Predicting seizures with artificial neural networks

Jeremy Angel and Matthew Wootten

1% of people around the world have epilepsy, a debilitating condition which prevents them from functioning in everyday life, constantly at risk of losing consciousness and control at any time. Camfield and Camfield (2010) studied a group of 15-30 year olds with idiopathic generalized epilepsy (no brain damage outside of epilepsy) and tonic-clonic seizures (seizures that cause whole-body convulsions). They found that 40% couldn't graduate high school, 33% were unemployed, and 27% had a psychiatric diagnosis. The form of epilepsy these patients had is considered mild; many cases are much worse.

Seizures occur when a defective cluster of neurons sends an unusually strong signal, causing a positive feedback loop where the neuron continuously sends out this same signal. This signal then spreads across the brain, affecting a large area resulting in a seizure. Thus, if we look at brain activity for signs of future abnormality, we may be able to predict an upcoming seizure.

Many existing seizure predictors use electroencephalograms (EEGs). EEGs are recordings of electrical activity in the brain: specifically, the differences in voltage between pairs of electrodes. Each pair of electrodes measures one channel. Such electrodes can be either intracranial (implanted within the head) or extracranial (attached to the scalp). While extracranial EEGs are much more practical, intracranial EEGs are more sensitive (Kuruvilla and Flink, 2003). An EEG recording is made from a relatively small number of channels, each of which is a voltage signal varying over time. By analyzing these, algorithms in the past have been able to predict seizures. Typically, these algorithms begin by picking out a number of *features*, little bits of information that previous research has shown are related to seizure occurrence.

However, the precise interactions between all of these features are fantastically complex. Not all patients experience seizures in the same way, so we need some mechanism to combine all the different features into a single prediction. Machine learning fits this specification well. At its core, machine learning (ML) encompasses the set of algorithms that work not by explicit specification, but by relying on some sort of training data to infer a pattern, which may be too complex for a human to interpret. These machine learning algorithms are able to find patterns regardless of what data they are looking at while explicitly specified algorithms are designed for a single problem. Many existing seizure prediction algorithms have been based on machine-learning methods, and the state-of-the-art algorithms are nearly all ML-based.

We sought to apply a relatively untested algorithm to the field of seizure prediction: spiking neural networks. While these are not new to machine learning, there has been relatively little interest, largely because the most recent advancements in deep learning have focused on more traditional neural network models. However, spiking neural networks have one property that would be very useful for our purposes: they can be efficiently run on low-power specialized hardware (Merolla et al., 2014). This would enable the network to run on an implantable medical device designed to suppress seizures. For comparison purposes, we also chose a more traditional class of neural networks (convolutional), as well as the less powerful support vector machines (SVMs), in order to establish that we needed the level of model complexity we were using.

We took input data from the IEEG database. We selected only one study (Study 005), because Freestone et al. (2017) found that pooling data across patients is ineffective as all patients are different. Different patients might have seizures originating from different locations or seizures that are caused by different triggers making it difficult for a machine learning algorithm to find a pattern across multiple patients. This study was an intracranial recording. We selected one channel (LTD4) which seemed to best represent the channels of the study in general. We selected 151 5-minute positive samples 30 minutes before each seizure, and negative samples randomly from everywhere in the EEG recording at least 30 minutes from an annotated seizure event. Since brain waves are commonly characterized by their frequency, we ran a Fourier transform on the samples to take 135 amplitudes at frequencies between 8-30 Hz for inputs to the convolutional neural network. We specifically chose the 8-30 Hz band (alpha and beta waves) because Darcey & Williamson (1985) found that particular band to have a higher-than-normal predictive value.

Artificial neural networks (referred to here as ANNs or just "neural networks"), one example of a machine learning algorithm, are a loose mathematical model of how the brain functions. They can be used to form remarkably good algorithms for problems that hand-designed algorithms cannot practically solve, such as recognizing complex features in images and signals. An ANN is a network of artificial neurons, which each perform a simple computation, which varies based on the type of neural network. Some of these are input neurons, which take the predictor variables as input. All other neurons take their state from other neurons in the network. Some neurons serve as outputs, which take on values that encode the prediction. Neurons which are neither inputs nor outputs are called "hidden" neurons; they have no expected value. These networks become useful when they are trained, a procedure that gives example inputs and outputs so that the network “learns” to correctly classify outputs (Schmidhuber, 2015).

ANNs are a large class of very different models. While all ANNs are composed of connected neurons, they differ by the connections between those neurons (*topology*), and in the behavior of the neurons themselves. These differences enable a specific *architecture* to be tailored to a specific application (Schmidhuber, 2015).

Multi-layer perceptrons are the simplest ANNs. A multi-layer perceptron network consists of a number of “layers” of neurons. Each layer is connected only to the previous layer as input and the next layer as output. These layers are fully-connected: every neuron in one layer is connected to every neuron in the next layer. Another more complex topology is the *convolutional* neural network. In contrast to the mostly homogeneous layers of multi-layer perceptrons, a convolutional neural network is made up of two distinct stages: a set of layers that perform a convolution and a number of fully connected layers. The convolutional layers apply a simple network to a large number of small snippets of the input: this process is known as a convolution. These outputs are then convolved together repeatedly, gradually finding larger and larger features by combining the outputs of these simple networks. Applying a convolution reduces the volume of data, and also allows the network to better ignore small differences in the position of inputs that have no relevance to the outcome. This is especially useful for multidimensional inputs, like images (Schmidhuber, 2015). Multiple time-varying signals (such as EEGs) can also be used as inputs, as each signal can be treated as a separate dimension (Ren and Wu, 2014). This transformed data is then passed along to the second stage, where it goes through the network as described above.

The previous sections discuss the layout of the neural networks, but make no reference to how the neurons themselves function. There are two broad categories into which neurons fall: traditional and spiking. Traditional neurons take in a single value from each input; all of the inputs together form the input vector. This is then combined with a vector of weights (values that determine how important inputs from certain neurons are), usually by dot product, to get a single value. A per-neuron constant called a bias is then added, then the *activation function* is applied. Common activation functions include sigmoid functions, hyperbolic tangent, and a simple $f(x) = max(x, 0)$, known as ReLU (Schmidhuber, 2015). Each of the output neurons then receives this value. This process continues, cascading throughout the network, until the values are read out from the last layer, the output neurons.

Spiking neural networks are another class of neurons that more closely model how the brain works, and have recently outperformed traditional neural networks in some situations, because they require less power to compute (Lee, Delbruck, and Pfeiffer, 2016). Spiking neurons use a different model. Instead of taking a single value as input, it takes a time-varying value with rapid increases, called a *spike train*. The most common model for a spiking neuron is an *integrate-and-fire neuron*. Integrate-and-fire neurons are modeled as an exponentially decaying signal, with weighted inputs from other neurons added on (Lee et al., 2016).

Most spiking neural networks are single-spiking. Single-spiking means that for every input transmission a neuron in the network receives, the neuron outputs one spike at most. However, Ghosh-Dastidar and Adeli (2009) described a multi-spiking neural network where each neuron can output more than one spike for every input transmission it receives. In a single-spiking neural network, a single connection (called a synapse) links each neuron to the layers next to it. In a multi-spiking neural network, each connection between neurons is made up of multiple synapses, and when a neuron outputs, it outputs along each of the synapses. Each of the synapses in a multi-spiking neural network have different weights and delays determining when the output is received and how much of an effect the output has on internal state of the receiving neuron. The internal state is what determines when a neuron fires. Ghosh-Dastidar and Adeli (2009) found that the multi-spiking neural network outperformed the single-spiking neural network with an accuracy of 92.3% while the single-spiking neural network achieved only an accuracy of 82% on a seizure detection task.

The network architecture alone does not account for any results it can achieve. A neural network must be *trained* in some way to become useful. Aside from the architecture of the network, which the network designer preselects, the only parameters the network has are the weights and biases. Training is fundamentally a multi-dimensional optimization problem, with each parameter being a dimension, and the quantity to minimize a function measuring how far the predictions are from correct (also known as a *loss function*). What "correct" means is often solved by first running through *training data*, data which has been pre-classified (Schmidhuber, 2015).

One highly useful technique for solving this optimization problem is known as *backpropagation*, which uses techniques from calculus to attempt to find a set of parameters where the slopes downward on the cost function (measures of how much of an improvement a nudge in that direction could make) are low. Gradient descent consists of taking large steps in the decreasing direction on the graph where there are large improvements to be made, and small ones where there are not (Schmidhuber, 2015).

Many older models of spiking neural networks, such as Cao et al. (2015), use looser variants of this basic gradient descent backpropagation algorithm, because the derivative of the internal state of a spiking neuron is not strictly defined where the spikes occur, as there is an instantaneous jump in membrane potential. These models resort to training the network on traditional-neuron ANNs, or training with another algorithm entirely. Cao et al. (2015) used a traditional convolutional neural network, trained with backpropagation, and then transformed the weights for use on a spiking neural network.

Lee et al. (2016) more recently described a spiking neural network with single spikes, which can support training directly on the spiking neural network, without the need for any transformation. They accomplished this by ignoring the discontinuities, which they found did not unduly reduce the network's accuracy. Also, while Cao et al. (2015) found that the spiking network was about 2% less accurate than the traditional network at detecting cars, Lee et al. (2016) found that the spiking network performed about 0.4% better at the related task of classifying handwritten digits.

Previous studies also considered support vector machines (SVMs). Pure support vector machines consider inputs to be vectors in a high-dimensional space. They then use hyperplanes to separate the inputs in a way that minimizes error (Cortes & Vapnik, 1995). These hyperplanes are known as decision boundaries; if a point lies on one side, the SVM assigns it to one class; points on the other side are assigned to the other class. We plan to use a variant of SVM that has warped decision boundaries, as described in Boser, Guyon, and Vapnik (1992). This should increase the accuracy of the SVM as it can classify data that is not linearly separable.

Neural networks have been used to predict epileptic seizures. Ramgopal et al. (2014) compared a number of previous attempts at predicting and detecting seizures. These techniques ranged from simple linear regression to multiple chained machine learning techniques, with varying accuracy, sensitivity, and false positive rates. The tradeoff between sensitivity and false positive rate occupies the field. Algorithms exist which achieve perfect sensitivity, and other algorithms achieve a zero false positive rate. However, these algorithms achieve these great results by doing poorly on the other, corresponding, statistic. Netoff et al. (2009), which never incorrectly flags a seizure when there is not one, only catches about 80% of seizures; D'Alessandro et al. (2005), which predicts every seizure that actually occurs, also flags an average of one phantom seizure an hour. Most studies in the field focus on detecting seizures, though there has been success at predicting them ahead of time, as discussed below. Most approaches used EEGs to look at brain activity.

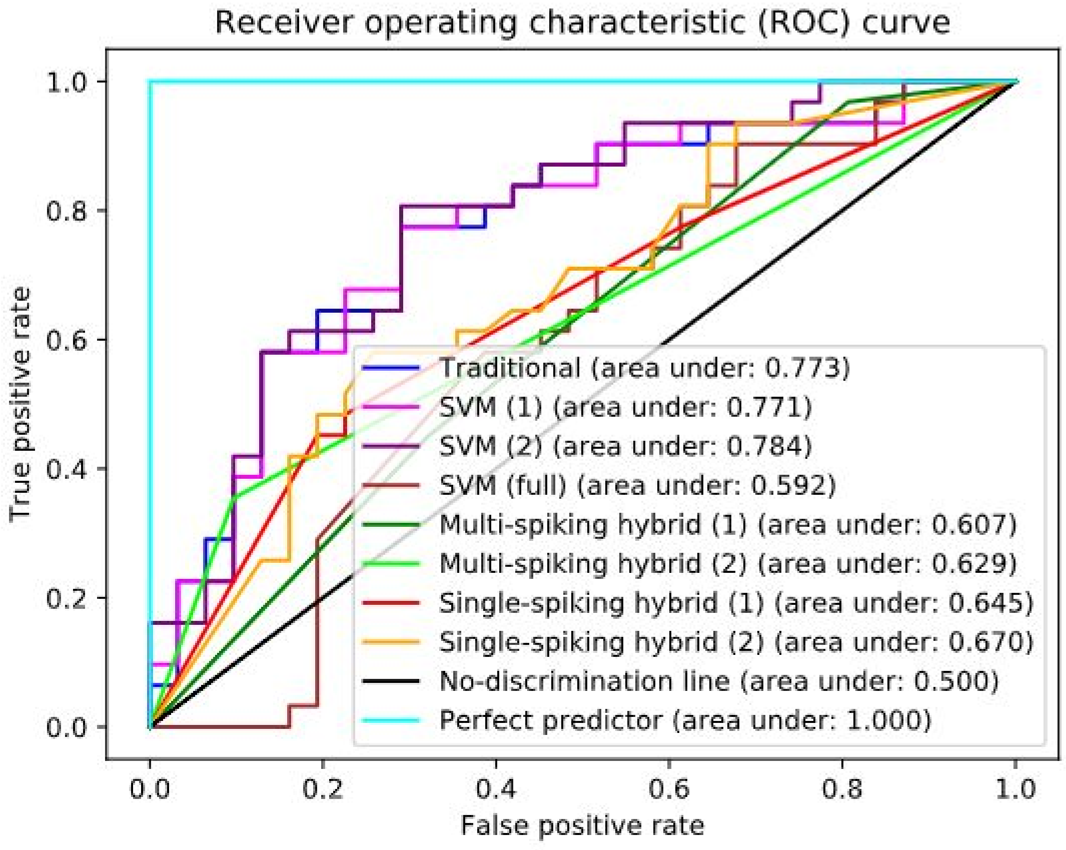
We tested a variety of models. They were all based off of a core convolutional layer; most of our models had additional classifiers as well. Neural networks function by progressively refining their outputs, so we took intermediate values from the networks, which necessarily contain enough information to derive the final output of (at least) the convolutional output, and fed those into other classifiers. We used the hidden layer either one or two layers removed from the output, and fed those in to either a multi-spiking (MuSpiNN) or a support vector machine. The hidden layers use a square 1x1 convolution and an ReLU activation function, while the output layer uses a linear activation. We trained each model for 500 epochs on 80% of the preprocessed EEG data, then tested it on the remaining 20% of the preprocessed EEG data to obtain an ROC curve.

MuSpiNN (1)'s inputs are taken from the last hidden layer of the CNN while MuSpiNN (2)'s inputs are taken from the second to last hidden layer of the CNN. We encoded the real numbers in the CNN output into spike times by taking the height of the intersections with various Gaussian distributions, as described in Ghosh-Dastidar and Adeli (2007). This model simulates how biological sensory neurons process input. The MuSpiNN layers are fully connected and the network is trained on 100 epochs and learns through back propagation via gradient descent.

The SVM (1) and (2) use the same inputs as the MuSpiNN (1) and (2), respectively. The SVM is run with a radial basis function (RBF) kernel.

All of the predictors output some sort of prediction confidence. We can use this to make a tunable detector, one that can be adjusted for different circumstances. For example, a patient at home would have a much higher tolerance for false negatives than a patient in surgery. Convolutional networks make this straightforward, as the outputs vary within the range 0-1 (barring a few exceptions). For the spiking neural network, the output must be shifted and scaled, because its natural output range is from 15-20. While SVMs do not admit a well-defined measure of confidence, we use the distance from the decision boundary as a proxy for one. We collect the these outputs, as well as the true labels from the annotations. We then iterate over each of the outputs, taking it as the cutoff for deciding between a positive or negative class and determining the false positive rate and true positive rate that result from that. The curve formed from these points is a receiver operating characteristic curve (ROC).

The area beneath this curve is the AUROC, which we used to evaluate the relative effectiveness of the neural network models. An ROC connecting the points (0, 0) and (1, 1) is called the *non-discrimination line*, and is achievable by random chance, while an ROC connecting (0, 0), (0, 1), and (1, 1) corresponds to a perfect classifier. The non-discrimination line has an AUROC of 0.5 while a perfect classifier would have an AUROC of 1. Overall, higher AUROCs signify better results.



We found that the best performance came from an SVM, specifically the one operating on the values two layers back from the end. This was very surprising, since we expected the SVMs to perform the worst of all our predictors, due to their relatively low computational capabilities. We hypothesize that this may be related to SVMs' relatively small number of parameters: it may simply be less prone to overfitting than more sophisticated models. Overfitting is when a model memorizes the data it is trained on instead of looking for patterns and is therefore unable to find patterns in new data, resulting in a low testing data accuracy. We also found that the convolutional network outperformed the spiking network. While this was contrary to our hypothesis, this was not shocking. We knew that the network would lose some amount of performance due to information loss in the transformations necessary to get the traditional output into a spiking input. We expected the magnitude of improvement from the spiking network to outweigh this loss; it did not.

Due to the poor performance of the spiking neural networks, we decided to discontinue their use. We were unable to harness their favorable run-time properties. While the execution step was relatively quick, the training step took much longer than comparable non-spiking networks. This is because non-spiking network computations can often be condensed into a small number of matrix multiplications, which are easier to optimize. However, the time dependencies of spiking networks make this approach unviable. This sharply limited the size of the networks we could use, which in turn decrease the potential model complexity and the resulting accuracy. Without having to spend time manipulating our pipeline to accommodate the spiking neural networks, we expect to be able to manipulate many more of the finer details of our model.

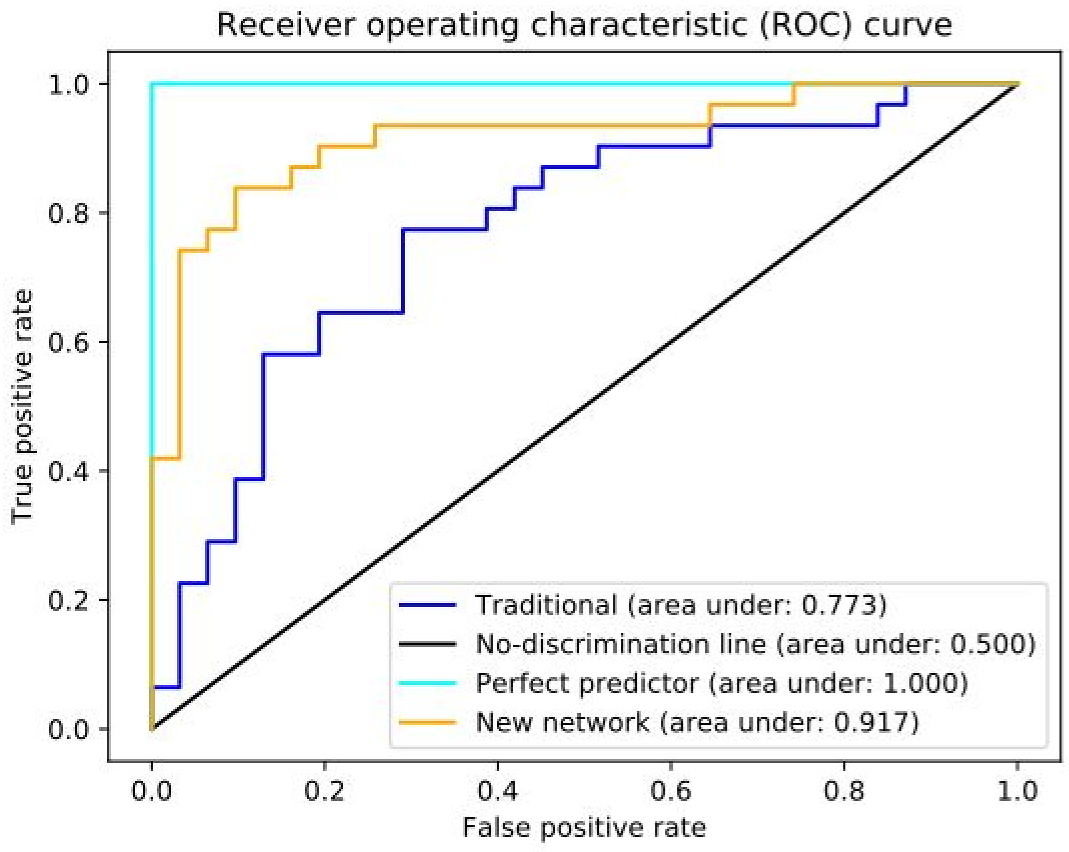
Our first task going forward will be implementing transfer learning. This is a similar concept to our network hybridization described above, except it involves coupling a network trained for one task to a network trained for a slightly different but related task. In our case, we will first train a network with the data from all patients pooled together. This network will have a greater volume of training data, but won't necessarily predict the cues associated with individual seizures. We can then take intermediate values from that network and use it to train smaller networks as the final set of layers for each patient. This will allow our model to specialize for individual patients, and gain some of the advantages of both the pooled and the individual models: large training set size and patient-specificity, respectively.

Specifically, we train the network on pairs of network channels, using the same preprocessing methods as our old setup to get the 135 frequency amplitudes for each

After that, we intend to move from a waveform-based model to a hand-selected feature-based model. We chose the Fourier-transformed input to our model because we were wary of discarding potentially important information, and suspected that our model would be better if we could let it learn the relevant features. However, after examining the results of Tsiouris et al. (2018), who used a feature-based model to great success, we suspect that capturing a wide variety of hand-selected features would allow the network to make good predictions even with a relatively small training sample.

We intend to use a feature set very similar to Tsiouris et al. (2018). One such feature is cross-correlation, a measure of how well-synchronized different channels are. Cross-correlation drops as brain waves start acting in an irregular manner in a particular region right before a seizure, before increasing rapidly as the seizure actually begins and the entire brain is engulfed in the feedback loop. Tsiouris et al. (2018) use a long-short term memory network (LSTM), which work very well for sequence modeling but are difficult to train. We plan to use convolutional networks since we do not currently need the more advanced capabilities of an LSTM, such as the ability to work with variable-length inputs. Convolutional networks have a dramatically simpler training process than LSTMs, so by modifying the various learning hyperparameters we may be able to achieve a better result.

So far, we have implemented a waveform-based model using a convolutional neural network. Currently, preliminary results of this “new network” on data from one patient has been promising, out performing all of our previously tested models with an AUROC of 0.917. For this wave-form based model, we looked at every pair of Fourier-transformed channels for every sample we took as our input and used the median prediction of all pairs as our final prediction. Our next step will be to expand this to be more general, look at feature selection, and tune our hyperparameters.


Receiver operating characteristic (ROC) curve
True positive rate
False positive rate
Traditional (area under: 0.773)
No-discrimination line (area under: 0.500)
Perfect predictor (area under: 1.000)
New network (area under: 0.917)