** $\theta_t$ *not converged* **do**
    $t \longleftarrow t + 1$
    (Get gradients w.r.t stochastic objective at time step t)
    $g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$
    (Update biased first moment estimate)
    $m_t \longleftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    (Update biased second raw moment estimate)
    $v_t \longleftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    (Compute bias-corrected first moment estimate)
    $\hat{m}_t \longleftarrow m_t / (1 - \beta_1^t)$
    (Compute bias-corrected second raw moment estimate)
    $\hat{v}_t \longleftarrow m_t / (1 - \beta_2^t)$
    (Compute bias-corrected first moment estimate)
    $\theta_t \longleftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
**end**
**return** $\theta_t$ *(Resulting parameters)*

**Algorithm 1:** Adam pseudocode

## 3.13 Recurrent Neural Networks

Recurrent neural networks are a type of artificial neural networks that are designed to work on sequences of input using a feedback loop to provide a form of memory. Most widely used recurrent units are the Gated Recurrent Unit (GRU) and the Long Short-Term Memory (LSTM) (section 3.13.1).

### 3.13.1 Long Short-Term Memory

Long Short-Term Memory was initially proposed in (Hochreiter and Schmidhuber, 1997) to solve the main issues with the then existing recurrent units: vanishing (section 3.7) and exploding gradients (section 3.8) and poor efficiency but it suffered from the inability to reset itself. A modified LSTM unit was proposed with a forget gate in (Gers et al., 2000) that solved the issue as mentioned earlier. Furthermore, the previous papers initialised bias to zero, but research showed positive bias could improve network performance (Jozefowicz et al., 2015). In this work, the latter, extended version is used with bias as implemented in Keras.

An LSTM unit has three gates: an input, output and forget gate, and a memory unit called the cell. These allow an LSTM unit to remember, forget and control how much of the memory and the new input should be used to compute the output. The forward-pass of an LSTM unit can be summarised using the following formulas (Gers et al., 2000) (Jozefowicz et al., 2015):

Input gate activation:

$$a^{in_j}(t) = \sigma_{in_j}(z_{in_j}(t)) \tag{11}$$

Output gate activation:

$$a^{out_j}(t) = \sigma_{out_j}(z_{out_j}(t)) \tag{12}$$

Forget gate activation:

$$a^{\varphi_j}(t) = \sigma_{\varphi_j}(z_{\varphi_j}(t)) \tag{13}$$

Cell internal state:

$$s_{c_j^v}(0) = 0 \tag{14}$$

$$s_{c_j^v}(t) = a^{\varphi_j}(t) s_{c_j^v}(t-1) + a^{in_j}(t) tanh(z_{c_j^v}(t)) \tag{15}$$

Cell output:

$$a^{c_j^v} = a^{out_j}(t) tanh(s_{c_j^v}(t) \tag{16}$$

where $\sigma$ is the hard sigmoid activation function (section 3.6.3) and $z$ is the weighed input (section 3.5).

Training of such units is done with a modified version of backpropagation (section 3.5) called Backpropagation Through Time (BPTT), which unrolls the recurrent units for every time step, computes the gradients and rolls the network back.

## 3.14 Convolutional Neural Networks

Convolutional neural networks are a type of artificial neural networks designed for processing data with a grid-like structure (Goodfellow et al., 2016). Such networks have been successfully used in several fields, such as computer vision, pattern recognition and so on. As the name suggests, they utilise the convolution operation. Such networks commonly consist of convolutional layers, pooling layers (section 3.15) and non-linear activation layers such as ReLU (section 3.6.4).

### 3.14.1 Convolutional Layer

Convolutional layers are the core building blocks of a convolutional network. They consist of a kernel that they convolve with their input:

$$s(t) = \int x(a)w(t-a)da \tag{17}$$

where $x$ is the layer input, $w$ is the kernel and $t$ is the amount of shift used. The output $s$ is commonly called the feature map.

Convolution has three properties that can improve the performance of a network: sparse connections, parameter sharing and equivariance to translation. As the kernel is generally significantly smaller than the input, convolutional layers are sparsely connected thus improving memory usage and efficiency. As the kernel is applied to every part of the input, parameters are adapted to work with the full input rather than just parts of it, further improving memory usage. As they are equivariant to translation, the following is true:

$$g(s(x)) = s(g(x)) \tag{18}$$

where $g$ is a transformation (Goodfellow et al., 2016).

## 3.15 Pooling Layers

Pooling layers provide a summary static of the nearby outputs of the previous layer (Goodfellow et al., 2016). They are most often found after convolutional layers. Common pooling layers include max pooling and average pooling.

### 3.15.1 Max pooling

Max pooling units output the maximum value found within their kernel. It is often used to provide approximate translation invariance in convolutional networks (section 3.14) and further dimensionality reduction. Intuitively, it can be interpreted as selecting the most prominent feature of the feature map.

## 3.16 Metrics

### 3.16.1 Receiver Operating Characteristic Curve

The receiver operating characteristic curve describes the classification capabilities of a predictive model in terms of its false positive rate and true positive rate at various thresholds. ROC analysis is a useful tool to determine the appropriate classification threshold for hard classification.

### 3.16.2 Area Under the ROC Curve

Area Under the Receiver Operating Characteristics Curve (AUCROC) is a useful metric for ranking binary classifiers with imbalanced classes. It can be interpreted as the probability of ranking a positive sample higher than a negative one. AUCROC is not differentiable, therefore cannot be used as a loss function in gradient-based learning algorithms.

### 3.16.3 Cross-Entropy Loss

Cross-entropy quantifies the difference between two distributions $P$ and $Q$ over the same event set $A$:

$$H^C(P,Q) = \sum_{x \in A} P(x) \frac{1}{Q(x)} \tag{19}$$

We can quantify the error caused by misclassification by using this as a loss function (de Boer et al., 2005). It is a common choice when the goal is to maximise AUCROC.

## 3.17 Ensemble Learning

Ensemble learning is a paradigm for combining models that use different algorithms (heterogeneous), hyper-parameters or input vectors (homogeneous). Intuitively, the various models should have different errors. By combining them, we can average out the error and get a model that is better than any individual model. It can be very tempting to use ensembles if we have many weak learners that are only marginally better than random choice. Developing many weak learners are considered easier than developing a single very good algorithm, so we can potentially save a significant amount of time by constructing ensembles at the cost of added complexity (Zhou, 2012). There are three common types of ensemble learning techniques we are going to discuss briefly in the following.

### 3.17.1 Bagging

Bagging or bootstrap aggregating is a method that combines weak learners trained on different bootstrap samples (samples with replacement). Most famous bagging algorithm is the random forest (section 3.18) (Zhou, 2012).

### 3.17.2 Boosting

Boosting combines weak learners trained on the same samples. Base learners are trained by adjusting the class weights based on the number of misclassifications (Zhou, 2012). Commonly used boosting methods include AdaBoost (Freund and Schapire, 1997) and Gradient Boosting (Friedman, 2001).

### 3.17.3 Stacking

Stacking combines different models using a meta-model. These models are frequently heterogeneous (Zhou, 2012).

## 3.18 Random Forest

Random forest is a homogeneous ensemble algorithm proposed in (Ho, 1995) that uses bagging with several randomly generated decision trees to achieve a model with better predicting capabilities for both classification and regression problems. Due to its design, it is robust against overfitting, it only has a few hyper-parameters and can be trained very quickly, making it an attractive option.

Trees are generated by first randomly selecting $M$ features out of the feature space with size $N$ where $M < N$. Then, splits are made to separate at least two classes at each split while also minimising the error of the split.

Once the trees are generated, classification is done using a discriminant function that combines the trees into a forest. The posterior probability that data point $x$ belongs to class $c$ in tree $j$ if the terminal node is $v_j(x)$ is defined as:

$$P(c|v_j(x)) = \frac{P(c, v_j(x))}{\sum_{l=1}^{n} P(c_l, v_j(x))} \tag{20}$$

This value is almost always 1 in case of fully split trees except for abnormal stops. Finally, using a discriminant function, we can calculate the probability of each class in a forest with size $t$ based on the posterior probabilities:

$$g_c(x) = \frac{1}{t} \sum_{j=1}^{t} P(c|v_j(x)) \tag{21}$$

In fully split trees without abnormal stops, this means the probability of a class in a random forest equals to the number of times a class is predicted divided by the number of trees.

## 3.19 Logistic Regression

Logistic regression is a statistical model in (Cox, 1958) that can be used for binary classification problems. It models posterior class probability that data point $x$ belongs to class $c$ using the sigmoid function (section 3.6.1):

$$P(c = \pm 1|x) = \frac{1}{1 + e^{-cw^T x}} \tag{22}$$

where $w \in R^n$ is the weight vector. Given training data $\{x_i, c_i\}_{i=1}^{l}, x_i \in R^n, c_i \in \{-1, 1\}$, logistic regression minimises the following log-likelihood:

$$P(w) = C \sum_{i=1}^{l} log(1 + e^{-c_i w^T x_i}) + \frac{1}{2} w^T w \tag{23}$$

where $C > 0$ is the penalty factor for regularisation that helps to prevent overfitting by penalising model complexity. Advantages of this method include the ability to describe the relationship between a feature and a class, robustness against overfitting, low number of hyper-parameters and quick training times. It can even be extended to multi-class classification problems by using a one-vs-all approach. However, it assumes linearly separable data (Yu et al., 2011).

## 3.20 Cross-Validation

Cross-validation is a technique commonly used with small datasets to prevent withholding data from training by splitting the data into several partitions and training the model repeatedly on a subset of the partitions while using the remaining partitions for validation.

# DATA COLLECTION AND PREPROCESSING

In this section, we discuss how the data was acquired, what challenges we faced using it, how it was preprocessed, and how we augmented it.

## 4.1 Dataset

The dataset includes data from two separate seizure prediction challenges.

The first dataset was published by the National Institutes of Health, the Epilepsy Foundation, and the American Epilepsy Society as part of a seizure prediction challenge on Kaggle, a platform for competitions in data science and predictive modelling. It consists of intracranial EEG recordings from 7 patients with naturally occurring epilepsy: five dogs and two human patients.

The second dataset was initially published by the University of Melbourne as part of another Kaggle challenge. However, it is only available from https://www.epilepsyecosystem.org now (Kuhlmann et al., 2018). This dataset was collected the same way as the first one, and it includes another three human patients.

## 4.2 Data Collection

The data was collected using an invasive ambulatory monitoring system, that was surgically placed inside the skull of each patient, hence the name intracranial EEG. During the surgery, electrodes are placed at locations chosen by medical professionals. The number and placement of these electrodes can differ from patient to patient, based on which parts of the patients' brain are suspected to be involved in the seizures. In Figure 1, we can see the recording system used to record the data for the dogs.

Once the electrodes were in place, EEG signals were recorded over a long period of .time, in some cases up to a year. During this period, several seizures were recorded for each patient. The dataset consists of one-hour sequences of ten-minute data segments, that were manually classified as either interictal ("between seizures") or preictal ("before seizure"). Interictal segments contain regular brain activity between seizures, while preictal ones contain brain activity that led up to a lead seizure with a five-minute seizure horizon. A lead seizure is defined as a seizure that happened at least four hours apart from another one. The reason for that is that seizures are known to cluster and predicting the first one is more useful than predicting follow-up seizures as they are expected to happen.

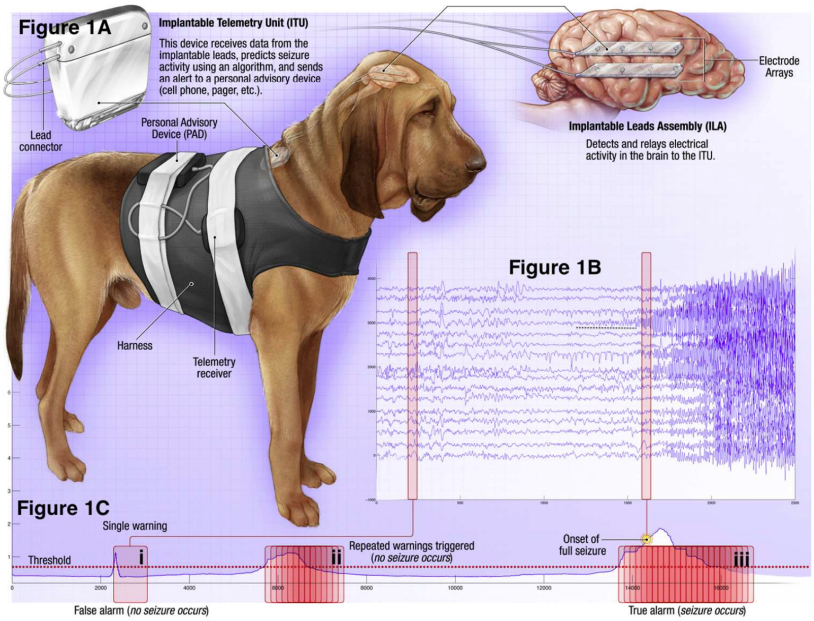



Figure 1: Recording system used for dogs (Howbert et al., 2014)

Also, any part of the segments can contain "data drop-out", where the implanted device failed to record data. Such drop-outs would cause zero signal values across all EEG channels. Furthermore, artefacts such as high amplitude rapid signals might also be present.

The number of channels, the sampling rate and the number of segments for each patient can be seen in Table 1. As we can see, we have a varying number of channels and also different sampling rates. Furthermore, the dataset is highly imbalanced, we have significantly less preictal segments than interictal ones, with over 90% of the segments being interictal.

## 4.3 Preprocessing

Data preprocessing is an essential first step to prepare the data for machine learning algorithms. By reducing the size and complexity of our data, we can significantly decrease the model complexity, the time it takes to train the model while we can also improve the performance by removing noise or information unrelated to our problem that might affect the results. We can see an unmodified sample in Figure 2.

Our preprocessing pipeline is an improved version of the pipeline that was used in a research done on dogs (Howbert et al., 2014):

1. Apply a Butterworth bandpass filter on each channel with a lower cutoff

| Patient | Channels | Sampling rate | Interictal | Preictal | Total |
|---|---|---|---|---|---|
| Dog 1 | 16 | 400 Hz | 958 | 48 | 1006 |
| Dog 2 | 16 | 400 Hz | 1410 | 132 | 1542 |
| Dog 3 | 16 | 400 Hz | 2305 | 114 | 2419 |
| Dog 4 | 16 | 400 Hz | 1737 | 154 | 1819 |
| Dog 5 | 15 | 400 Hz | 629 | 42 | 671 |
| Patient 1 | 15 | 5000 Hz | 233 | 30 | 263 |
| Patient 2 | 24 | 5000 Hz | 178 | 32 | 210 |
| Patient 3 | 16 | 400 Hz | 619 | 279 | 898 |
| Patient 4 | 16 | 400 Hz | 2123 | 240 | 2363 |
| Patient 5 | 16 | 400 Hz | 2097 | 273 | 5631 |
| Total | | | 12279 | 1344 | 13623 |

Table 1: Details of the recordings for each patient

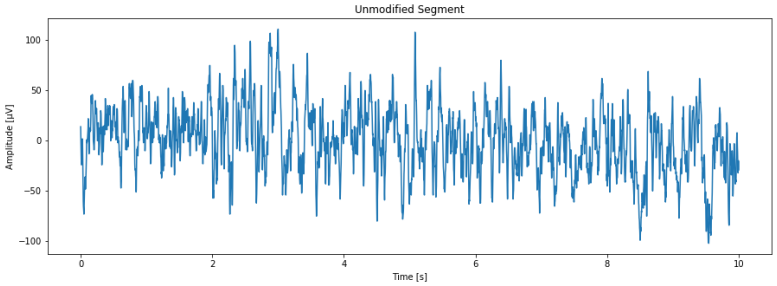



Figure 2: Original ten-second segment

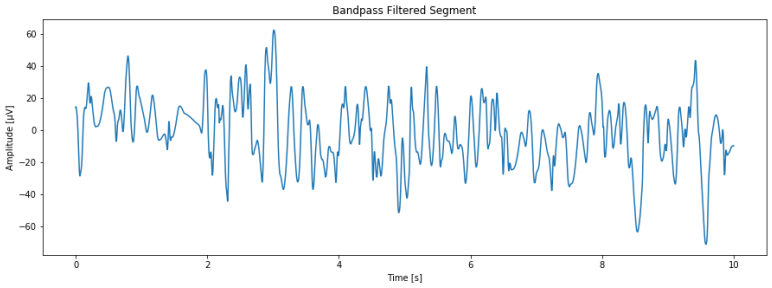



Figure 3: Bandpass filtered segment

frequency of 0.5 Hz and a higher cutoff frequency of 180 Hz. This removes high-frequency noise and frequencies not typically associated with brain activity while leaving most of the data intact. While 180 Hz might seem high for brain activity, research proves that even higher frequencies are useful for epilepsy (Zijlmans et al., 2012). We can see an exaggerated example of this in Figure 3

2. Apply discrete Fourier transformation on the ten-minute clips with a ten-second sliding window and a 50% overlap ratio:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi i n k/N} \tag{24}$$

where $n = 0, ..., N-1$, $f_k$ is the input data, and $N$ is the number of samples. This converts the signal from the time-domain to the frequency-domain for each ten-second segment. This makes analysing brain activity easier by allowing us to look at the data in terms of brain waves. By introducing overlap, the performance can be further improved by making it easier to detect epileptic signs at the end of the windows. We can see how the segment looks like in the frequency-domain in Figure 4.

3. For each segment, split the data into 8 frequency bands that are mostly associated with different brain waves: delta (0-4 Hz), theta (4-8 Hz), alpha (8-12 Hz), beta (12-30 Hz), low-gamma 1 (30-50 Hz), low-gamma 2 (50-70 Hz), high-gamma 1 (70-100 Hz) and high-gamma 2 (100-180 Hz) (Park et al., 2011). For each segment, a power-in-band feature is computed by summing the power of band frequencies:

$$PIB_{band} = max(0, \sum_{n \in band} log_{10}(|F_n|)) \tag{25}$$

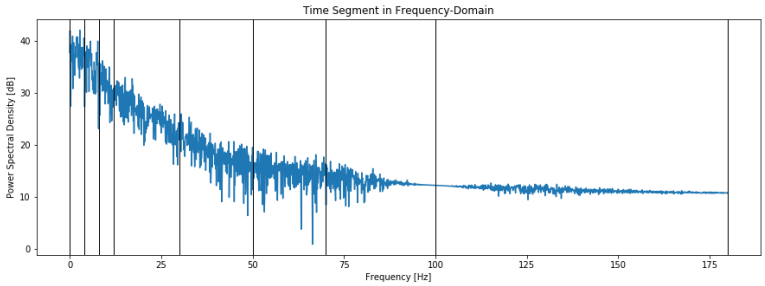



Figure 4: Segment in frequency-domain

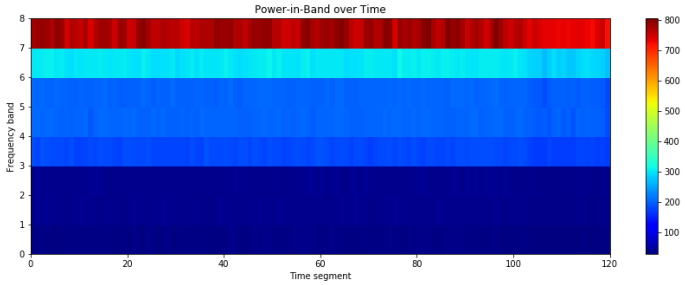



Figure 5: Power-in-Band over Time

where $X$ is the Fourier component for the given frequency. Due to the high-resolution sliding window we used, the earlier mentioned data drop-outs can lead to large negative numbers, that can potentially skew the standardisation. $PIB$ values below zero are treated as zeros to prevent this. We can see the power values of the bands over time in Figure 5.

4. The data is zero-padded to 24 channels to get the feature vectors to the same size.

5. Finally, the dataset is standardised to zero mean and unit variance. This helps the network learn faster by getting all the features on the same scale. For each frequency band $i$, we calculate the mean $m_i$ and standard deviation $\sigma_i$ across the training set, and then we standardise it:

$$StdPIB_i = \frac{PIB_i - m_i}{\sigma_i} \tag{26}$$

It is important to note that standardisation is applied to the whole

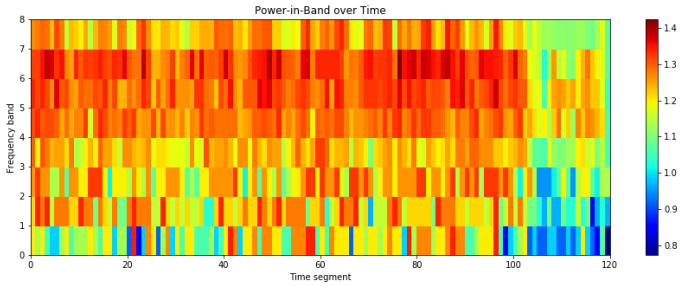



Figure 6: Standardised Power-in-Band over Time

dataset, including the validation and test set, but the means and standard deviations are computed from the training set only. This is because that the validation and test set are supposed to be unbiased estimators. For anything training related, these sets contain previously unseen data points. We can see our final data in Figure 6.

Advantage of this pipeline is the significant dimensionality reduction; the size of the data is reduced from 220 GB to just 1.3 GB. Also, the Fourier transformation step is unaffected by the different sampling rates; therefore resampling of the data is unnecessary. How the different changes increased the prediction compared to the original pipeline can be found in section 7.1.

## 4.4 Variable Number of Channels

As can be seen in Table 1, different patients have a different number of electrodes implanted. This means depending on the preprocessing pipeline used, we have a different number of features for patients, but most neural networks take a fixed size feature vector. To deal with this, we have a few choices:

- Zero-padding the input to 24 channels, filling missing features with zeros. The advantage of this method is its simplicity. However, zero values might be present in the data naturally as well.
- Recursive neural networks, which were designed to be used with variable size input, such as trees. (Goller and Kuchler, 1996)
- Recurrent neural networks have internal memory that is typically used to deal with arbitrary size time-distributed data.

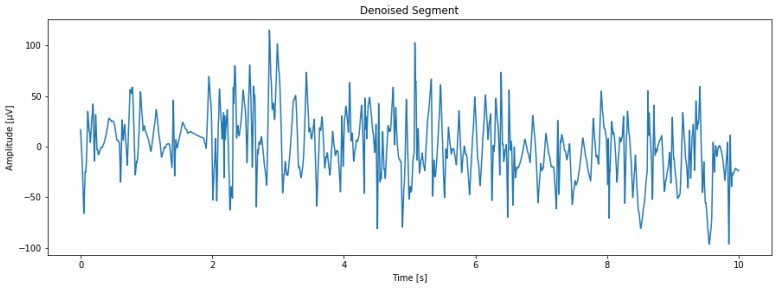



Figure 7: Denoised ten-second signal segment

Zero-padding was chosen due to its simplicity. As we calculate the power for a small number of frequency bands from hundreds of frequency components, the power-in-band should virtually never be zero naturally. Zero values should only happen in the case of data drop-outs, in which case the network would treat them the same as zero-padded features. Using a recursive neural network would significantly increase the complexity of the model, and recurrent neural networks are more appropriate when we are dealing with a variable number of time steps.

## 4.5 Signal Denoising

Since we are dealing with raw EEG signals, noise is expected. A standard approach to denoising EEG signals is wavelet decomposition (Mamun et al., 2013). A signal is decomposed into multi-resolution parts based on a basic function. In case of EEG, the functions from the Daubechies family are generally used as the basic function. After decomposition, a noise threshold is determined, and the signal is reconstructed with respect to the noise threshold.

We experimented with denoising the data using a 'db4' wavelet with four levels of decomposition with VisuShrink or universal thresholding. As can be seen on an exaggerated example in Figure 7, the signal is smoother, but we might also have removed critical epileptic features.

Then, the previously discussed preprocessing pipeline was applied to the denoised signals. In most cases, this made small or no difference in Power-in-Band values compared to the noisy data and provided no performance increase for the algorithms, and thus denoising the data was deemed unnecessary.

## 4.6 Data Augmentation

Data augmentation is deriving new data points from the ones we have, thus increasing the amount of data available to us. There's previous research on successfully augmenting EEG data by rotating it around its axis (Krell and Kim, 2017), and teams in the competitions also created new data points by reconstructing the sequences and creating new ten-minute segments (Korshunova, 2015).

We experimented with two simple approaches to augment the data. First, we shuffled the EEG channels in the recordings. Using this approach, we hoped to prevent the network from learning non-existent spatial correlations in the data caused by the different number of channels and unknown electrode placements. Furthermore, different patients experience seizures in different brain regions. We hoped to improve generalisation across patients while also preventing overfitting by shuffling the channel, which could potentially enhance the performance with unseen patients who experience different seizures from the patients we used for the training.

The other approach is shuffling the time steps in the recordings. This approach would eliminate the time distribution of the data, which could prove useful in models without internal memory. This might have the disadvantage of not detecting prolonged seizure signs, and it might only provide a useful prediction at the end of the ten-minute segments.

Keras already supports data augmentation by providing a way to train network using data generators. A generator was implemented that creates batches of data for training by taking random data points with replacement and shuffling the appropriate dimensions of the recordings. In each epoch, new random batches of data are generated and used for training.

This approach was also found to be useful for prediction. By predicting the probability for several samples then calculating the mean, performance can be improved.

# DESIGNING THE NETWORK

In this chapter, we are going to discuss how the different layers and hyper-parameters of the network were chosen.

## 5.1 Layer Types

First, the different types of layers were determined based on the goals we want to achieve.

The feature vector is first fed into a number of time-distributed convolutional layers (section 3.14.1), which provide further feature extraction, providing more useful information for the upcoming layers of the network. By making the layers time-distributed, each time step has its layers, and the output would still be time-distributed, which is useful for the recurrent layers. Each of these layers is followed by a ReLU (section 3.6.4) activation layer to provide non-linearity in the network. As convolution is only equivariant to translation, max-pooling layers (section 3.15.1) were also introduced to provide approximate translation invariance, thus helping to find the features even if they are somewhere else in the feature vector.

Recurrent layers were also considered as they were designed to be used with time-series. However, convolution neural networks have also been used successfully with a part of our dataset (Korshunova, 2015) and with time series data in general (Zhao et al., 2017). Two most common unit options are LSTM (Long-Short Term Memory) (section 3.13.1) and GRU (Gated Recurrent Unit) layers. GRU is more computationally efficient, making the training faster while providing comparable performance to LSTM in many tasks (Chung et al., 2014). LSTM was chosen as the training time was not an issue. For further classification, all intermediate values of these units are returned. With these layers, we can look at a time series and remember relevant information that can be used to predict a seizure.

Since this is a classification problem, we also need dense layers at the end of our network. Dense layers are generally the best for classification problems; however, fully convolutional classifier networks do exist, such as the YOLO real-time object detection network (Redmon and Farhadi, 2016). These layers were also followed by ReLU activation layers as well. Dropout layers (section 3.12) were also used to prevent overfitting and improve generalisation; however, this might be redundant with the use of batch normalisation.

The output of our network should be the probability of an upcoming seizure based on the input feature vector. A single fully connected unit

with sigmoid activation function was chosen as the output of the network to achieve this as it is differentiable, non-linear and its shape makes it an appropriate choice for classification problems.

Later on, batch normalisation (section 3.10) layers were also introduced before non-linear activations, as they proved to significantly speed up the training time while providing the same performance. These layers did not perform well with the default parameters in Keras, the loss function was oscillating, and the momentum had to be decreased to prevent that.

## 5.2 Number of Layers and Units

The number and size of each layer were chosen through experiments. Initially, one of each layer type was used with a reasonable amount of units, generally a number between the input and the output size of the layer. Then, for each layer, the number of units was incrementally increased until the performance increase was insignificant. Model complexity was prioritised over negligible performance gains as any such algorithm would be deployed on a small, implanted device with limited resources. Once the unit sizes were chosen for a layer, more layers were added, and the process was restarted.

Due to the limited time, increments were chosen to get a good performing model in a few trainings. Given enough time, an evolutionary algorithm could have been deployed to find a better model (Koehn, 1994). Such an approach has been successfully used in the past and performed better than any hand-picked model, but it required a significant amount of resources that are not available to us (Real et al., 2018).

To further optimise the model for embedded devices and speed up finding a better model, a technique called pruning (Manessi et al., 2017) could be utilised. With pruning, we could pick larger layers, train the network and let the pruning algorithm find the weights that don't contribute to the final output.

The final design of the network with recurrent layers can be seen in Figure 8, and without recurrent layers in Figure 9.

## 5.3 Training parameters

We had to decide what loss function to use. The metric we would like to maximise is the AUCROC (section 3.16.2); however, this is not a differentiable function, which is essential for gradient descent. We needed a proxy objective, that correlates well with our original goal. For such cases, binary cross entropy (section 3.16.3) is a common choice. As Keras already provided this as a loss function, this was chosen for now. However, there's research on

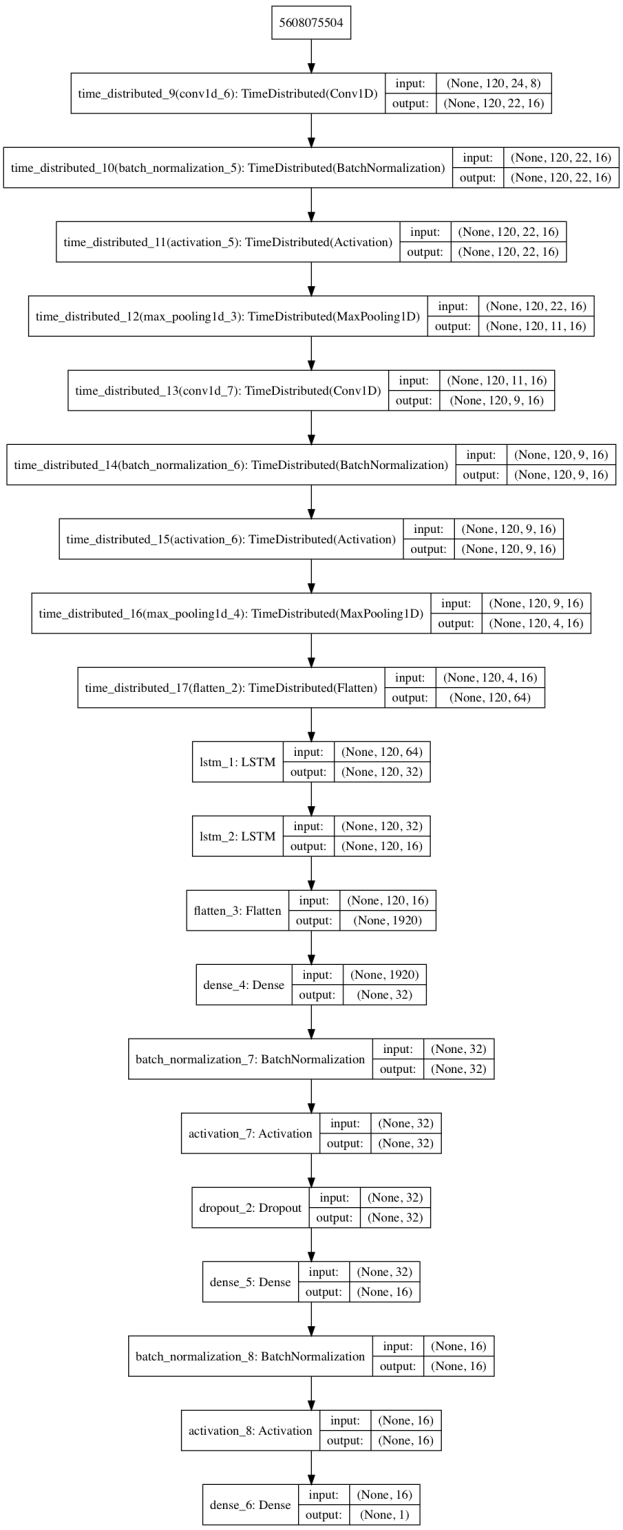



Figure 8: Final convolutional recurrent neural network design

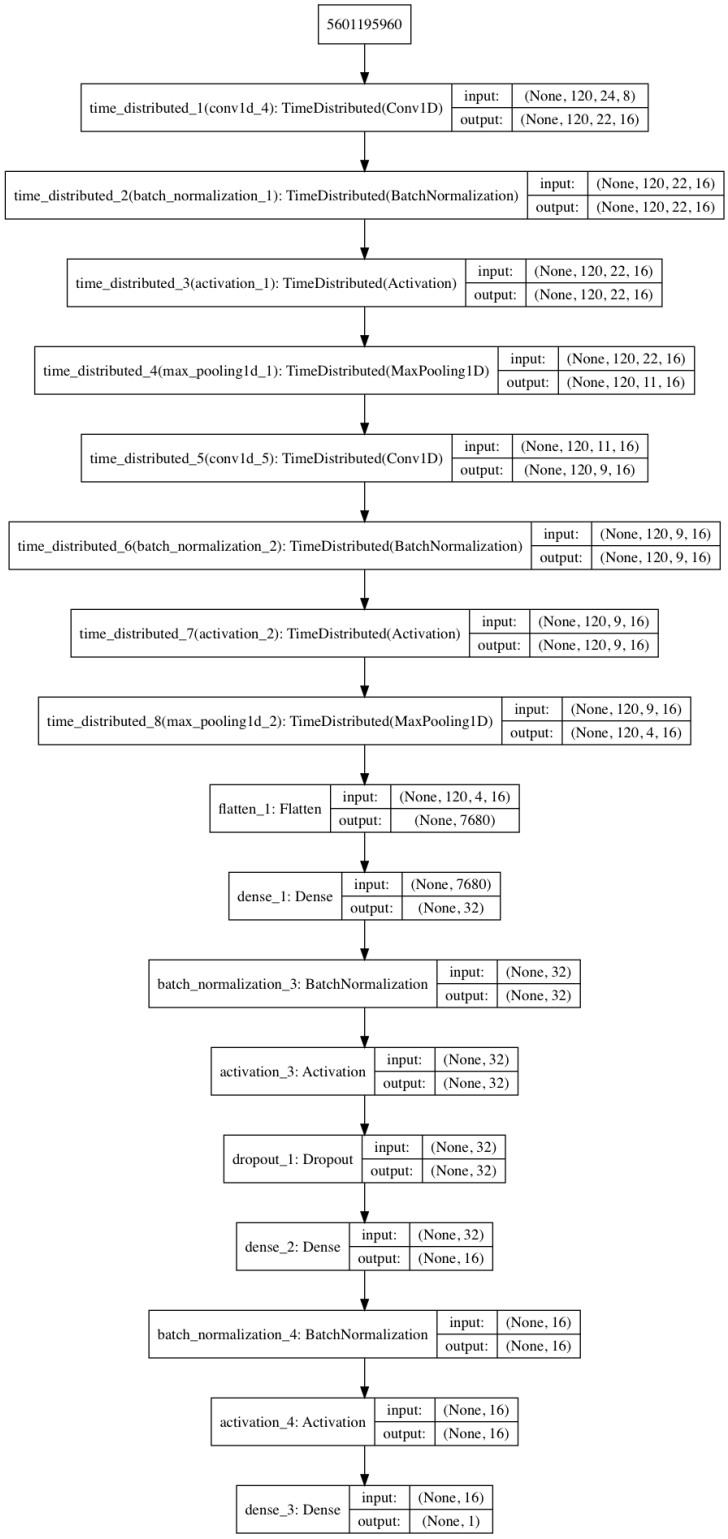



Figure 9: Final convolutional neural network design

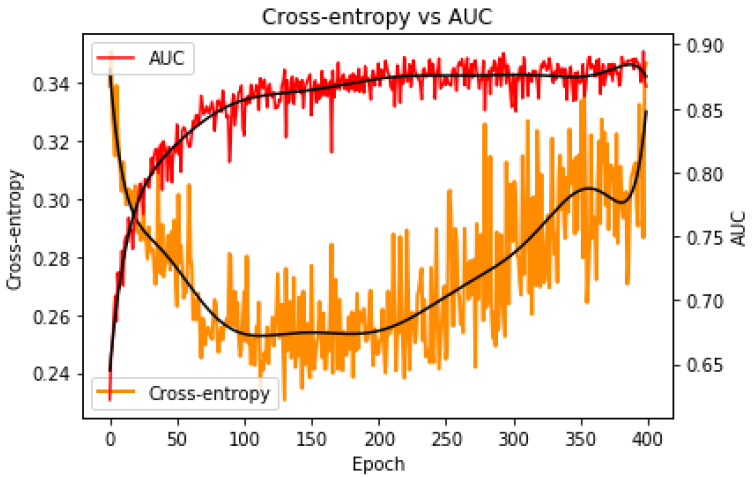



Figure 10: Cross entropy and AUCROC correlation

directly optimising AUCROC by using a differentiable approximation function (Yan et al., 2003). In Figure 10, we can see how AUCROC changes with cross entropy.

An optimiser also had to be chosen for training. For this, Adam (section 3.11.1) was selected, a state-of-the-art optimiser, which is commonly used in deep learning. It provides excellent results without spending much time tuning its hyper-parameters by adaptively changing the learning rate for each parameter.

As mentioned earlier (section 4.1), the classes are highly imbalanced, which could slow down or even prevent the network from learning the preictal class. The class weights were adjusted proportionately to their ratio in the training data to improve the training:

$$W_{class} = \frac{N}{C \cdot N_{class}} \tag{27}$$

where $N$ is the number of samples, $C$ is the number of classes.

The training was done in mini-batches with 200 samples at a time (section 3.9).

# ENSEMBLE MODEL

In this section, two more standalone models and an ensemble model of the three models are introduced.

## 6.1 Random Forest

A model was trained on the same augmented training data using the random forest algorithm (section 3.18). The forest consists of one hundred trees with unlimited depth and each tree are trained with $\sqrt{N}$ randomly chosen features. The class weights were adjusted the same way as described in section 5.3.

A disadvantage of this model is that it is not aware of the time distributed nature of the data, and that will likely affect its performance. The second proposed data augmentation method should help with that (section 4.6).

## 6.2 Logistic Regression

Logistic regression (section 3.19) was used in (Howbert et al., 2014) for seizure prediction and achieved better performance than random guessing. Based on that research, we also trained a logistic regression classifier on the same augmented training data as used for other models. The class weights were adjusted the same way as described in section 5.3.

This model suffers from the same problem as the random forest model, namely that it ignores the time distributed nature of the data. The second proposed data augmentation method should help with that (section 4.6).

## 6.3 Ensemble

Once the three models were trained, three heterogeneous stacking meta-classifiers (section 3.17) were trained on the validation data. The data was split into ten stratified folds to ensure classes are balanced amongst the splits. The training was done using cross-validation (section 3.20). The first model is a simple weighted average of the predictions of the models. The second model is a random forest with 50 trees and unlimited depth. The third model is a simple fully connected neural network with L1 (section 3.3.1) and L2 regularisation (section 3.3.2), binary cross entropy loss and Adam as the optimiser. The network structure can be seen in Figure 11.

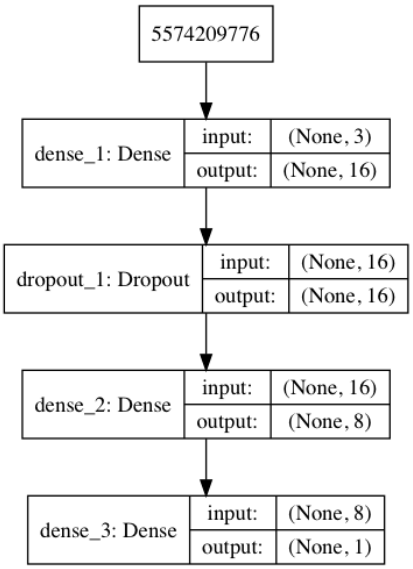



Figure 11: Ensemble classifier neural network

| | Score | Difference |
|---|---|---|
| 60s | 0.8384 | 0.0% |
| 30s | 0.8846 | 5.5% |
| 10s | 0.9035 | 7.7% |
| 5s | 0.8975 | 7.0% |

Table 2: AUC scores for different sliding window sizes

# RESULTS AND EVALUATION

In this section, we discuss our findings and results compared to other algorithms based on the validation data and the private test data from the on-going competition. The training was done on 80% of all the data, the validation set is 10%, and the test set is the private test set, which is the remaining 10%. The ROC curve data for the private test set was kindly provided by the competition hosts.

## 7.1 Preprocessing

We looked at how the network performance is affected by changes to the preprocessing pipeline. It is important to note, that sometimes the tests were only done on a subset of all the data, depending on when the test was carried out during the project. Therefore, the AUC scores might not be consistent between tests done on different parameters, however, this should not affect the usefulness of the changes once used on all the data.

We experimented with increased sliding window resolutions; this can be seen in Table 2. We achieved the best performance by using a ten-second window. It seems the lower the resolution of the sliding window, the more likely the epileptic signs disappear amongst regular brain activity, while if the resolution is too high, we risk not detecting the signs. While the difference between the five and ten-second results is insignificant, the training time and the size of the data is significantly larger with the increase of the resolution. For further tests, a ten-second sliding window was used.

We also introduced overlap in the sliding windows. This further increased our AUC score from 0.8620 to 0.8950, which is a 3.8% increase. For further tests, overlap was used.

Experiments were done on the frequency bands used as well. Set-up 1 is from the original pipeline, Set-up 2 is based on further research (Park et al., 2011) (Korshunova, 2015) and Set-up 3 was done by splitting alpha into alpha 1 and alpha 2 (Garcia-Rill et al., 2016), and beta into beta 1, beta

| | Score | Difference |
|---|---|---|
| Set-up 1 | 0.8842 | 0.0% |
| Set-up 2 | 0.9035 | 2.2% |
| Set-up 3 | 0.8983 | 1.6% |

Table 3: AUC scores for different frequency band configurations

| | AUC without Augmentation | AUC with Augmentation |
|---|---|---|
| Deep Neural Network | 0.80 | 0.91 |
| Random Forest | 0.80 | 0.84 |
| Logistic Regression | 0.75 | 0.78 |

Table 4: Performance of the different models

2 and beta 3 (Kropotov, 2016). The results can be seen in Table 3. As we can see, the configuration with eight bands provided the best results. For further tests, the eight-band set-up was used.

## 7.2 Data Augmentation

We looked at the effect of the proposed data augmentation methods first. We found that the first approach, shuffling the EEG channels, is not useful and the models either don't get better or fail to learn. We suspect this means that there's still useful spatial data even though the channels are not necessarily the same between patients. The other approach; however, turned out to provide significant improvements and was used in all further tests.

This was done based on the validation data, and the results can be seen in Table 4. For the neural network model, the tests were done on the model in Figure 8, as that proved to work better without augmentation and had the same performance as the model in Figure 9 once augmentation was used. However, for all further tests, the latter model was used, as it is significantly faster to train and is less complicated.

According to the results, the performance of all the models increased significantly with the use of the new data points. The model that benefited the most is the neural network, where the increase is 13.75%, which confirms that neural networks generally need more data to learn. As suspected, the models without internal memory also benefitted significantly from this. For the rest of the tests, the models were trained on augmented data.

| | AUC |
|---|---|
| Deep Neural Network | 0.8168 |
| Ensemble with Weighted Average | 0.8042 |
| Ensemble with Neural Network | 0.7979 |
| Ensemble with Random Forest | 0.7888 |
| Random Forest | 0.7350 |

Table 5: Results of the different algorithms on the private test set

## 7.3 Ensemble

Next, we looked at the different ensemble averaging methods used. The ensembles were trained on the validation set with cross-validation; therefore the results are based on the private test results and are compared to the individual models. The results can be seen in Table 5. As we can see, none of the ensembles provided any improvement over the deep network model, so they were not used for the rest of the tests. This is probably due to the models having similar errors, therefore they could not be averaged out and due to the lower performance of the random forest and the logistic regression models, the average performance decreased.

Compared to the results in Table 4, we can also notice a significant drop in AUC scores between the validation and test sets. In the case of the neural network, the difference is -10.2%, and in the case of the random forest, it is -14.3%. This is most likely caused by the validation set also containing patients from the other dataset, and the discrepancy could be avoided by only using patients from the competition for validation purposes.

## 7.4 Competition

The best performing model was compared to the top disclosed algorithms in the competition (Kuhlmann et al., 2018). The results can be seen in Table 6. As we can see, the proposed model performs better on the private test set than the top algorithms while only using a single feature and a single algorithm. Other than the team ranking at the 45th place, none of the top teams used neural networks in their model.

Note, that ROC curves are known to be prone to errors. Using a statistical method proposed in (Hanley and McNeil, 1983), a confidence interval can be computed, and it is possible to compare the AUC scores of the different algorithms and see if there is a significant difference between them. Unfortunately, the data required for this is not available to us, but the competition hosts kindly provided this statistic computed at 95% significance level, that

| | Rank | Score | Features | Algorithms |
|---|---|---|---|---|
| Proposed model | N/A | 0.8168 | 1 | 1 |
| Notsorandomanymore | 1st | 0.8070 | 8 | 4 |
| Arete Associates | 2nd | 0.7990 | 8 | 1 |
| GarethJones | 3rd | 0.7965 | 5 | 2 |
| QingnanTang | 4th | 0.7946 | 3 | 3 |
| Nullset | 5th | 0.7936 | 7 | 5 |
| tralala boum boum pouet pouet | 6th | 0.7920 | 9 | 3 |
| Michaln | 8th | 0.7907 | 10 | 1 |
| H Chipicito+Solver World | 45th | 0.7397 | 3 | 3 |

Table 6: Results of the competition

showed no significant difference between our solution and the top solution.

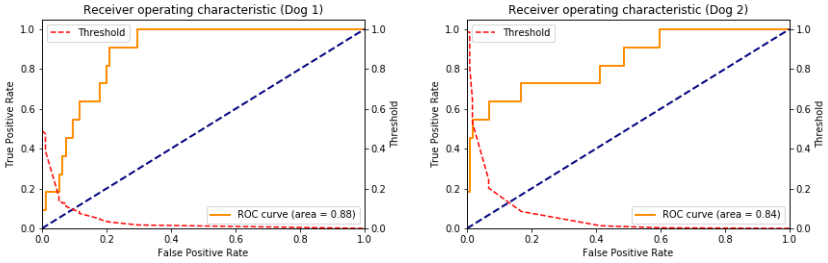
## 7.5 ROC Analysis

ROC curve (section 3.16.1) analysis is used to judge the predictive capabilities of the best performing model. Using this curve, we can find an optimal threshold for hard classification based on our needs. Due to the nature of the task, false positives, as in falsely predicting an upcoming seizure is considered better than missing a seizure. Based on other publications, target specificity (true negative rate) was chosen as 75% (Kuhlmann et al., 2018).

First, we look at the ROC curves of all patients in the validation set to see how well it generalises. As we can see in Figure 12, the algorithm seems to generalise well between species and provide reasonably consistent results with all the dogs. For human patients, the algorithm still seems to have good performance; however, patient 2 seems to have significantly lower AUC score, which is not surprising given the different set-up used for data collection and the low number of samples for this patient.

Next, let's look at the ROC curves of the patients in the private test set of the on-going competition. The ROC curve for all patients can be seen in Figure 13. At 75% sensitivity, the true positive rate is 70.24%, and the hard classification threshold is 0.0726. The confusion matrix was computed at this threshold and can be seen in Table 7.

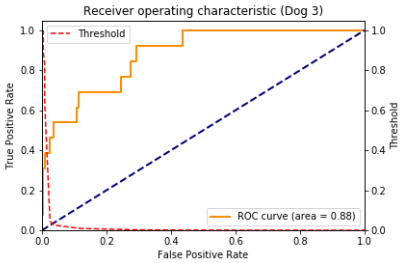
As can be seen, 24.9% of interictal segments are predicted as preictal, which might cause unnecessary anxiety in patients and 29.7% of all preictal segments were predicted to be interictal, which could provide a false sense of security for patients, which could potentially lead to dangerous situations. However, depending on the seizure prevention technique used, this could still



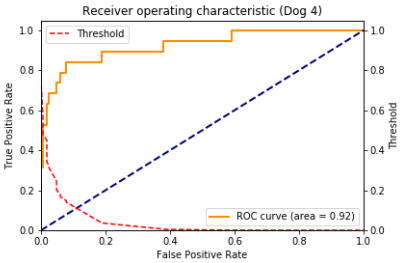

(a) Dog 1



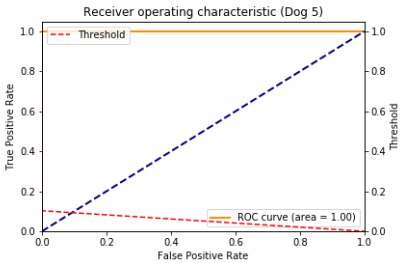

(b) Dog 2



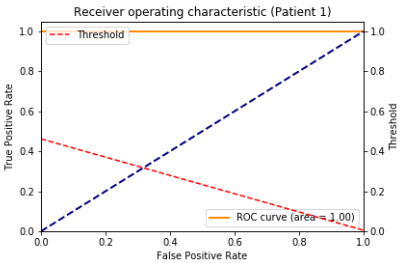

(c) Dog 3



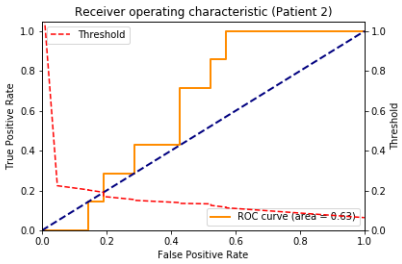

(d) Dog 4



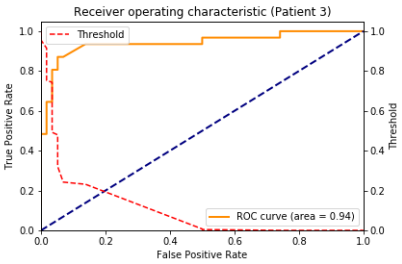

(e) Dog 5



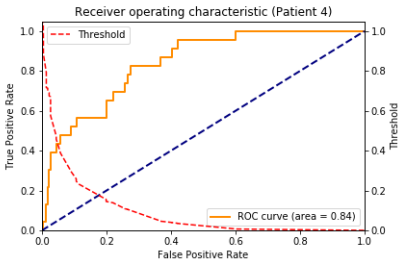

(f) Patient 1



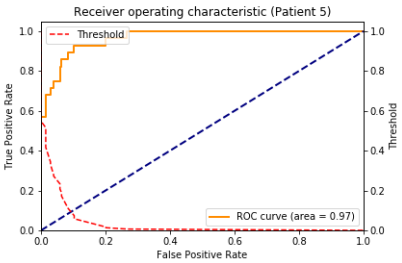

(g) Patient 2




(h) Patient 3




(i) Patient 4




(j) Patient 5



Figure 12: ROC curves of individual patients in the validation set

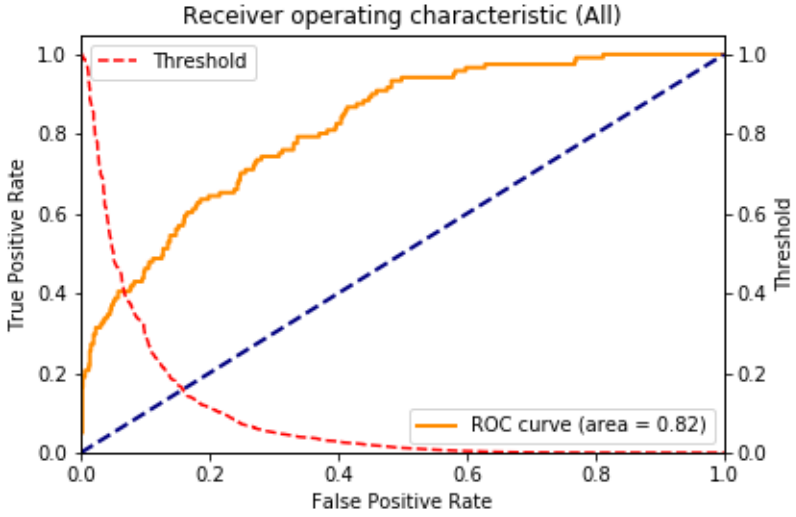


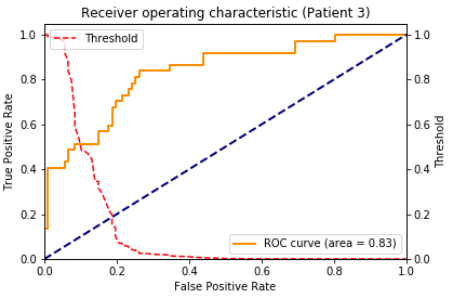

Figure 13: ROC curve of all patients

potentially improve the quality of life of patients, for example by reducing the dosage of a medication.

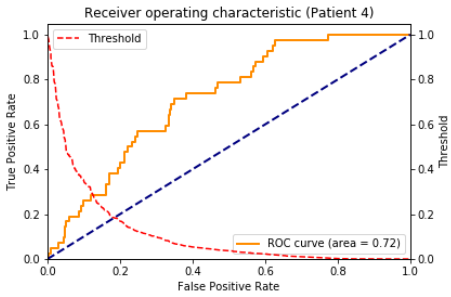
Next, we looked at the ROC curves of individual patients as can be seen in Figure 14 to see if there is a difference between the patients. As we can see, there's a significant difference between the patients. For Patient 4, the algorithm potentially wouldn't prove useful in practice as the true positive rate is only 57.1%, however, for Patient 5 the algorithm could provide a significant increase in quality of life with a true positive rate of 90.5%. For Patient 3, the algorithm performs just above the average with a true positive rate of 78.4% and could potentially prove useful. Note, that the winning algorithm in the Kaggle competition only achieved a minimum AUC of 0.5522 across patients, so it seems our algorithm generalises better across patients (Kuhlmann et al., 2018).

These differences are likely caused by the different types of seizures the patients experience and patient-specific classifiers would likely increase the usefulness of the algorithm. This could be achieved by training the network with all the data for a number of epochs, then using the weights as the initial weights for training patient-specific models.

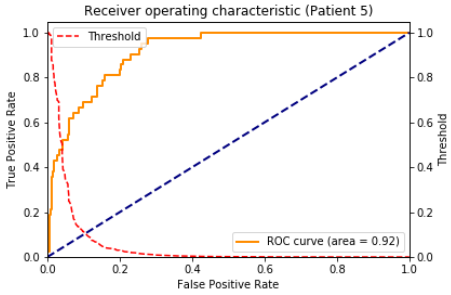
Note, that based on the very low classification threshold found, the output of our network seems to be a poor probability distribution estimator. Using a technique such as Platt-scaling (Platt, 2000), the outputs could be calibrated to provide more meaningful probability values.




(a) Patient 3




(b) Patient 4




(c) Patient 5

Figure 14: ROC curves of individual patients

| True \ Predicted | Interictal | Preictal |
|---|---|---|
| Interictal | 903 | 300 |
| Preictal | 36 | 85 |

Table 7: Confusion matrix for all patients at 0.0726 classification threshold

# DISCUSSION

In this section, we discuss our achievement, the deficiencies of our approach and suggest future work possibilities.

## 8.1 Achievements

As part of the project, we proposed a new data augmentation approach, which is simple, computationally efficient, yet provides significant improvements in all the models we tried.

We experimented with a common preprocessing pipeline used by several algorithms and achieved better performance by increasing the resolution of the sliding window and by introducing overlap. During the experiments, we identified a possible issue with higher resolution sliding windows that can prevent models from learning and proposed a simple solution. While this is not a new approach nor we claim this has not been done before, the precision of our sliding window is higher than the precision of other algorithms, and overlaps are rarely used as well.

We propose a deep convolutional recurrent neural network model, which is capable of competing with all the top algorithms submitted as part of the seizure prediction competition, yet only uses a single feature. The model has a low memory usage and could potentially be used in implanted devices.

We also experimented with ensembles; however, these did not provide any performance improvement.

## 8.2 Deficiencies and Future Work

Our solution only uses a single feature, which might hold back its performance. There's evidence that other features, such as distribution statistics and time of day can improve model performance. This could potentially be explored in the future.

The algorithm is trained on all the data and is not patient-specific. Such models when tested on the full dataset with more patients had a significant decrease in performance. This could potentially render our algorithm useless in practice; however, we do not have access to that dataset to verify this. If we had more time, patient-specific classifiers could be trained, which might also improve performance for Patient 4.

In some cases, such as the loss function and the way the variable size input was handled, simpler approaches were chosen over potentially better solutions. Given the time, implementing and testing experimental loss functions could be done. Instead of the zero-padding approach, using a recursive

network might be more appropriate as well, however, that would potentially increase model complexity.

While we proposed a new augmentation method, we ignored obvious augmentation approaches such as reconstructing the one-hour sequences and creating new segments. Neural networks are known to benefit from more data and given more time this would be worth exploring.

Our ensemble solution did not provide any improvements; however, they have been successfully used by almost every team. Using better models, such as extremely randomised trees, which were a common choice for top algorithms, we could potentially further increase the performance of the model at the risk of increasing model complexity and potentially making the algorithm unusable in practice.

The network structure could probably be improved. Given the resources, an evolutionary algorithm could be developed to find a better solution.

The output of the network seems to estimate the probability distribution poorly; this could be calibrated using a technique such as Platt-scaling (Platt, 2000).

The training was done in mini-batches of 200 samples. Research suggests for best generalisation batches of 2 to 32 should be used; however, due to the significantly higher training time, we could not use batches that small (Masters and Luschi, 2018).

Given access to more data, further tests could have been carried out, such as using the model with previously unseen patients. Using datasets recorded at higher sampling frequencies could potentially improve the predictive capabilities too.

# CONCLUSION

In this section we briefly discuss what we achieved compared to our initial goals and discuss if we managed to make a useful contribution to the field.

Our initial goal was proposing an algorithm that could predict upcoming seizures and provide an improvement in the quality of life for epilepsy patients. While the proposed algorithm has better performance than random guessing and potentially achieves better than the top algorithms for these patients, the accuracy is not high enough to significantly improve the lives of the majority of the patients. However, there's evidence that in some cases it might be an almost perfect solution and for patients who already undergone surgery, it might still be useful.

However, due to the invasive nature of the surgery, the algorithm should not be considered a real solution yet; however, we believe the work was not in vain and could provide useful insights for future research in the field.

# References


Abramovici, S. and Bagić, A. (2016). Chapter 10 - epidemiology of epilepsy. In Aminoff, M. J., Boller, F., and Swaab, D. F., editors, *Neuroepidemiology*, volume 138 of *Handbook of Clinical Neurology*, pages 159 – 171. Elsevier.

Brinkmann, B. H., Wagenaar, J., Abbot, D., Adkins, P., Bosshard, S. C., Chen, M., Tieng, Q. M., He, J., Muñoz-Almaraz, F. J., Botella-Rocamora, P., Pardo, J., Zamora-Martinez, F., Hills, M., Wu, W., Korshunova, I., Cukierski, W., Vite, C., Patterson, E. E., Litt, B., and Worrell, G. A. (2016). Crowdsourcing reproducible seizure forecasting in human and canine epilepsy. *Brain*, 139(6):1713–1722.

C. Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. 12:2121–2159.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Cook, M. J., O'Brien, T. J., Berkovic, S. F., Murphy, M., Morokoff, A., Fabinyi, G., D'Souza, W., Yerra, R., Archer, J., Litewka, L., Hosking, S., Lightfoot, P., Ruedebusch, V., Sheffield, W. D., Snyder, D., Leyde, K., and Himes, D. (2013). Prediction of seizure likelihood with a long-term, implanted seizure advisory system in patients with drug-resistant epilepsy: a first-in-man study. *The Lancet Neurology*, 12(6):563–571.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242.

de Boer, P.-T., Kroese, D., Mannor, S., and Rubinstein, R. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67. Imported from research group DACS (ID number 277).

De Fauw, J., Ledsam, J. R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D., van den Driessche, G., Lakshminarayanan, B., Meyer, C., Mackinder, F., Bouton, S., Ayoub, K., Chopra, R., King, D., Karthikesalingam, A., Hughes, C. O., Raine, R., Hughes, J., Sim, D. A., Egan, C., Tufail, A., Montgomery, H., Hassabis, D., Rees, G., Back, T., Khaw, P. T., Suleyman, M., Cornebise, J., Keane, P. A., and Ronneberger, O. (2018).

Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine*.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232.

Garcia-Rill, E. E., D'Onofrio, S., Luster, B. R., Mahaffey, S., Urbano, F. J., and Phillips, C. (2016). The 10 hz frequency: A fulcrum for transitional brain states. *Translational brain rhythmicity*, 1 1:7–13.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Goller, C. and Kuchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352 vol.1.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Hanley, J. A. and McNeil, B. J. (1983). A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843. PMID: 6878708.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–, Washington, DC, USA. IEEE Computer Society.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Howbert, J. J., Patterson, E. E., Stead, S. M., Brinkmann, B., Vasoli, V., Crepeau, D., Vite, C. H., Sturges, B., Ruedebusch, V., Mavoori, J., Leyde, K., Sheffield, W. D., Litt, B., and Worrell, G. A. (2014). Forecasting seizures in dogs with naturally occurring epilepsy. *PLOS ONE*, 9(1):1–8.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2342–2350. JMLR.org.

Keras (2018). keras-team/keras.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kiral-Kornek, I., Roy, S., Nurse, E., Mashford, B., Karoly, P., Carroll, T., Payne, D., Saha, S., Baldassano, S., O'Brien, T., Grayden, D., Cook, M., Freestone, D., and Harrer, S. (2018). Epileptic seizure prediction using big data and deep learning: Toward a mobile system. *EBioMedicine*, 27:103–111.

Koehn, P. (1994). Combining genetic algorithms and neural networks: The encoding problem.

Korshunova, I. (2015). Epileptic seizure prediction using deep learning.

Krell, M. M. and Kim, S. K. (2017). Rotational data augmentation for electroencephalographic data. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 471–474.

Kropotov, J. D. (2016). Chapter 2.3 - beta and gamma rhythms. In Kropotov, J. D., editor, *Functional Neuromarkers for Psychiatry*, pages 107 – 119. Academic Press, San Diego.

Kuhlmann, L., Karoly, P., Freestone, D. R., Brinkmann, B. H., Temko, A., Barachant, A., Li, F., Titericz, Jr., G., Lang, B. W., Lavery, D., Roman, K., Broadhead, D., Dobson, S., Jones, G., Tang, Q., Ivanenko, I., Panichev, O., Proix, T., Náhlík, M., Grunberg, D. B., Reuben, C.,

Worrell, G., Litt, B., Liley, D. T. J., Grayden, D. B., and Cook, M. J. (2018). Epilepsyecosystem.org: crowd-sourcing reproducible seizure prediction with long-term human intracranial eeg. *Brain*, page awy210.

Lang, B. W., Lavery, D., Roman, K., Dobson, S., and Broadhead, D. (2016). Seizure prediction using short-term and long-term features.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mamun, M., Al-Kadi, M., and Marufuzzaman, M. (2013). Effectiveness of wavelet denoising on electroencephalogram signals. *Journal of Applied Research and Technology*, 11(1):156 – 160.

Manessi, F., Rozza, A., Bianco, S., Napoletano, P., and Schettini, R. (2017). Automated pruning for deep neural network compression. *CoRR*, abs/1712.01721.

Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks.

Mirowski, P. W., LeCun, Y., Madhavan, D., and Kuzniecky, R. (2008). Comparing svm and convolutional networks for epileptic seizure prediction from intracranial eeg. In *2008 IEEE Workshop on Machine Learning for Signal Processing*, pages 244–249.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., and Salem, A.-B. M. (2018). Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, 3(1):68 – 71.

Nielsen, M. A. (2018). Neural networks and deep learning.

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

Platt, J. (2000). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. 10.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018). Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548.

Redmon, J. and Farhadi, A. (2016). Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354 EP –.

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

Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Yan, L., Dodier, R., Mozer, M. C., and Wolniewicz, R. (2003). Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 848–855. AAAI Press.

Yu, H.-F., Huang, F.-L., and Lin, C.-J. (2011). Dual coordinate descent methods for logistic regression and maximum entropy models. *Mach. Learn.*, 85(1-2):41–75.

Zhao, B., Lu, H., Chen, S., Liu, J., and Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition.

Zijlmans, M., Jiruska, P., Zelmann, R., Leijten, F. S., Jefferys, J. G., and Gotman, J. (2012). High-frequency oscillations as a new biomarker in epilepsy. *Annals of Neurology*, 71(2):169–178.

# APPENDICES

## 10.1 Source Code

All files for the project can be found on the git server of the University of Birmingham at the following URL:

https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2017/dxv732

All code can be found in a Jupyter notebook in project/project.ipynb. All code in this file is my own work.

To run, Jupyter with Python 3.6 is needed with the following Python modules installed: numpy, pywavelets, h5p,y tqdm, matplotlib, scipy, numba, keras, tensorflow, scikit-learn