Deep Learning Approaches for Automated Seizure Detection from Scalp EEGs

Meysam Golmohammadi[1, 2], Vinit Shah[2], Iyad Obeid[2], Joseph Picone[2]

[1] Internet Brands, El Segundo, California, USA
[2] The Neural Engineering Data Consortium, Temple University, Philadelphia, Pennsylvania, USA

Abstract. Scalp electroencephalograms (EEGs) are the primary means by which physicians diagnose brain-related illnesses such as epilepsy and seizures. Automated seizure detection using clinical EEGs is a very difficult machine learning problem due to the low fidelity of a scalp EEG signal. Nevertheless, despite the poor signal quality, clinicians can reliably diagnose illnesses from visual inspection of the signal waveform. Commercially available automated seizure detection systems, however, suffer from unacceptably high false alarm rates. Deep learning algorithms that require large amounts of training data have not previously been effective on this task due to the lack of big data resources necessary for building such models and the complexity of the signals involved. The evolution of big data science, most notably the release of the Temple University EEG (TUEG) Corpus, has motivated renewed interest in this problem.

In this chapter, we discuss the application of a variety of deep learning architectures to automated seizure detection. Architectures explored include multilayer perceptrons, convolutional neural networks (CNNs), long short-term memory networks (LSTMs), gated recurrent units and residual neural networks. We use the TUEG Corpus, supplemented with data from Duke University, to evaluate the performance of these hybrid deep structures. Since TUEG contains a significant amount of unlabeled data, we also discuss unsupervised pre-training methods used prior to training these complex recurrent networks.

Exploiting spatial and temporal context is critical for accurate disambiguation of seizures from artifacts. We explore how effectively several conventional architectures are able to model context and introduce a hybrid system that integrates CNNs and LSTMs. The primary error modalities observed by this state-of-the-art system were false alarms generated during brief delta range slowing patterns such as intermittent rhythmic delta activity. A variety of these types of events have been observed during inter-ictal and post-ictal stages. Training models on such events with diverse morphologies has the potential to significantly reduce the remaining false alarms. This is one reason we are continuing our efforts to annotate a larger portion of TUEG. Increasing the data set size significantly allows us to leverage more advanced machine learning methodologies.

Keywords: Deep Learning, Convolutional Neural Networks, Electroencephalography, Generative Adversarial Networks, Long Short-Term Networks, Recurrent Neural Networks, seizure detection

1 Introduction

An EEG records the electrical activity along the scalp and measures spontaneous electrical activity of the brain. The signals measured along the scalp can be correlated with brain activity, which makes it a primary tool for diagnosis of brain-related illnesses [1, 2]. Electroencephalograms (EEGs) are used in a broad range of health care institutions to monitor and record electrical activity in the brain. EEGs are essential in the diagnosis of clinical conditions such as epilepsy, depth of anesthesia, coma, encephalopathy, brain death and even in the progression of Alzheimer's disease [3, 4].

Manual interpretation of EEGs is time-consuming since these recordings may last hours or days. It is also an expensive process as it requires highly trained experts. Therefore, high performance automated analysis of EEGs can reduce time of diagnosis and enhance real-time applications by flagging sections of the signal that need further review. Many methods have been developed over the years [5], including time-frequency digital signal processing techniques [6, 7], autoregressive spectral analysis [8], wavelet analysis [9], nonlinear dynamical analysis [10], multivariate techniques based on simulated leaky integrate-and-fire neurons [11–13] and expert systems that attempt to mimic a human observer [14]. In spite of recent research progress in this field, the transition of automated EEG analysis technology to commercial products in operational use in clinical settings has been limited, mainly because of unacceptably high false alarm rates [15–17].

In recent years, progress in machine learning and big data resources has enabled a new generation of technology that is approaching acceptable levels of performance for clinical applications. The main challenge in this task is to operate with an extremely low false alarm rate. A typical critical care unit contains 12 to 24 beds. Even a relatively low false alarm rate of 5 false alarms (FAs) per 24 hours per patient, which translates to between 60 and 120 false alarms per day, would overwhelm healthcare staff servicing these events. This is especially true when one considers the amount of other equipment that frequently trigger alerts [18]. In this chapter, we discuss the application of deep learning technology to the automated EEG interpretation problem and introduce several promising architectures that deliver performance close to the requirements for operational use in clinical settings.

1.1 Leveraging Recent Advances in Deep Learning

Machine learning has made tremendous progress over the past three decades due to rapid advances in low-cost highly-parallel computational infrastructure, powerful machine learning algorithms, and, most importantly, big data. Although contemporary approaches for automatic interpretation of EEGs have employed more modern machine learning approaches such as neural networks [19, 20] and support vector machines [21], state-of-the-art machine learning algorithms have not previously been utilized in EEG analysis because of a lack of big data resources. A significant big data resource, known as the TUH EEG Corpus (TUEG) is now available creating a unique opportunity to evaluate high performance deep learning approaches [22]. This database includes detailed physician reports and patient medical histories, which are critical to the application of deep learning. However, transforming physicians' reports into meaningful information that can be exploited by deep learning paradigms is proving to be challenging because the mapping of reports to underlying EEG events is nontrivial.

Though modern deep learning algorithms have generated significant improvements in performance in fields such as speech and image recognition, it is far from trivial to apply these approaches to new domains, especially applications such as EEG analysis that rely on waveform interpretation. Deep learning approaches can be viewed as a broad family of neural network algorithms that use a large number of layers of nonlinear processing units to learn a mapping between inputs and outputs. These algorithms are usually trained using a combination of supervised and unsupervised learning. The best overall approach is often determined empirically and requires extensive experimentation for optimization. There is no universal theory on how to arrive at the best architecture, and the results are almost always heavily data dependent. Therefore, in this chapter we will present a variety of approaches and establish some well-calibrated benchmarks of performance. We explore two general classes of deep neural networks in detail.

The first class is a Convolutional Neural Network (CNN), which is a class of deep neural networks that have revolutionized fields like image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing through end to end learning from raw data [23]. An interesting characteristic of CNNs that was leveraged in these applications is their ability to learn local patterns in data by using convolutions, more precisely cross-correlation, as their key component. This property makes them a powerful candidate for modeling EEGs which are inherently multichannel signals. Each channel in an EEG possesses some spatial significance with respect to the type and locality of a seizure event [24]. EEGs also have an extremely low signal to noise ratio and events of interest such as seizures are easily confused with signal artifacts (e.g., eye movements) or benign variants (e.g., slowing) [25]. The spatial property of the signal is an important cue for disambiguating these types of artifacts from seizures. These properties make modeling EEGs more challenging compared to more conventional applications like image recognition of static images or speech recognition using a single microphone. In this study, we adapt well-known CNN architectures to be more suitable for automatic seizure detection. Leveraging a high-performance time-synchronous system that provides accurate segmentation of the signal is also crucial to the development of these kinds of systems. Hence, we use a hidden Markov model (HMM) based approach as a non-deep learning baseline system [26].

Optimizing the depth of a CNN is crucial to achieving state-of-the-art performance. Best results are achieved on most tasks by exploiting very deep structures (e.g., thirteen layers are common) [27, 28]. However, training deeper CNN structures is more difficult since they are prone to degradation in performance with respect to generalization and suffer from convergence problems. Increasing the depth of a CNN incrementally often saturates sensitivity and also results in a rapid decrease in sensitivity. Often increasing the number of layers also increases the error on the training data due to convergence issues, indicating that the degradation in performance is not created by overfitting. We address such degradations in performance by designing deeper CNNs using a deep residual learning framework (ResNet) [29].

We also extend the CNN approach by introducing an alternate structure, a deep convolutional generative adversarial network (DCGAN) [30] to allow unsupervised training. Generative adversarial networks (GANs) [31] have emerged as powerful techniques for learning generative models based on game theory. Generative models use an analysis by synthesis approach to learn the essential features of data required for high performance classification using an unsupervised approach. We introduce techniques to stabilize the training of DCGAN for spatio-temporal modeling of EEGs.

The second class of network that we discuss is a Long Short-Term Memory (LSTM) network. LSTMs are a special kind of recurrent neural network (RNN) architecture that can learn long-term dependencies. This is achieved by introducing a new structure called a memory cell and by adding multiplicative gate units that learn to open and close access to the constant error flow [32]. It has been shown that LSTMs are capable of learning to bridge minimal time lags in excess of 1,000 discrete time steps. To overcome the problem of learning long-term dependencies in modeling of EEGs, we describe a few hybrid systems composed of LSTMs that model both spatial relationships (e.g., cross-channel dependencies) and temporal dynamics (e.g., spikes). In an alternative approach for sequence learning of EEGs, we propose a structure based on gated recurrent units (GRUs) [33]. A GRU is a gating mechanism for RNNs that is similar in concept to what LSTMs attempt to accomplish. It has been shown that GRUs can outperform many other RNNs, including LSTM, in several datasets [33].

1.2 Big Data Enables Deep Learning Research

Recognizing that deep learning algorithms require large amounts of data to train complex models, especially when one attempts to process clinical data with a significant number of artifacts using specialized models, we have developed a large corpus of EEG data to support this kind of technology development. The TUEG Corpus is the largest publicly available corpus of clinical EEG recordings in the world. The most recent release, v1.1.0, includes data from 2002 – 2015 and contains over 23,000 sessions from over 13,500 patients – over 1.8 years of multichannel signal data in total [22]. This dataset was collected at the Department of Neurology at Temple University Hospital. The data includes sessions taken from outpatient treatments, Intensive Care Units (ICU) and Epilepsy Monitoring Units (EMU), Emergency Rooms (ER) as well as several other locations within the hospital. Since TUEG consists entirely of clinical data, it contains many real-world artifacts (e.g., eye blinking, muscle artifacts, head movements). This makes it an extremely challenging task for machine learning systems and differentiates it from most research corpora currently available in this area. Each of the sessions contains at least one EDF file and one physician report. These reports are generated by a board-certified neurologist and are the official hospital record. These reports are comprised of unstructured text that describes the patient, relevant history, medications, and clinical impression. The corpus is publicly available from the Neural Engineering Data Consortium (*www.nedcdata.org*).

EEG signals in TUEG were recorded using several generations of Natus Medical Incorporated's NicoletTM EEG recording technology [34]. The raw signals consist of multichannel recordings in which the number of channels varies between 20 and 128 channels [24, 35]. A 16-bit A/D converter was used to digitize the data. The sample frequency varies from 250 Hz to 1024 Hz. In our work, we resample all EEGs to a sample frequency of 250 Hz. The Natus system stores the data in a proprietary format that has been exported to EDF with the use of NicVue v5.71.4.2530. The original EEG records are split into multiple EDF files depending on how the session was annotated by the attending technician. For our studies, we use the 19 channels associated with a standard 10/20 EEG configuration and apply a Transverse Central Parasagittal (TCP) montage [36, 37].

A portion of TUEG was annotated manually for seizures. This corpus is known as the TUH EEG Seizure Detection Corpus (TUSZ) [38]. TUSZ is also the world's largest publicly available corpus of annotated data for seizure detection that is unencumbered. No data sharing or IRB agreements are needed to access the data. TUSZ contains a rich variety of seizure morphologies. Variation in onset and termination, frequency and amplitude, and locality and focality protect the corpus from a bias towards one type of seizure morphology. TUSZ, which reflects a seizure detection task, is the focus of the experiments presented in this chapter. For related work on six-way classification of EEG events, see [26, 39, 40].

We have also included an evaluation on a held-out data set based on the Duke University Seizure Corpus (DUSZ) [41]. The DUSZ database is collected solely from the adult ICU patients exhibiting non-convulsive seizures. These are continuous EEG (cEEG) records [42] where most seizures are focal and slower in frequency. TUSZ in contrast contains records from a much broader range of patients and morphologies. A comparison of these two corpora is shown in Table 1. The evaluation sets are comparable in terms of the number of patients and total amount of data, but TUSZ contains many more sessions collected from each patient.

It is important to note that TUSZ was collected using several generations of Natus Incorporated EEG equipment [34], while DUSZ was collected at a different hospital, Duke University, using a Nihon Kohden system [43]. Hence, using DUSZ as a held-out evaluation set is an important benchmark because it tests the robustness of the

Table 1. An overview of the corpora used to develop the technology described in this chapter

Description	TUSZ		DUSZ
	Train	**Eval**	**Eval**
Patients	64	50	45
Sessions	281	229	45
Files	1,028	985	45
Seizure (secs)	17,686	45,649	48,567
Non-Seizure (secs)	596,696	556,033	599,381
Total (secs)	614,382	601,682	647,948

models to variations in the recording conditions. Deep learning systems are notoriously prone to overtraining, so this second data set represents important evidence that the results presented here are generalizable and reproducible on other tasks.

2 Temporal Modeling of Sequential Signals

The classic approach to machine learning, shown in Fig. 1, involves an iterative process that begins with the collection and annotation of data and ends with an open set, or blind, evaluation. Data is usually sorted into training (train), development test set (dev_test) and evaluation (eval). Evaluations on the dev_test data is used to guide system development. One cannot adjust system parameters based on the outcome of evaluations on the eval set but can use these results to assess overall system performance. We typically iterate on all aspects of this approach, including expansion and repartitioning of the training and dev_test data, until overall system performance is optimized.

We often leverage previous stages of technology development to seed, or initialize, models used in a new round of development. Further, there is often a need to temporally segment the data, for example automatically labeling events of interest, to support further explorations of the problem space. Therefore, it is common when exploring new applications to begin with a familiar technology. As previously mentioned, EEG signals have a strong temporal component. Hence, a likely candidate for establishing good baseline results is an HMM approach, since this algorithm is particularly strong at automatically segmenting the data and localizing events of interest.

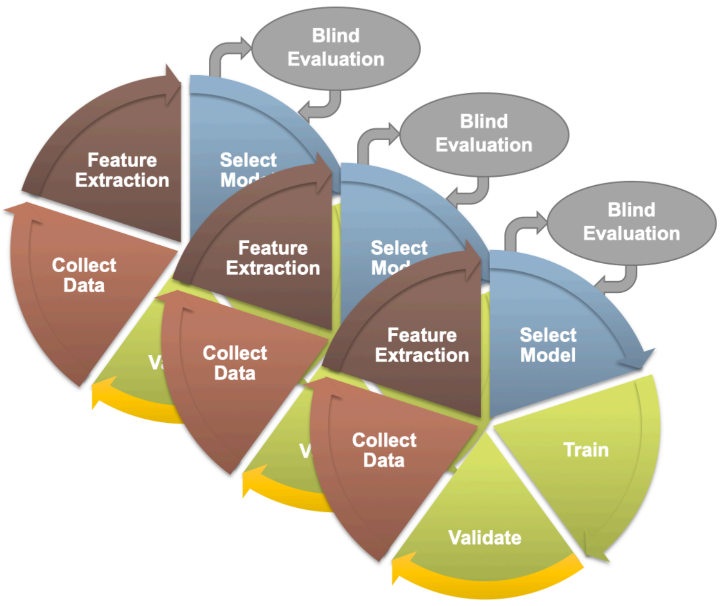

Fig. 1. An overview of a typical design cycle for machine learning

HMM systems typically operate on a sequence of vectors referred to as features. In this section, we briefly introduce the feature extraction process we have used, and we describe a baseline system that integrates hidden Markov models for sequential decoding of EEG events with deep learning for decision-making based on temporal and spatial context.

2.1 A Linear Frequency Cepstral Coefficient Approach to Feature Extraction

The first step in our machine learning systems consists of converting the signal to a sequence of feature vectors [44]. Common EEG feature extraction methods include temporal, spatial and spectral analysis [45, 46]. A variety of methodologies have been broadly applied for extracting features from EEG signals including a wavelet transform, independent component analysis and autoregressive modeling [47, 48]. In this study, we use a methodology based on mel-frequency cepstral coefficients (MFCC) which have been successfully applied to many signal processing applications including speech recognition [44]. In our systems, we use linear frequency cepstral coefficients (LFCCs) since a linear frequency scale provides some slight advantages over the mel scale for EEG signals [40]. A block diagram summarizing the feature extraction process used in this work is presented in Fig. 2. Though it is increasingly popular to operate directly from sampled data in a deep learning system, and let the system learn the best set of features automatically, for applications in which there is limited annotated data, it is often more beneficial to begin with a specific feature extraction algorithm. Experiments with different types of features [49] or with using sampled data directly [50] have not shown a significant improvement in performance.

Harati et. al. [40] did an extensive exploration of many of the common parameters associated with feature extraction and optimized the process for six-way event classification. We have found this approach, which leverages a popular technique in speech recognition, is remarkably robust across many types of machine learning applications.

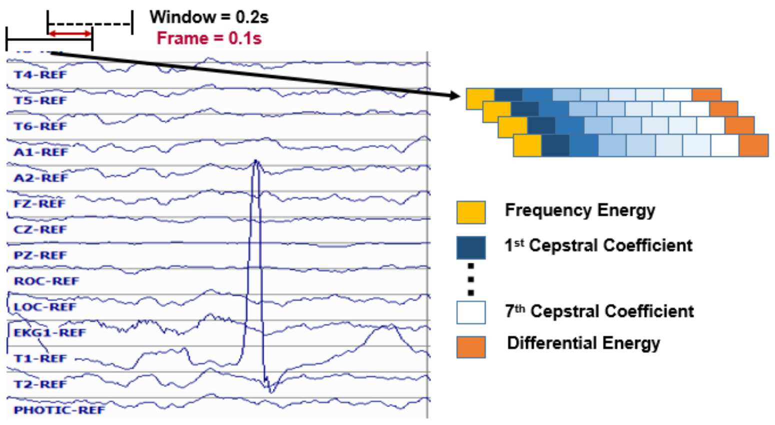

Fig. 2. Base features are calculated using linear frequency cepstral coefficients

The LFCCs are computed by dividing raw EEG signals into shorter frames using a standard overlapping window approach. A high resolution Fast Fourier Transform (FFT) is computed next. The spectrum is downsampled with a filter bank composed of an array of overlapping bandpass filters. Finally, the cepstral coefficients are derived by computing a discrete cosine transform of the filter bank's output [44]. In our experiments, we discarded the zeroth-order cepstral coefficient, and replaced it with a frequency domain energy term which is calculated by adding the output of the oversampled filter bank after they are downsampled:

$$\mathrm{E_f} = \log\left(\sum_{\mathrm{k}=0}^{\mathrm{N}-1} |\mathrm{X(k)}|^2\right). \tag{1}$$

We also introduce a new feature, called differential energy, that is based on the long-term differentiation of energy. Differential energy can significantly improve the results of spike detection, which is a critical part of seizure detection, because it amplifies the differences between transient pulse shape patterns and stationary background noise. To compute the differential energy term, we compute the energy of a set of consecutive frames, which we refer to as a window, for each channel of an EEG:

$$E_d = \max_m \left(E_f(m)\right) - \min_m \left(E_f(m)\right). \tag{2}$$

We have used a window of 9 frames which is 0.1 secs in duration, corresponding to a total duration of 0.9 secs, to calculate differential energy term. Even though this term is a relatively simple feature, it resulted in a statistically significant improvement in spike detection performance [40].

Our experiments have also shown that using regression-based derivatives of features, which is a popular method in speech recognition [44], is effective in the classification of EEG events. We use the following definition for the derivative:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}. \tag{3}$$

Eq. (3) is applied to the cepstral coefficients, c_t, to compute the first derivatives, which are referred to as delta coefficients. Eq. (3) is then reapplied to the first derivatives to compute the second derivatives, which are referred to as delta-delta coefficients. Again, we use a window length of 9 frames (0.9 secs) for the first derivative and a window length of 3 (0.3 secs) for the second derivative. The introduction of derivatives helps the system discriminate between steady-state behavior, such as that found in a periodic lateralized epileptiform discharges (PLED) event, and impulsive or nonstationary signals, such as that found in spikes (SPSW) and eye movements (EYEM).

Through experiments designed to optimize feature extraction, we found best performance can be achieved using a feature vector length of 26. This vector includes nine absolute features consisting of seven cepstral coefficients, one frequency-domain energy term, and one differential energy term. Nine deltas are added for these nine absolute features. Eight delta-deltas are added because we exclude the delta-delta term for differential energy [40].

2.2 Temporal and Spatial Context Modeling

HMMs are among the most powerful statistical modeling tools available today for signals that have both a time and frequency domain component [51]. HMMs have been used quite successfully in sequential decoding tasks like speech recognition [52], cough detection [53] and gesture recognition [54] to model signals that have sequential properties such as temporal or spatial evolution. Automated interpretation of EEGs is a problem like speech recognition since both time domain (e.g., spikes) and frequency domain information (e.g., alpha waves) are used to identify critical events [55]. EEGs have a spatial component as well.

A left-to-right channel-independent GMM-HMM, as illustrated in Fig. 3, was used as a baseline system for sequential decoding [26]. HMMs are attractive because training is much faster than comparable deep learning systems, and HMMs tend to work well when moderate amounts of annotated data are available. We divide each channel of an EEG into 1 sec epochs, and further subdivide these epochs into a sequence of 0.1 sec frames. Each epoch is classified using an HMM trained on the subdivided epoch. These epoch-based decisions are postprocessed by additional statistical models in a process that parallels the language modeling component of a speech recognizer. Standard three state left-to-right HMMs [51] with 8 Gaussian mixture components per state were used. The covariance matrix for each mixture component was assumed to be a diagonal matrix – a common assumption for cepstral-based features. Though we evaluated both channel-dependent and channel-independent models, channel-independent models were ultimately used because channel-dependent models did not provide any improvement in performance.

Supervised training based on the Baum-Welch reestimation algorithm was used to train two models – seizure and background. Models were trained on segments of data containing seizures based on manual annotations. Since seizures comprise a small percentage of the overall data (3% in the training set; 8% in the evaluation set), the amount of non-seizure data was limited to be comparable to the amount of seizure data, and

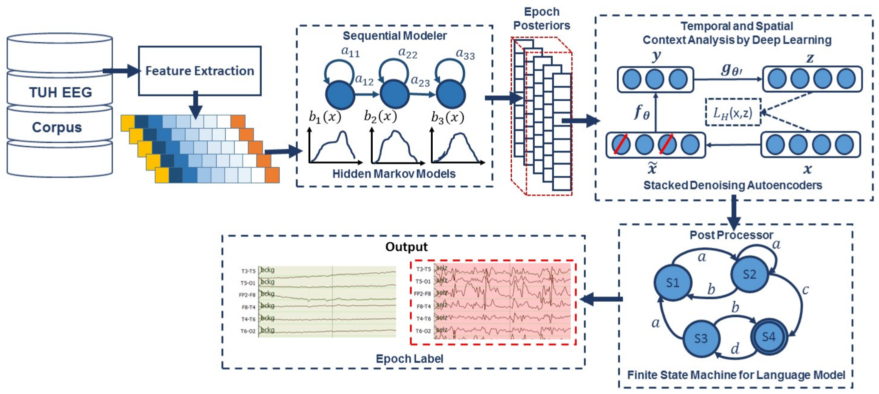

Fig. 3. A hybrid architecture based on HMMs

non-seizure data was selected to include a rich variety of artifacts such as muscle and eye movements. Twenty iterations of Baum-Welch were used though performance is not very sensitive to this value. Standard Viterbi decoding (no beam search) was used in recognition to estimate the model likelihoods for every epoch of data. The entire file was not decoded as one stream because of the imbalance between the seizure and background classes – decoding was restarted for each epoch.

The output of the epoch-based decisions was postprocessed by a deep learning system. Our baseline system used a Stacked denoising Autoencoder (SdA) [56, 57] as shown in Fig. 3. SdAs are an extension of stacked autoencoders and are a class of deep learning algorithms well-suited to learning knowledge representations that are organized hierarchically [58]. They also lend themselves to problems involving training data that is sparse, ambiguous or incomplete. Since inter-rater agreement is relatively low for seizure detection [16], it made sense to evaluate this type of algorithm as part of a baseline approach.

An N-channel EEG was transformed into N independent feature streams. The hypotheses generated by the HMMs were postprocessed using a second stage of processing that examines temporal and spatial context. We apply a third pass of postprocessing that uses a stochastic language model to smooth hypotheses involving sequences of events so that we can suppress spurious outputs. This third stage of postprocessing provides a moderate reduction in false alarms.

Training of SdA networks are done in two steps: (1) pre-training in a greedy layer-wise approach [58] and (2) fine-tuning by adding a logistic regression layer on top of the network [59]. The output of the first stage of processing is a vector of two likelihoods for each channel at each epoch. Therefore, if we have 22 channels and 2 classes (seizure and background), we will have a vector of dimension 2 x 22 = 44 for each epoch.

Each of these scores is independent of the spatial context (other EEG channels) or temporal context (past or future epochs). To incorporate context, we form a supervector consisting of N epochs in time using a sliding window approach. We find it beneficial to make N large – typically 41. This results in a vector of dimension 41 x 44 = 1,804 that needs to be processed each epoch. The input dimensionality is too high considering the amount of manually labeled data available for training and the computational requirements. To deal with this problem we used Principal Components Analysis (PCA) [60, 61] to reduce the dimensionality to 20 before applying the SdA postprocessing.

The parameters of the SdA model are optimized to minimize the average reconstruction error using a cross-entropy loss function. In the optimization process, a variant of stochastic gradient descent is used called "Minibatch stochastic gradient descent" (MSGD) [62]. MSGD works identically to stochastic gradient descent, except that we use more than one training example to make each estimate of the gradient. This technique reduces variance in the estimate of the gradient, and often makes better use of the hierarchical memory organization in modern computers.

The SdA network has three hidden layers with corruption levels of 0.3 for each layer. The number of nodes per layer are: 1st layer (connected to the input layer) = 800, 2nd layer = 500, 3rd layer (connected to the output layer) = 300. The parameters for pre-training are: learning rate = 0.5, number of epochs = 150, batch size = 300. The parameters for fine-tuning are: learning rate = 0.1, number of epochs = 300, batch size = 100. The overall result of the second stage is a probability vector of dimension two containing a likelihood that each label could have occurred in the epoch. A soft decision paradigm is used rather than a hard decision paradigm because this output is smoothed in the third stage of processing. A more detailed explanation about the third pass of processing is presented in [63].

3 Improved Spatial Modeling Using CNNs

Convolutional Neural Networks (CNNs) have delivered state of the art performance on highly challenging tasks such as speech [64] and image recognition [28]. These early successes played a vital role in stimulating interest in deep learning approaches. In this section we explore modeling of spatial information in the multichannel EEG signal to exploit our knowledge that seizures occur on a subset of channels [2]. The identity of these channels also plays an important role localizing the seizure and identifying the type of seizure [65].

3.1 Deep Two-Dimensional Convolutional Neural Networks

CNN networks are usually composed of convolutional layers and subsampling layers followed by one or more fully connected layers. Consider an image of dimension $W \times H \times N$, where W and H are the width and height of the image in pixels, and N is the number of channels (e.g. in an RGB image, $N = 3$ since there are three colors). Two-dimensional (2D) CNNs commonly used in sequential decoding problems such as speech or image recognition typically consist of a convolutional layer that will have K filters (or kernels) of size $M \times N \times Q$ where M and N are smaller than the dimension of the data and Q is smaller than the number of channels. The image can be subsampled by skipping samples as you convolve the kernel over the image. This is known as the stride, which is essentially a decimation factor. CNNs have a large learning capacity that can be controlled by varying their depth and breadth to produce K feature maps of size $(W - M + 1) \times (H - N + 1)$ for a stride of 1, and proportionally smaller for larger strides. Each map is then subsampled using a technique known as max pooling [66], in which a filter is applied to reduce the dimensionality of the map. An activation function, such as a rectified linear unit (ReLU), is applied to each feature map either before or after the subsampling layer to introduce nonlinear properties to the network. Nonlinear activation functions are necessary for learning complex functional mappings.

In Fig. 4, a system that combines CNN and a multi-layer perceptron (MLP) [28] is shown. Drawing on our image classification analogy, each image is a signal where the width of the image (W) is the window length multiplied by the number of samples per second, the height of the image (H) is the number of EEG channels and the number of image channels (N) is the length of the feature vector. This architecture includes six convolutional layers, three max pooling layers and two fully-connected layers. A rectified linear unit (ReLU) nonlinearity is applied to the output of every convolutional and fully-connected layer [67].

In our optimized version of this architecture, a window duration of 7 secs is used. The first convolutional layer filters the input of size of 70 × 22 × 26 using 16 kernels of size 3 × 3 with a stride of 1. The input feature vectors have a dimension of 26, while there are 22 EEG channels. The window length is 70 because the features are computed every 0.1 secs, or 10 times per second, and the window duration is 7 sec. These kernel sizes and strides were experimentally optimized [26].

The second convolutional layer filters its input using 16 kernels of size 3 × 3 with a stride of 1. The first max pooling layer takes as input the output of the second convolutional layer and applies a pooling size of 2 × 2. This process is repeated two times with kernels of size 32 and 64. Next, a fully-connected layer with 512 neurons is applied and the output is fed to a 2-way sigmoid function which produces a two-class decision. This two-class decision is the final label for the given epoch, which is 1 sec in duration. Neurologists usually review EEGs using 10 sec windows, so we attempt to use a similar amount of context in this system. Pattern recognition systems often subdivide the signal into small segments during which the signal can be considered quasi-stationary. A simple set of preliminary experiments determined that a reasonable tradeoff between computational complexity and performance was to split a 10 sec window, which is what neurologists use to view the data, into 1 sec epochs [40].

In our experiments, we found structures that are composed of two consecutive convolutional layers before a pooling layer perform better than structures with one convolutional layer before a pooling layer. Pooling layers decrease the dimensions of the data and thereby can result in a loss of information. Using two convolutional layers before pooling mitigates the loss of information. We find that using very small fields

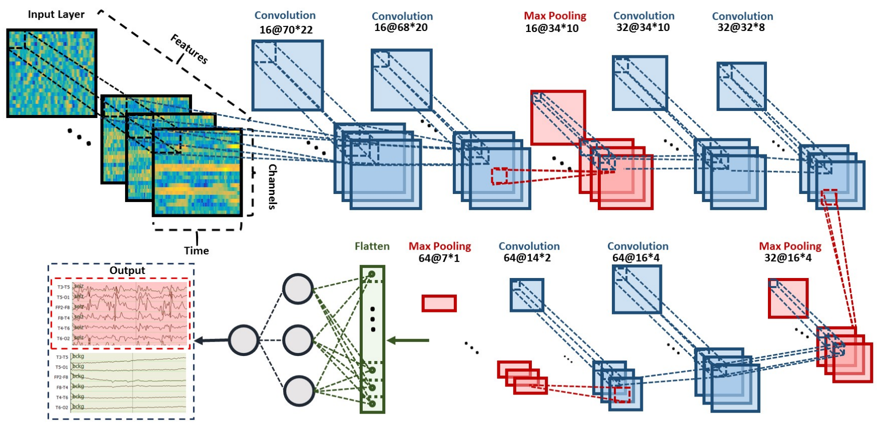

Fig. 4. A two-dimensional decoding of EEG signals using a CNN/MLP hybrid architecture

throughout the architecture (e.g., 3 x 3) performs better than larger fields (e.g. 5×5 or 7×7) in the first convolutional layer.

3.2 Augmenting CNNs with Deep Residual Learning

The depth of a CNN plays an instrumental role in its ability to achieve high performance [27, 28]. As many as thirteen layers are used for challenging problems such as speech and image recognition. However, training deeper CNN structures is more difficult since convergence and generalization become issues. Increasing the depth of CNNs, in our experience, tends to increase the error on evaluation dataset. As we add more convolutional layers, sensitivity first saturates and then degrades quickly. We also see an increase in the error on the training data when increasing the depth of a CNN, indicating that overfitting is actually not occurring. Such degradations in performance can be addressed by using a deep residual learning framework known as a ResNet [29]. ResNets introduce an "identity shortcut connection" that skips layers. Denoting the desired underlying mapping as $H(x)$, we map the stacked nonlinear layers using $F(x) = H(x) - x$, where x is the input. The original mapping is recast into $F(x) + x$. It can be shown that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping [29].

The deep residual learning structure mitigates two important problems: vanishing/exploding gradients and saturation of accuracy when the number of layers is increased. As the gradient is backpropagated to earlier layers, repeated multiplication of numbers less than one often makes the gradient infinitively small. Performance saturates and can rapidly degrade due to numerical precision issues. Our structure addresses these problems by reformulating the layers as learning residual functions with reference to the layer inputs instead of learning unreferenced functions.

An architecture for our ResNet approach is illustrated in Fig. 5. The shortcut connections between the convolutional layers make training of the model tractable by allowing information to propagate effectively through this very deep structure. The network consists of 6 residual blocks with two 2D convolutional layers per block. These convolutional layers are followed by a fully connected layer and a single dense neuron as the last layer. This brings the total number of layers in this modified CNN structure to 14. The 2D convolutional layers all have a filter length of (3, 3). The first 7 layers of this architecture have 32 filters while the last layers have 64 filters. We increment the number of filters from 32 to 64, since the initial layers represent generic features, while the deeper layers represent more detailed features. In other words, the richness of the data representations increases because each additional layer forms new kernels using combinations of the features from the previous layer.

Except for the first and last layers of the network, before each convolutional layer we apply a Rectified Linear Unit (ReLU) as an activation function [68]. ReLU is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value it returns that value (e.g., $f(x) = \max(0, x)$). To overcome the problem of overfitting in deep learning structures

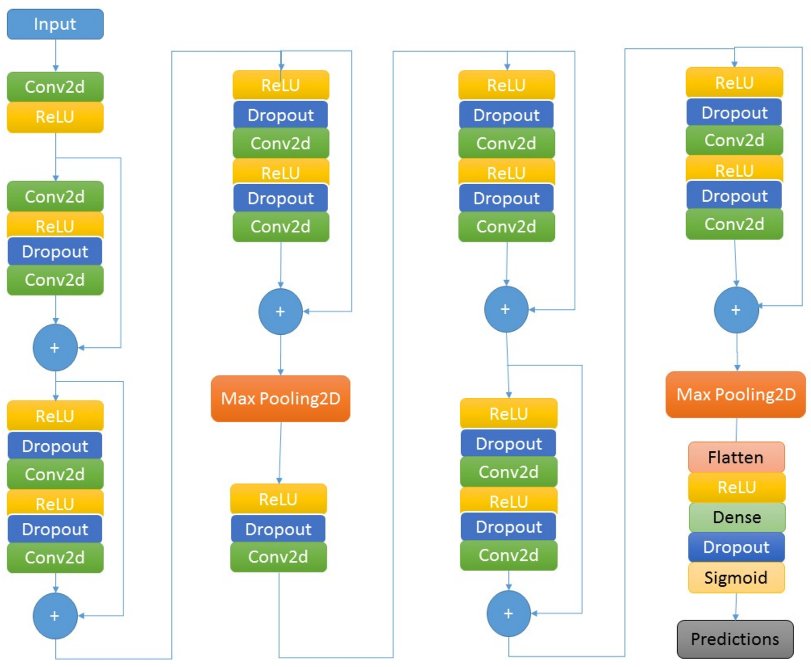

Fig. 5. A deep residual learning framework, ResNet, is shown.

with a large number of parameters, we use dropout [69] as our regularization method between the convolutional layers and after ReLU. Dropout is a regularization technique for addressing overfitting by randomly dropping units along with their connections from the deep learning structures during training. We use the Adam optimizer [70] which is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. After parameter tuning, we apply Adam optimization using the following parameters (according to the notation in their original paper): $\alpha = 0.00005, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$, and $decay = 0.0001$.

The deep learning systems described thus far have incorporated fully supervised training and discriminative models. Next, we introduce a generative deep learning structure based on convolutional neural networks that leverages unsupervised learning techniques. These are important for biomedical applications where large amounts of fully annotated data are difficult to find.

3.3 Unsupervised Learning

Machine learning algorithms can generally be split into two categories: generative and discriminative. A generative model learns the joint probability distribution of $P(X, Y)$ where X is an observable variable and Y is the target variable. These models learn the statistical distributions of the input data rather than simply classifying the data as one of C output classes. Hence the name, generative, since these methods learn to replicate the underlying statistics of the data. GMMs trained using a greedy clustering algorithm or HMMs trained using the Expectation Maximization (EM) algorithm [71] are well-known examples of generative models. A discriminative model, on the other hand, learns the conditional probability of the target Y, given an observation X, which we denote $P(Y|X)$ [72]. Support Vector Machines [73] and Maximum Mutual Information Estimation (MMIE) [74] are two well-known discriminative models.

Generative adversarial networks (GANs) [31] have emerged as a powerful learning paradigm technique for learning generative models for high-dimensional unstructured data. GANs use a game theory approach to find the Nash equilibrium between a generator and discriminator network [75]. A basic GAN structure consists of two neural networks: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. These two networks are trained simultaneously via an adversarial process. In this process, the generative network, G, transforms the input noise vector z to generate synthetic data $G(z)$. The training objective for G is to maximize the probability of D making a mistake about the source of the data.

The output of the generator is a synthetic EEG – data that is statistically consistent with an actual EEG but is fabricated entirely by the network. The second network, which is the discriminator, D, takes as input either the output of G or samples from real world data. The output of D is a probability distribution over possible input sources. The output of the discriminator in GAN determines if the signal is a sample from real world data or synthetic data from the generator.

The generative model, G, and the discriminative model, D, compete in a two-player minimax game with a value function, $V(G; D)$, in a way that D is trained to maximize the probability of assigning the correct label to both the synthetic and real data from G [31]. The generative model G is trained to fool the discriminator by minimizing $log(1 - D(G(z)))$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[log\, D(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z))] \, . \quad (4)$$

During the training process, our goal is to find a Nash equilibrium of a non-convex two-player game that minimizes both the generator and discriminator's cost functions [75].

A deep convolutional generative adversarial network (DCGAN) is shown in Fig. 6. The generative model takes 100 random inputs and maps them to a matrix with size of [21, 22, 250], where 21 is the window length (corresponding to a 21 sec duration), 22 is number of EEG channels and 250 is number of samples per sec. Recall, in our study, we resample all EEGs to a sample frequency of 250 Hz [40]. The generator is composed

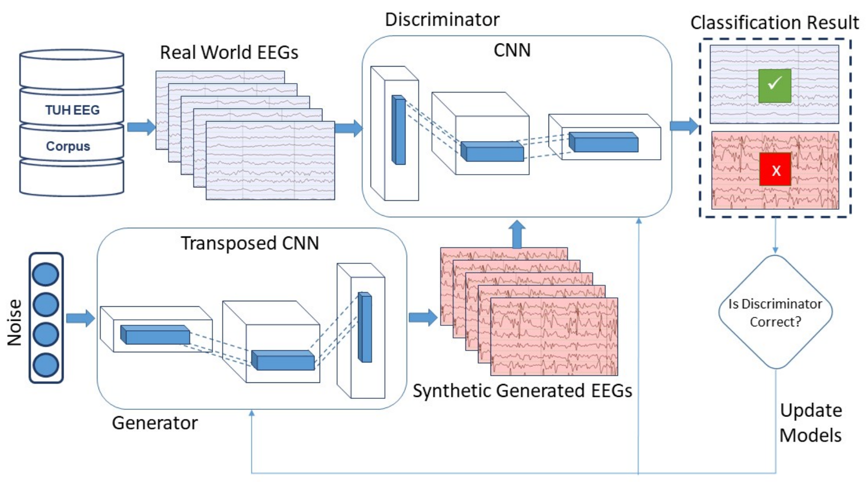

Fig. 6. An unsupervised learning architecture is shown that uses DCGANs

of transposed CNNs with upsamplers. Transposed convolution, also known as fractionally-strided convolution, can be implemented by swapping the forward and backward passes of a regular convolution [31]. We need transposed convolutions in the generators since we want to go in the opposite direction of a normal convolution. For example, in this case we want to compose the vector of [21, 22, 250] from 100 random inputs. Using transposed convolutional layers, we can transform feature maps to a higher-dimensional space. Leaky ReLUs [68] are used for the activation function and dropout layers are used for regularization. Adam is used as the optimizer and binary cross-entropy [76] is used as the loss function.

In this architecture, the discriminative model accepts vectors from two sources: synthetic data generators and real data (raw EEGs in this case). It is composed of strided convolutional neural networks [77]. Strided convolutional neural networks are like regular CNNs but with a stride greater than one. In the discriminator we replace the usual approach of convolutional layers with max pooling layers with strided convolutional neural networks. This is based on our observations that using convolutional layers with max pooling makes the training of DCGAN unstable. This is due to the fact that using strided convolutional layers, the network learns its own spatial downsampling, and convolutional layers with max pooling tend to conflict with striding.

Finding the Nash equilibrium, which is a key part of the GAN approach, is a challenging problem that impacts convergence during training. Several recent studies address the instability of GANs and suggest techniques to increase the training stability of GANs [77]. We conducted a number of preliminary experiments and determined that these techniques were appropriate:

In the discriminator:

- pretraining of the discriminator;
- one-sided label smoothing;
- eliminating fully connected layers on top of convolutional features;
- replacing deterministic spatial pooling functions (such as max pooling) with strided convolutions.

In the generator:

- using an ReLU activation for all layers except for the output;
- normalizing the input to [-1, 1] for the discriminator;
- using a $tanh()$ activation in the last layer except for the output;
- using leaky ReLU activations in the discriminator for all layers except for the output;
- freezing the weights of discriminator during adversarial training process;
- unfreezing weights during discriminative training;
- eliminating batch normalization in all the layers of both the generator and discriminator.

The GAN approach is attractive for a number of reasons including creating an opportunity for data augmentation. Data augmentation is common in many state-of-the-art deep learning systems today [78], allowing the size of the training set to be increased as well as exposing the system to previously unseen patterns during training.

4 LEARNING TEMPORAL DEPENDENCIES

The duration of events such as seizures can vary dramatically from a few seconds to minutes. Further, neurologists use significant amounts of temporal context and adaptation in manually interpreting EEGs. They are very familiar with their patients and often can identify the patient by examining the EEG signal, especially when there are certain types of anomalous behaviors. In fact, they routinely use the first minute or so of an EEG to establish baseline signal conditions [65], or normalize their expectations, so that they can more accurately determine anomalous behavior. Recurrent neural networks (RNN), which date back to the late 1980's [79], have been proposed as a way to learn such dependencies. Prior to this, successful systems were often based on approaches such as hidden Markov models, or used heuristics to convert frame-level output into longer-term hypotheses. In this section, we introduce several architectures that model long-term dependencies.

4.1 Integration of Incremental Principal Component Analysis with LSTMs

In the HMM/SdA structure proposed in Section 2.2, PCA was used prior to SdA for dimensionality reduction. Unlike HMM/SdA, applying LSTM networks directly to features requires more memory efficient approaches than PCA, or the memory requirements of the network can easily exceed the available computational resources (e.g., low-cost graphics processing units such as the Nvidia 1080ti have limited amount of memory – typically 8 Gbytes). Incremental principal components analysis (IPCA) is an effective technique for dimensionality reduction [61, 80]. This algorithm is often more memory efficient than PCA. IPCA has constant memory complexity proportional to the batch size, and it enables use of large datasets without a need to load the entire file or dataset into memory. IPCA builds a low-rank approximation for the input data using an amount of memory which is independent of the number of input data samples. It is still dependent on the dimensionality of the input data features but allows more direct control of memory usage by changing the batch size.

In PCA, the first k dominant principal components, $y_1(n), y_2(n), \ldots, y_k(n)$, are computed directly from the input, $x(n)$ as follows:

```
$For\ n = 1, 2, \ldots, do\ the\ following:$
    1) $x_1(n) = x(n).$
    2) $For\ i = 1, 2, \ldots, \min(k, n), do:$
        $a)\ if\ i = n, initialize\ the\ i\ principal\ component\ as\ y_i(n) = x_i(n);$
        $b)\ otherwise\ compute:$
```

$$y_i(n) = \frac{\mathrm{n-1-p}}{n} y_i(n-1) + \frac{1+p}{n} x_i(n) x_i^T(n) \frac{y_i(n-1)}{||y_i(n-1)||}, \tag{5}$$

$$x_{i+1}(n) = x_i(n) - x_i^T(n) \frac{y_i(n)}{||y_i(n)||} \frac{y_i(n)}{||y_i(n)||}, \tag{6}$$

where the positive parameter p is called the amnesic parameter. Typically, p ranges from 2 to 4. Then the eigenvector and eigenvalues are given by:

$$e_i = \frac{y_i(n)}{||y_i(n)||} \ and\ \lambda_i = ||y_i(n)||\,. \tag{7}$$

In Fig. 7, we present an architecture that integrates IPCA and LSTM [26]. In this system, samples are converted to features and the features are delivered to an IPCA layer that performs spatial context analysis and dimensionality reduction. The output of the IPCA layer is delivered to a one-layer LSTM for seizure classification task. The input to the IPCA layer is a vector whose dimension is the product of the number of channels, the number of features per frame and the number of frames of context. Preliminary experiments have shown that 7 seconds of temporal context performs well.

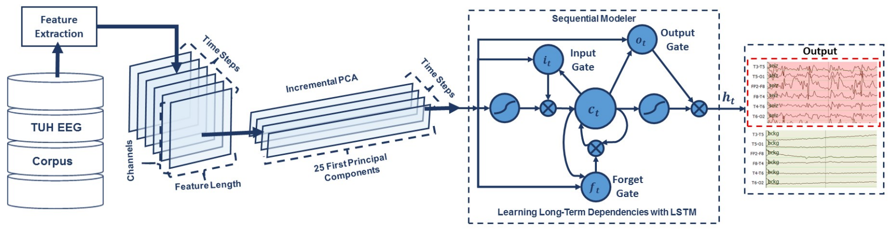

Fig. 7. An architecture that integrates IPCA and LSTM

The corresponding dimension of the vector input to IPCA is 22 channels × 26 features × 7 seconds × 10 frames/second, or a total of 4004 elements. A batch size of 50 is used and the dimension of its output is 25 elements per frame at 10 frames/second. In order to learn long-term dependencies, one LSTM with a hidden layer size of 128 and batch size of 128 is used along with Adam optimization and a cross-entropy loss function.

4.2 End-to-End Sequence Labeling Using Deep Architectures

In machine learning, sequence labeling is defined as assigning a categorial label to each member of a sequence of observed values. In automatic seizure detection, we assign one of two labels: seizure or non-seizure. This decision is made every epoch, which is typically a 1 sec interval. The proposed structures are trained in an end-to-end fashion, requiring no pre-training and no pre-processing, beyond the feature extraction process that was explained in Section 2.1. For example, for an architecture composed of a combination of CNN and LSTM, we do not train CNN independently from LSTM, but we train both jointly. This is challenging because there are typically convergence issues when attempting this.

In Fig. 8, we integrate 2D CNNs, 1-D CNNs and LSTM networks, which we refer to as CNN/LSTM, to better exploit long-term dependencies [26]. Note that the way that we handle data in CNN/LSTM is different from the CNN/MLP system presented in Fig. 4. The input EEG features vector sequence can be thought of as being composed of frames distributed in time where each frame is an image of width (W) equal to the length of a feature vector. The height (H) equals the number of EEG channels and the number of image channels (N) equals one. The input to the network consists of T frames where T is equal to the window length multiplied by the number of frames per second. In our optimized system, where features are available 10 times per second, a window duration of 21 seconds is used. The first 2D convolutional layer filters 210 frames (T = 21 × 10) of EEGs distributed in time with a size of 26 × 22 × 1 (W = 26, H = 22,

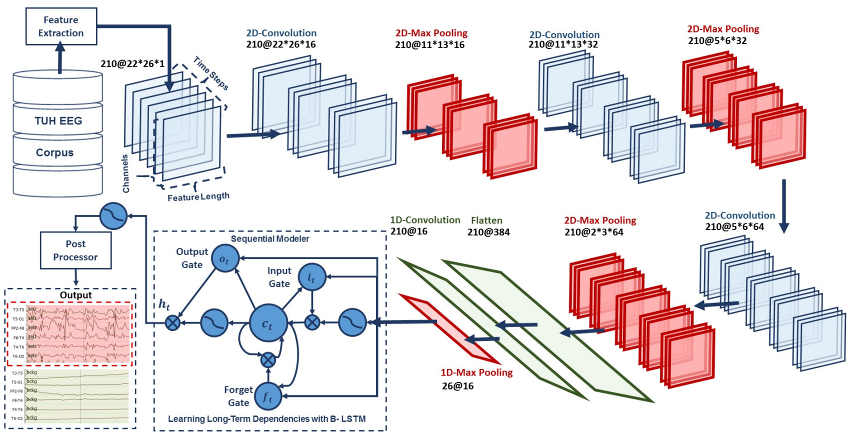

Fig. 8. A deep recurrent convolutional architecture

N = 1) using 16 kernels of size 3 × 3 with a stride of 1. The first 2D max pooling layer takes as input a vector which is 260 frames distributed in time with a size of 26 × 22 × 16 and applies a pooling size of 2 × 2. This process is repeated two times with two 2D convolutional layers with 32 and 64 kernels of size 3 × 3 respectively and two 2D max pooling layers with a pooling size 2 × 2.

The output of the third max pooling layer is flattened to 210 frames with a size of 384 × 1. Then a 1D convolutional layer filters the output of the flattening layer using 16 kernels of size 3 which decreases the dimensionality in space to 210 × 16. Next, we apply a 1D max pooling layer with a size of 8 to decrease the dimensionality to 26 × 16. This is the input to a deep bidirectional LSTM network where the dimensionality of the output space is 128 and 256. The output of the last bidirectional LSTM layer is fed to a 2-way sigmoid function which produces a final classification of an epoch. To overcome the problem of overfitting and force the system to learn more robust features, dropout and Gaussian noise layers are used between layers [69]. To increase nonlinearity, Exponential Linear Units (ELU) are used [81]. Adam is used in the optimization process along with a mean squared error loss function.

More recently, Chung et al. [33] proposed another type of recurrent neural network, known as a gated recurrent unit (GRU). A GRU architecture is similar to an LSTM but without a separate memory cell. Unlike LSTM, a GRU does not include output activation functions and peep hole connections. It also integrates the input and forget gates into an update gate to balance between the previous activation and the candidate activation. The reset gate allows it to forget the previous state. It has been shown that the performance of a GRU is on par with an LSTM, but a GRU can be trained faster [26]. The architecture is similar to that Fig. 8, but we simply replace LSTM with GRU, in a way that the output of 1D max pooling is the input to a GRU where the dimensionality of the output space is 128 and 256. The output of the last GRU is fed to a 2-way sigmoid function which produces a final classification of an epoch. These two approaches, LSTM and GRU, are evaluated as part of a hybrid architecture that integrates CNNs with RNNs [82].

4.3 Temporal Event Modeling Using LSTMs

A final architecture we wish to consider is a relatively straightforward variation of an LSTM network. LSTMs are a special type of recurrent neural network which contains forget and output gates to control the information flow during its recurrent passes. LSTM networks have proven to be outperform conventional RNNs, HMMs and other sequence learning methods in numerous applications such as speech recognition and handwriting recognition [83, 84]. Our first implementation of LSTM was a hybrid network of both HMM and LSTM networks. A block diagram of HMM/LSTM system is shown in Fig. 9. Similar to the HMM/SdA model discussed before, the input to the second layer of the system, which is the first layer of LSTMs, is a vector of dimension 2 × 22 × window length. We use PCA to reduce the dimensionality of the input vector to 20 and pass it to the LSTM model. A window size of 41 secs (41 epochs at 1 sec per epoch) is used for a 32-node single hidden layer LSTM network. The final layer uses a

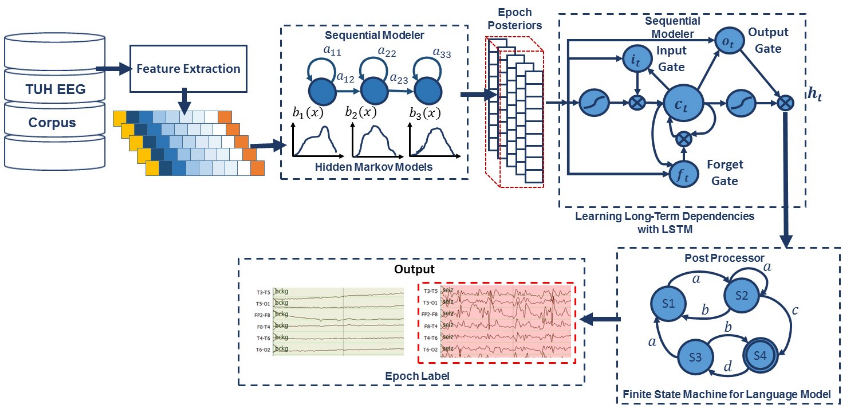

Fig. 9. A hybrid architecture that integrates HMM and LSTM.

dense neuron with a sigmoid activation function. The parameters of the models are optimized to minimize the error using a cross-entropy loss function and Adam [70].

Next, we use a 3-layer LSTM network model. Identification of a seizure event is done based on an observation of a specific type of epileptiform activity called "spike and wave discharges" [85]. The evolution of these activities across time helps identify a seizure event. These events can be observed on individual channels. Once observed, the seizures can be confirmed based on their focality, signal energy and its polarity across spatially close channels. The architecture is shown in Fig. 10.

In the preprocessing step, we extract a 26-dimensional feature vector for an 11-frame context centered around the current frame. The output dimensionality for each frame is 10 x 26 (left) + 26 (center) + 10 x 26 (right) = 546. The static LSTM cells are used with a fixed batch size of 64 and a window size of 7 seconds. The data is randomly split into subsets where 80% is used for training and 20% is used for cross-validation during optimization. The features are normalized and scaled down to a range of [0, 1] on a file

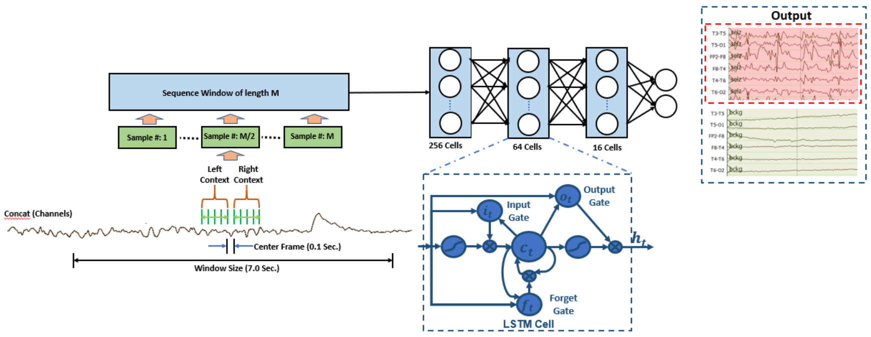

Fig. 10. A channel-based long short-term memory (LSTM) architecture

basis, which helps the gradient descent algorithm (and its variants) to converge much faster [86]. Shuffling was performed on batches to avoid training biases.

The network includes 3 LSTM layers with (256, 64, 16) hidden layers followed by a 2-cell dense layer. The activation function used for all LSTM layers is a hyperbolic tangent function, tanh(), except for the final layer, which uses a softmax function to compress the range of output values to [0,1] so they resemble posterior probabilities. A cross-entropy function is used for calculating loss. Stochastic gradient descent with Nesterov momentum is used for optimization. Nesterov momentum attempts to increase the speed of training by introducing a momentum term based on accumulated gradients of its previous steps and a correction term in the direction of the current gradient [87]. This tends to reduce the amount of overshoot during optimization.

The optimization is performed on the training data at a very high learning rate of 1.0 for the first five epochs. Cross-validation is performed after each epoch. After five epochs, if the cross-validation loss stagnates for three consecutive epochs (referred to as "patience = 3"), learning rates are halved after each iteration until it anneals to zero. If the model fails to show consistent performance on a cross-validation set, then it reverts to the previous epoch's weights and restarts training until convergence. This method helps models avoid overfitting on the training data as long as the training and cross-validation sets are equally diverse.

The outputs of the models are fed to a postprocessor which is described in more detail in Section 5. This postprocessor is designed based on domain knowledge and observed system behavior to remove spurious and misleading detections. This is implemented to incorporate spatial context. The postprocessor sets a threshold for hypothesis confidence, the minimum number of channels for target event detection and a duration constraint which must be satisfied for detection. For example, if multiple channels consistently detected spike and wave discharges in the same 9-second interval, this event would be permitted as a valid output. Outputs from a fewer number of channels or over a smaller duration of time would be suppressed.

We have now presented a considerable variety of deep learning architectures. It is difficult to predict which architecture performs best on a given task without extensive experimentation. Hence, in the following section, we review a wide-ranging study of how these architectures perform on the TUSZ seizure detection task.

5 EXPERIMENTATION

Machine learning is at its core an experimental science when addressing real-world problems of scale. Real world data is complex and poses many challenges that require a wide variety of technologies to solve and can mask the benefits of one specific algorithm. Therefore, it is important that a rigorous evaluation paradigm be used to guide architecture decisions. In this chapter, we are focusing on the TUSZ Corpus because it is a very comprehensive dataset and it offers a very challenging task.

The evaluation of machine learning algorithms in biomedical fields for applications involving sequential data lacks standardization. Common quantitative scalar evaluation metrics such as sensitivity and specificity can often be misleading depending on the requirements of the application. Evaluation metrics must ultimately reflect the needs of users yet be sufficiently sensitive to guide algorithm development. Feedback from critical care clinicians who use automated event detection software in clinical applications has been overwhelmingly emphatic that a low false alarm rate, typically measured in units of the number of errors per 24 hours, is the single most important criterion for user acceptance. Though using a single metric is not often as insightful as examining performance over a range of operating conditions, there is a need for a single scalar figure of merit. Shah et al. [88] discuss the deficiencies of existing metrics for a seizure detection task and propose several new metrics that offer a more balanced view of performance. In this section, we compare the architectures previously described using one of these measures, the Any-Overlap Method (OVLP). We also provide detection error tradeoff (DET) curves [89].

5.1 Evaluation Metrics

Researchers in biomedical fields typically report performance in terms of sensitivity and specificity [90]. In a two-class classification problem such as seizure detection, we can define four types of errors:

- True Positives (TP): the number of 'positives' detected correctly
- True Negatives (TN): the number of 'negatives' detected correctly
- False Positives (FP): the number of 'negatives' detected as 'positives'
- False Negatives (FN): the number of 'positives' detected as 'negatives'

Sensitivity (TP/(TP+FN)) and specificity (TN/(TN+FP)) are derived from these quantities. There are a large number of auxiliary measures that can be calculated from these four basic quantities that are used extensively in the literature. For example, in information retrieval applications, systems are often evaluated using accuracy ((TP+TN)/(TP+FN+TN+FP)), precision (TP/(TP+FP)), recall (another term for sensitivity) and F1 score ((2•Precision•Recall)/(Precision + Recall)). However, none of these measures address the time scale on which the scoring must occur or how you score situations where the mapping of hypothesized events to reference events is ambiguous. These kinds of decisions are critical in the interpretation of scoring metrics such as sensitivity for many sequential decoding tasks such as automatic seizure detection [89, 91, 92].

In some applications, it is preferable to score every unit of time. With multichannel signals, such as EEGs, scoring for each channel for each unit of time might be appropriate since significant events such as seizures occur on a subset of the channels present in the signal. However, it is more common in the literature to simply score a summary decision per unit of time, such as every 1 sec, that is based on an aggregation of the per-channel inputs (e.g., a majority vote). We refer to this type of scoring as epoch-based [93, 94]. An alternative, that is more common in speech and image recognition applications, is term-based [50, 95], in which we consider the start and stop time of the event, and each event identified in the reference annotation is counted once. There are fundamental differences between the two conventions. For example, one event containing many epochs will count more heavily in an epoch-based scoring scenario. Epoch-based scoring generally weights the duration of an event more heavily since each unit of time is assessed independently.

Term-based metrics score on an event basis and do not count individual frames. A typical approach for calculating errors in term-based scoring is the Any-Overlap Method (OVLP) [92]. This approach is illustrated in Fig. 11. TPs are counted when the hypothesis overlaps with the reference annotation. FPs correspond to situations in which a hypothesis does not overlap with the reference.

OVLP is a more permissive metric that tends to produce much higher sensitivities. If an event is detected in close proximity to a reference event, the reference event is considered correctly detected. If a long event in the reference annotation is detected as multiple shorter events in the hypothesis, the reference event is also considered correctly detected. Multiple events in the hypothesis annotation corresponding to the same event in the reference annotation are not typically counted as FAs. Since the FA rate is a very important measure of performance in critical care applications, this is another cause for concern. However, since OVLP metric is the most popular choice in the neuroengineering community, we present our results in terms of OVLP.

Note that results are still reported in terms of sensitivity, specificity and false alarm rate. But, as previously mentioned, how one measures the errors that contribute to these measures is open for interpretation. Shah et al. [92] studied this problem extensively and showed that many of these measures correlate and are not significantly different in terms of the rank ordering and statistical significance of scoring differences for the

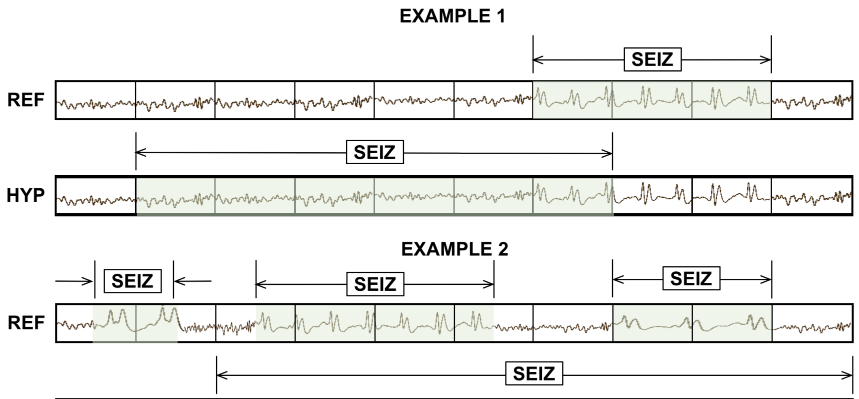

Fig. 11. OVLP scoring is very permissive about the degree of overlap between the reference and hypothesis. For example, in Example 1, the TP score is 1 with no false alarms. In Example 2, the system detects 2 out of 3 seizure events, so the TP and FN scores are 2 and 1 respectively.

TUSZ task. We provide a software package that allows researchers to replicate our metrics and reports on many of the most popular metrics [91].

5.2 Postprocessing with Heuristics Improves Performance

Because epoch-based scoring produces a hypothesis every epoch (1 sec in this case), and these are scored against annotations that are essentially asynchronous, there is an opportunity to improve performance by examining sequences of epochs and collapsing multiple events into a single hypothesis. We have experimented with heuristic approaches to this as well as deep learning-based approaches and have found no significant advantage for the deep learning approaches. As is well known in machine learning research, a good heuristic can be very powerful. We apply a series of heuristics, summarized in Fig. 12, to improve performance. These heuristics are very important in reducing the false alarm rate to an acceptable level.

The first heuristic we apply is a popular method that focuses on a model's confidence in its output. Probabilistic filters [96] are implemented to only consider target events which are detected above a specified probability threshold. This method tends to suppress spurious long duration events (e.g., slowing) and extremely short duration events (e.g., muscle artifacts). This decision function is applied on the seizure (target) labels only. We compare each seizure label's posterior with the threshold value. If the posterior is above the threshold, the label is kept as is; otherwise, it is changed to the non-seizure label, which we denote "background."

Our second heuristic was developed after performing extensive error analysis. The most common types of errors we observed were false detections of background events as seizures (FPs) which tend to occur in bursts. Usually these erroneous bursts occur

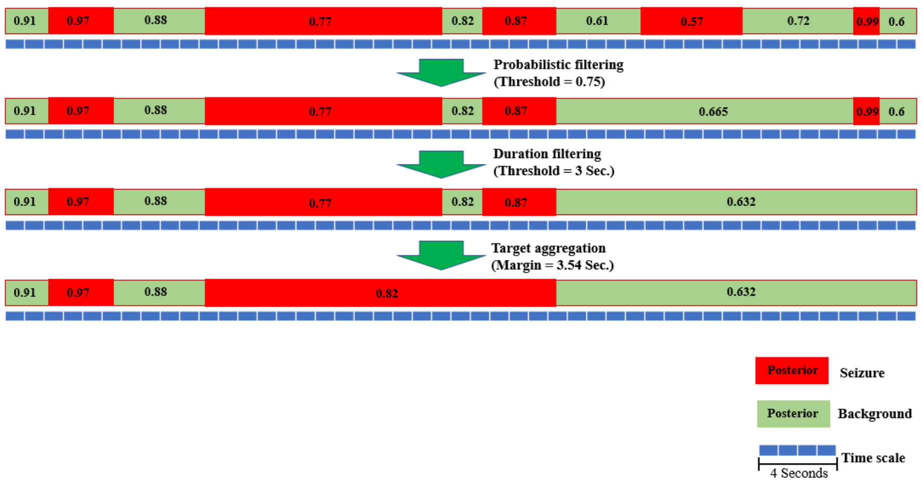

Fig. 12. An illustration of the postprocessing algorithms used to reduce the FA rate

for a very small duration of time (e.g., 3 to 7 seconds). To suppress these, any seizure event whose duration is below a specified threshold is automatically considered as a non-seizure, or background, event.

Finally, we also implement a smoothing method that collapses sequences of two seizure events separated by a background event into one long seizure event. This is typically used to eliminate spurious background events. If seizures are observed in clusters separated by small intervals of time classified as background events, these isolated events are most likely part of one longer seizure event. In this method, we apply a nonlinear function that computes a pad time to extend the duration of an isolated event. If the modified endpoint of that event overlaps with another seizure event, the intervening background event is eliminated. We used a simple regression approach to derive a quadratic function that produces a padding factor: $w(x) = -0.0083d^2 + 0.45d - 0.66$, were d is the duration of the event. This method tends to reduce isolated background events when they are surrounding by seizure events, thereby increasing the specificity.

The combination of these three postprocessing methods tends to decrease sensitivity slightly and reduce false alarms by two orders of magnitude, so their impact is significant. The ordering in which these methods is applied is important. We apply them in the order described above to achieve optimal performance.

5.3 A Comprehensive Evaluation of Hybrid Approaches

A series of experiments was conducted to optimize the feature extraction process. These are described in detail in [40]. Subsequent attempts to replace feature extraction with deep learning-based approaches have resulted in a slight degradation in performance. A reasonable tradeoff between computational complexity and performance was to split the 10 sec window, popular with neurologists who manually interpret these waveforms, into 1 sec epochs, and to further subdivide these into 0.1 sec frames. Hence, features were computed every 0.1 sec using a 0.2 sec overlapping analysis window. The output of the feature extraction system is 22 channels of data, where in each channel, a feature vector of dimension 26 corresponds to every 0.1 secs. This type of analysis is very compatible with the way HMM systems operate, so it was a reasonable starting point for this work.

We next evaluated several architectures using these features as inputs on TUSZ. These results are presented in Table 2. The related DET curve is illustrated Fig. 13. An expanded version of the DET curve in Fig. 13 that compares the performance of these architectures in a region of the DET curve where the false positive rate, also known as the false alarm (FA) rate, is low is presented in Fig. 14. Since our focus is achieving a low false alarm rate, behavior in this region of the DET curve is very important. As previously mentioned, these systems were evaluated using the OVLP method, though results are similar for a variety of these metrics.

It is important to note that the accuracy reported here is much lower than what is often published in the literature on other seizure detection tasks. This is due to a

Table 2. Performance of the proposed architectures on TUSZ

System	Sensitivity	Specificity	FA/24 Hrs.
HMM	30.32%	80.07%	244
HMM/SdA	35.35%	73.35%	77
HMM/LSTM	30.05%	80.53%	60
IPCA/LSTM	32.97%	77.57%	73
CNN/MLP	39.09%	76.84%	77
CNN/GRU	30.83%	91.49%	21
ResNet	30.50%	94.24%	13
CNN/LSTM	30.83%	97.10%	6
Channel-Based LSTM	39.46%	95.20%	11

combination of factors including (1) the neuroscience community has favored a more permissive method of scoring that tends to produce much higher sensitivities and lower false alarm rates; and (2) TUSZ is a much more difficult task than any corpus previously released as open source. The evaluation set was designed to be representative of common clinical issues and includes many challenging examples of seizures. We have achieved much higher performance on other publicly available tasks, such as the Children's Hospital of Boston MIT (CHB-MIT) Corpus, and demonstrated that the performance of these techniques exceeds that of published or commercially-available technology. TUSZ is simply a much more difficult task and one that better represents the clinical challenges this technology faces.

Also, note that the HMM baseline system, which is shown in the first row of Table **2**, and channel-based LSTM, which is shown in the last row of Table 2, operate on each channel independently. The other methods consider all channels simultaneously by

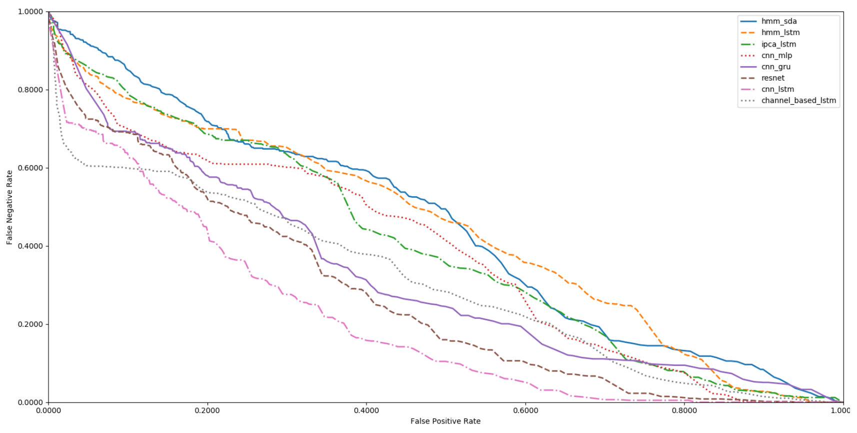

Fig. 13. A DET curve comparison of the proposed architectures on TUSZ.

using a supervector that is a concatenation of the feature vectors for all channels. The baseline HMM system only classifies epochs (1 sec in duration) using data from within that epoch. It does not look across channels or across multiple epochs when performing epoch-level classification.

From Table 2 we can see that adding a deep learning structure for temporal and spatial analysis of EEGs can decrease the false alarm rate dramatically. Further, by comparing the results of HMM/SdA with HMM/LSTM, we find that a simple one-layer LSTM performs better than 3 layers of SdA due to LSTM's ability to explicitly model long-term dependencies. Note that in this case the complexity and training time of these two systems is comparable.

The best overall systems shown in Table 2 are CNN/LSTM and channel-based LSTM. CNN/LSTM is a doubly deep recurrent convolutional structure that models both spatial relationships (e.g., cross-channel dependencies) and temporal dynamics (e.g., spikes). For example, CNN/LSTM does a much better job rejecting artifacts that are easily confused with spikes because these appear on only a few channels, and hence can be filtered based on correlations between channels. The depth of the convolutional network is important since the top convolutional layers tend to learn generic features while the deeper layers learn dataset specific features. Performance degrades if a single convolutional layer is removed. For example, removing any of the middle convolutional layers results in a loss of about 4% in the sensitivity. However, it is important to note that the computational complexity of the channel-based systems is significantly higher than the systems that aggregate channel-based features into a single vector, since the channel-based systems are decoding each channel independently.

As shown in Fig. 13 and Fig. 14, we find that CNN/LSTM has a significantly lower FA rate than CNN/GRU. We speculate that this is due to the fact that while a GRU unit controls the flow of information like the LSTM unit, it does not have a memory unit. LSTMs can remember longer sequences better than GRUs. Since seizure detection

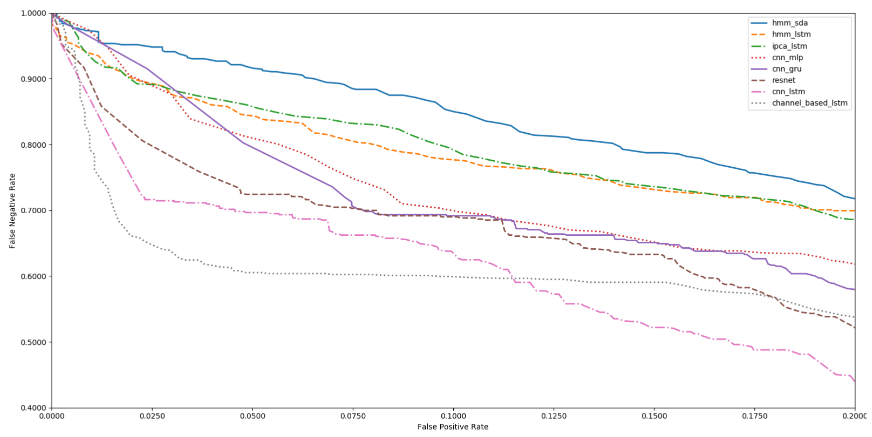

Fig. 14. An expanded comparison of performance in a region where the FP rate is low.

requires modeling long distance relationships, we believe this explains why there is a difference in performance between the two systems.

The time required for training for CNN/GRU was 10% less than CNN/LSTM. The training time of these two systems is comparable since most of the cycles are spent training the convolutional layers. We also observe that the ResNet structure improves the performance of CNN/MLP, but the best overall system is still CNN/LSTM.

We have also conducted an open-set evaluation of the best systems, CNN/LSTM and channel-based LSTM, on a completely different corpus – DUSZ. These results are shown in Table 3. A DET curve is shown in Fig. 15. This is an important evaluation because none of these systems were exposed to DUSZ data during training or development testing. Parameter optimizations were performed only on TUSZ data. As can be seen, at high FA rates, performance between the two systems is comparable. At low FA rates, however, CNN/LSTM performance on TUSZ is lower than on DUSZ. For channel-based LSTM, in the region of low FA rate, performance on TUSZ and DUSZ is very similar. This is reflected by the two middle curves in Fig. 15. The differences in performance for channel-based LSTM when the data changes are small. However, for CNN/LSTM, which gives the best overall performance on TUSZ, performance decreases rapidly on DUSZ. Recall that we did not train these systems on DUSZ – this is true open set testing. Hence, we can conclude in this limited study that channel-based LSTM generalized better than the CNN/LSTM system.

Table 3. A comparison of several CNN and LSTM architectures on DUSZ

System	Data	Sensitivity	Specificity	FA/24 Hrs.
CNN/LSTM	TUSZ	30.83%	97.10%	6
CNN/LSTM	DUSZ	33.71%	70.72%	40
Channel-Based LSTM	TUSZ	39.46%	95.20%	11
Channel-Based LSTM	DUSZ	42.32%	86.93%	14

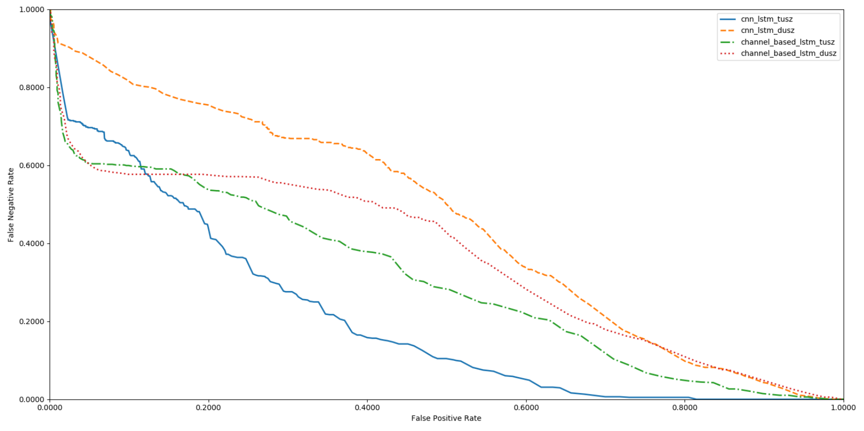

Fig. 15. Performance of CNN/LSTM and channel-based LSTM on TUSZ and DUSZ.

5.4 Optimization of Core Components

Throughout these experiments, we observed that the choice of optimization method had a considerable impact on performance. The CNN/LSTM system was evaluated using a variety of optimization methods, including Stochastic gradient descent (SGD) [70], RMSprop [97], Adagrad [98], Adadelta [99], Adam [70], Adamax [70] and Nadam [100]. These results are shown in Table 4. The best performance is achieved with Adam, a learning rate of $\alpha = 0.0005$, a learning rate decay of 0.0001, exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the moment estimates and a fuzz factor of $\epsilon = 10^{-8}$. The parameters follow the notation described in [70]. Table 4 also illustrates that Nadam delivers comparable performance to Adam. Adam combines the advantages of AdaGrad which works well with sparse gradients, and RMSProp which works well in non-stationary settings.

Similarly, we evaluated our CNN/LSTM using different activation functions, as shown in Table 5. ELU delivers a small but measurable increase in sensitivity, and more importantly, a reduction in false alarms. The ELU activation function is defined as:

$$\mathrm{f(x)} = \begin{cases} x & x > 0 \\ \alpha.(e^x - 1) & x \le 0 \end{cases}, \tag{8}$$

where α is slope of negative section. The derivative of the ELU activation function is:

$$\acute{\mathrm{f}}\mathrm{(x)} = \begin{cases} 1 & x > 0 \\ \alpha.e^x & x \le 0 \end{cases}. \tag{9}$$

The ReLU activation function is defined as:

$$\mathrm{f(x)} = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases}. \tag{10}$$

The corresponding derivative is:

$$\acute{\mathrm{f}}\mathrm{(x)} = \begin{cases} 1 & x > 0 \\ 0 & x \le 0 \end{cases}. \tag{11}$$

Table 4. Comparison of optimization algorithms

System	Sensitivity	Specificity	FA/24 Hrs.
SGD	23.12%	72.24%	44
RMSprop	25.17%	83.39%	23
Adagrad	26.42%	80.42%	31
Adadelta	26.11%	79.14%	33
Adam	30.83%	97.10%	6
Adamax	29.25%	89.64%	18
Nadam	30.27%	92.17%	14

Table 5. A comparison of activation functions

System	Sensitivity	Specificity	FA/24 Hrs.
Linear	26.46%	88.48%	25
Tanh	26.53%	89.17%	21
Sigmoid	28.63%	90.08%	19
Softsign	30.05%	90.51%	18
ReLU	30.51%	94.74%	11
ELU	30.83%	97.10%	6

ELU is very similar to ReLU except for negative inputs. ReLUs and ELUs accelerate learning by decreasing the gap between the normal gradient and the unit natural gradient [81]. ELUs push the mean towards zero but with a significantly smaller computational footprint. In the region where the input is negative ($x < 0$), since an ReLU's gradient is zero, the weights will not get adjusted. Those neurons which connect into that state will stop responding to variations in error or input. This is referred to as the dying ReLU problem. But unlike ReLUs, ELUs have a clear saturation plateau in their negative region, allowing them to learn a more robust and stable representation.

Determining the proper initialization strategy for the parameters in the model is part of the difficulty in training. Hence, we investigated a variety of initialization methods using the CNN/LSTM structure introduced in Fig. 8. These results are presented in Table 6. The related DET curve is illustrated in Fig. 16. In our experiments, we observed that proper initialization of weights in a convolutional recurrent neural network is critical to convergence. For example, initialization of tensor values to zero or one completely stalled the convergence process. Also, as we can see in Table 6, the FA rate of the system in the range of 30% sensitivity can change from 7 to 40, for different initialization methods. This decrease in performance and deceleration of convergence arises because some initializations can result in the deeper layers receiving inputs with small variances, which in turn slows down back propagation, and retards the overall convergence process.

Table 6. A comparison of FA rates on a CNN/LSTM system

System	Sensitivity	Specificity	FA/24 Hrs.
Orthogonal	30.8%	96.9%	7
Lecun Uniform	30.3%	96.5%	8
Glorot Uniform	31.0%	94.2%	13
Glorot Normal	29.5%	92.4%	18
Variance Scaling	31.8%	92.1%	19
Lecun Normal	31.8%	92.1%	19
He Normal	31.3%	91.1%	22
Random Uniform	30.2%	90.0%	25
Truncated Normal	31.6%	87.8%	31
He Uniform	29.2%	85.1%	40

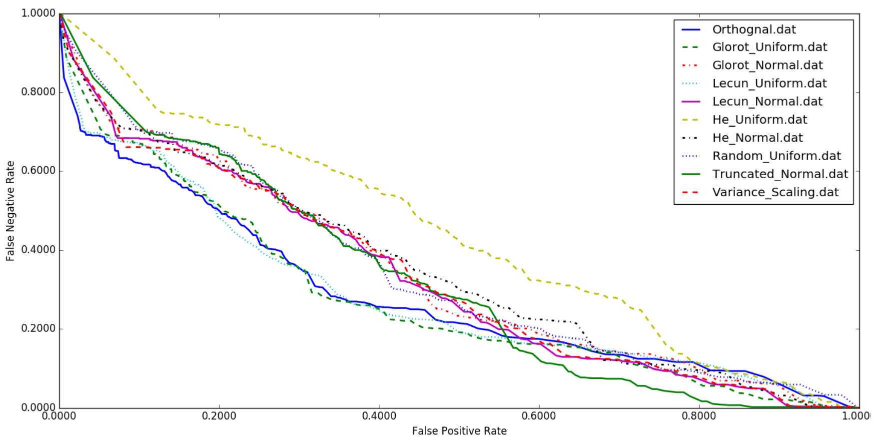

Fig. 16. A comparison of different initialization methods for CNN/LSTM

Best performance is achieved using orthogonal initialization [101]. This method is a simple yet effective way of combatting exploding and vanishing gradients. In orthogonal initialization, the weight matrix is chosen as a random orthogonal matrix, i.e., a square matrix W for which $W^T W = I$. Typically, the orthogonal matrix is obtained from the QR decomposition of a matrix of random numbers drawn from a normal distribution. Orthogonal matrices preserve the norm of a vector and their eigenvalues have an absolute value of one. This means that no matter how many times we perform repeated matrix multiplication, the resulting matrix doesn't explode or vanish. Also, in orthogonal matrices, columns and rows are all orthonormal to one another, which helps the weights to learn different input features. For example, if we apply orthogonal initialization on a CNN architecture, in each layer, each channel has a weight vector that is orthogonal to the weight vectors of the other channels.

Overfitting is a serious problem in deep neural nets with many parameters. We have explored five popular regularization methods to address this problem. The techniques collectively known as L1, L2 and L1/L2 [102] prevent overfitting by adding a regularization term to the loss function. The L1 regularization technique, also known as Lasso regression, is defined as adding the sum of weights to the loss function:

$$Cost\ Function = Loss\ Function + \lambda \sum_{i=1}^{k} |w_i| \,, \tag{12}$$

where w is the weight vector and λ is a regularization parameter. The L2 technique, also known as ridge regression, is defined as adding the sum of the square of the weights to the loss function:

$$Cost\ Function = Loss\ Function + \lambda \sum_{i=1}^{k} w_i^2 \,. \tag{13}$$

The L1/L2 technique is a combination of both techniques:

$$Cost\ Function = Loss\ Function + \lambda \sum_{i=1}^{k} |w_i| + \lambda \sum_{i=1}^{k} w_i^2 \,. \tag{14}$$

In an alternative approach, we used dropout to prevents units from excessively co-adapting by randomly dropping units and their connections from the neural network during training.

We also studied the impact of introducing zero-centered Gaussian noise to the network. In this regularization method, which is considered a random data augmentation method [103], we add zero-centered Gaussian noise with a standard deviation of 0.2 to all hidden layers in the network as well as the visible or input layer. The results of these experiments are presented in Table 7 along with a DET curve in Fig. 17. While L1/L2 regularization has the best overall performance, in the region where FA rates are low, the dropout method delivers a lower FA rate. The primary error modalities observed were false alarms generated during brief delta range slowing patterns such as intermittent rhythmic delta activity [25]. Our closed-loop experiments demonstrated that all regularization methods presented in Table 7, unfortunately, tend to increase the false alarm rate for slowing patterns.

Finally, in Fig. 18, an example of an EEG that is generated by the DCGAN structure of Fig. 6 is shown. Note that to generate these EEGs, we use a generator block in DCGAN in which each EEG signal has a 7 sec duration. We apply a 25 Hz low pass filter on the output of DCGAN, since most of the cerebral signals observed in scalp

Table 7. A comparison of performance for different regularizations

System	Sensitivity	Specificity	FA/24 Hrs.
L1/L2	30.8%	97.1%	6
Dropout	30.8%	96.9%	7
Gaussian	30.8%	95.8%	9
L2	30.2%	95.6%	10
L1	30.0%	43.7%	276

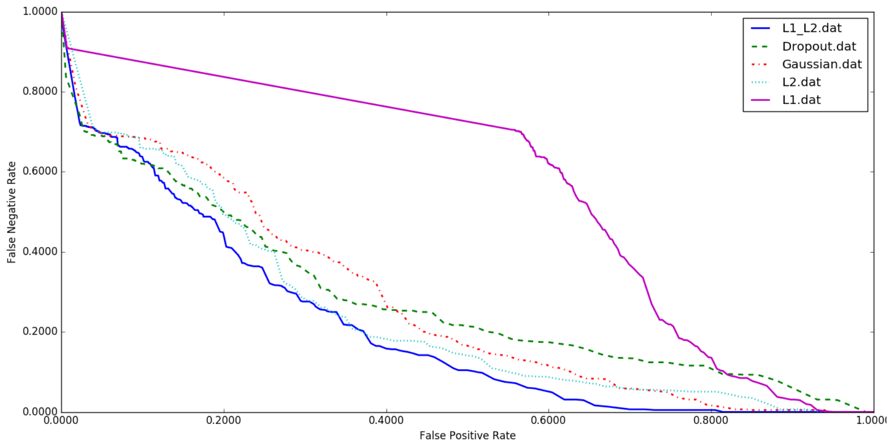

Fig. 17. A comparison of different regularization methods for CNN/LSTM

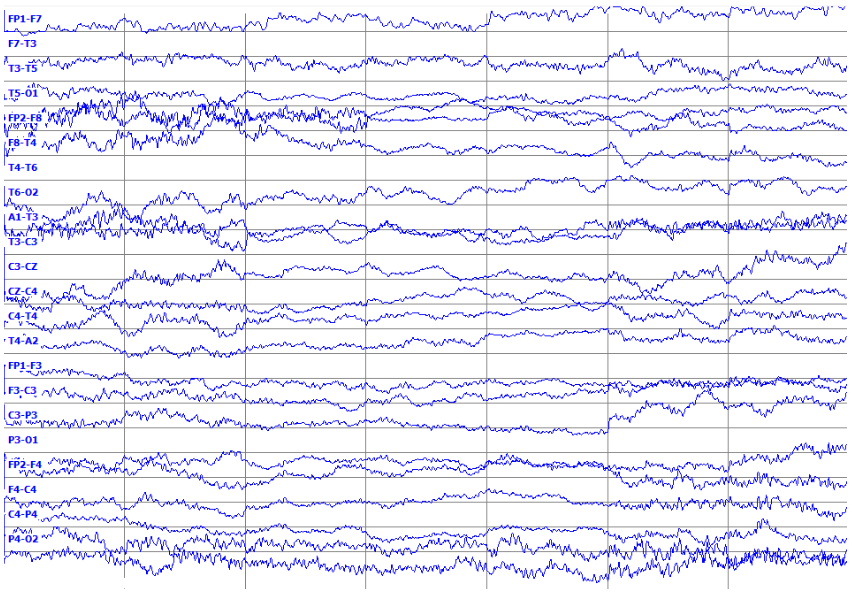

Fig. 18. Synthetic EEG waveforms generated using DCGAN.

EEGs fall in the range of 1–20 Hz (in standard clinical recordings, activity below or above this range is likely to be an artifact). Unfortunately, in a simple pilot experiment in which we randomly mixed actual EEGs with synthetic EEGs, expert annotators could easily detect the synthetic EEGs, which was a bit discouraging. Seizures in the synthetic EEGs were sharper and more closely resembled a slowing event. Clearly, more work is needed with this architecture.

However, our expert annotators also noted that the synthetic EEGs did exhibit focality. An example of focality is when activity is observed on the CZ-C4 channel, we would expect to observe the inverse of this pattern on the C4-T4 channel. As can be seen in Fig. 18, in last two seconds of the generated EEG, we observe slowing activity on the CZ-C4 channel and the inverse pattern of the same slowing activity on the C4-T4 channel. Hence, it is possible to generate synthetic multi-channel EEG signals with DCGAN that resemble clinical EEGs. However, DCGAN is not yet at the point where it is generating data that is resulting in an improvement in the performance of our best systems.

6 CONCLUSIONS

EEGs remain one of the main clinical tools that physicians use to understand brain function. New applications of EEGs are emerging including diagnosis of head trauma-related injuries which offer the potential to vastly expand the market for EEGs. A board-certified EEG specialist is required by law to interpret an EEG and produce a diagnosis. Since it takes several years of additional training post-medical school for a physician to qualify as a clinical specialist, the ability to generate data far exceeds the available expertise to interpret these data, creating a critical bottleneck. Despite rapid advances in deep learning in recent years, automatic interpretation of EEGs is still a very challenging problem.

We have introduced a variety of deep learning architectures for automatic classification of EEGs including a hybrid architecture that integrates CNN and LSTM technology. Two systems are particularly promising: CNN/LSTM and channel-based LSTM. While these architectures deliver better performance than other deep structures, their performance still does not meet the needs of clinicians. Human performance on similar tasks is in the range of 75% sensitivity with a false alarm rate of 1 per 24 hours [16]. The false alarm rate is particularly important to critical care applications since it impacts the workload experienced by healthcare providers.

The primary error modalities for our deep learning-based approaches were false alarms generated during brief delta range slowing patterns such as intermittent rhythmic delta activity. A variety of these types of artifacts have been observed during inter-ictal and post-ictal stages. Training models on such events with diverse morphologies is potentially one way to reduce the remaining false alarms. This is one reason we are continuing our efforts to annotate a larger portion of TUSZ.

We are also exploring the potential of supervised GAN frameworks for spatio-temporal modeling of EEGs. Most of the research on GANs is focused on either unsupervised learning or supervised learning using conditional GANs. Given that the annotation process to produce accurate labels is expensive and time-consuming, we are exploring semi-supervised learning in which only a small fraction of the data has labels. GANs can be used to perform semi-supervised classification by using a generator-discriminator pair to learn an unconditional model of data and then tune the discriminator using the small amount of labeled data for prediction.

We are also continuing to manually label EEG data. We invite you to register at our project web site, *www.isip.piconepress.com/projects/tuh_eeg*, to be kept aware of the latest developments.

References

1. Ilmoniemi, R., Sarvas, J.: Brain Signals: Physics and Mathematics of MEG and EEG. The MIT Press, Boston, Massachusetts, USA (2019)
2. Ebersole, J.S., Pedley, T.A.: Current practice of clinical electroencephalography. Wolters Kluwer, Philadelphia, Pennsylvania, USA (2014)
3. Yamada, T., Meng, E.: Practical guide for clinical neurophysiologic testing: EEG. Lippincott Williams & Wilkins, Philadelphia, Pennsylvania, USA (2017)
4. Ercegovac, M., Berisavac, I.: Importance of EEG in intensive care unit. Clin.

Neurophysiol. 126, e178–e179 (2015). doi:10.1016/j.clinph.2015.04.027

5. Ney, J.P., van der Goes, D.N., Nuwer, M.R., Nelson, L.: Continuous and routine EEG in intensive care: utilization and outcomes, United States 2005-2009. Neurology. 81, 2002–2008 (2016). doi:10.1212/01.wnl.0000436948.93399.2a

6. Boashash, B.: Time-Frequency Signal Analysis and Processing: A Comprehensive Reference. Academic Press, Inc., London, UK (2015)

7. Gotman, J.: Automatic detection of seizures and spikes. J. Clin. Neurophysiol. 16, 130–140 (1999)

8. Li, P., Wang, X., Li, F., Zhang, R., Ma, T., Peng, Y., Lei, X., Tian, Y., Guo, D., Liu, T., Yao, D., Xu, P.: Autoregressive model in the Lp norm space for EEG analysis. J. Neurosci. Methods. 240, 170–178 (2015). doi:10.1016/j.jneumeth.2014.11.007

9. Li, Y., Luo, M.-L., Li, K.: A multiwavelet-based time-varying model identification approach for time–frequency analysis of EEG signals. Neurocomputing. 193, 106–114 (2016). doi:10.1016/j.neucom.2016.01.062

10. Rodrıguez-Bermudez, G., Pedro J. Garcıa-Laencina: Analysis of EEG Signals using Nonlinear Dynamics and Chaos: A review. Appl. Math. Inf. Sci. 9, 2309–2321 (2015). doi:10.12785/amis/090512

11. Eichler, M., Dahlhaus, R., Dueck, J.: Graphical Modeling for Multivariate Hawkes Processes with Nonparametric Link Functions. J. Time Ser. Anal. 38, 225–242 (2017). doi:10.1111/jtsa.12213

12. Schad, A., Schindler, K., Schelter, B., Maiwald, T., Brandt, A., Timmer, J., Schulze-Bonhage, A.: Application of a multivariate seizure detection and prediction method to non-invasive and intracranial long-term EEG recordings. Clin. Neurophysiol. 119, 197–211 (2008)

13. Schindler, K., Wiest, R., Kollar, M., Donati, F.: Using simulated neuronal cell models for detection of epileptic seizures in foramen ovale and scalp EEG. Clin. Neurophysiol. 112, 1006–17 (2001). doi:10.1016/S1388-2457(01)00522-3

14. Deburchgraeve, W., Cherian, P.J., De Vos, M., Swarte, R.M., Blok, J.H., Visser, G.H., Govaert, P., Van Huffel, S.: Automated neonatal seizure detection mimicking a human observer reading EEG. Clin. Neurophysiol. 119, 2447–2454 (2008). doi:10.1016/j.clinph.2008.07.281

15. Baumgartner, C., Koren, J.P.: Seizure detection using scalp-EEG. Epilepsia. 59, 14–22 (2018). doi:10.1111/epi.14052

16. Haider, H.A., Esteller, R.D., Hahn, C.B., Westover, M.J., Halford, J.W., Lee, J.M., LaRoche, S.: Sensitivity of quantitative EEG for seizure identification in the intensive care unit. Neurology. 87, 935–944 (2016).

doi:10.1212/WNL.0000000000003034

17. Varsavsky, A., Mareels, I.: Patient un-specific detection of epileptic seizures through changes in variance. In: Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 3747–3750. IEEE, New York, New York, USA (2006)

18. Bridi, A.C., Louro, T.Q., Da Silva, R.C.L.: Clinical Alarms in intensive care: implications of alarm fatigue for the safety of patients. Rev. Lat. Am. Enfermagem. 22, 1034 (2014). doi:10.1590/0104-1169.3488.2513

19. Ahmedt-Aristizabal, D., Fookes, C., Denman, S., Nguyen, K., Sridharan, S., Dionisio, S.: Aberrant epileptic seizure identification: A computer vision perspective. Seizure Eur. J. Epilepsy. 65, 65–71 (2019). doi:10.1016/j.seizure.2018.12.017

20. Ramgopal, S.: Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy. Epilepsy Behav. 37, 291–307 (2014). doi:10.1016/j.yebeh.2014.06.023

21. Alotaiby, T., Alshebeili, S., Alshawi, T., Ahmad, I., Abd El-Samie, F.: EEG seizure detection and prediction algorithms: a survey. EURASIP J. Adv. Signal Process. 1–21 (2014). doi:10.1186/1687-6180-2014-183

22. Obeid, I., Picone, J.: The Temple University Hospital EEG Data Corpus. Front. Neurosci. Sect. Neural Technol. 10, 1–8 (2016). doi:10.3389/fnins.2016.00196

23. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature. 521, 436 (2015). doi:10.1038/nature14539

24. Shah, V., Golmohammadi, M., Ziyabari, S., von Weltin, E., Obeid, I., Picone, J.: Optimizing Channel Selection for Seizure Detection. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium. IEEE. 1–5, Philadelphia, Pennsylvania, USA (2017). doi: 10.1109/SPMB.2017.8257019

25. von Weltin, E., Ahsan, T., Shah, V., Jamshed, D., Golmohammadi, M., Obeid, I., Picone, J.: Electroencephalographic Slowing: A Primary Source of Error in Automatic Seizure Detection. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium. IEEE. 1–5, Philadelphia, Pennsylvania, USA (2017). doi: 10.1109/SPMB.2017.8257018

26. Golmohammadi, M., Ziyabari, S., Shah, V., Obeid, I., Picone, J.: Deep Architectures for Spatio-Temporal Modeling: Automated Seizure Detection in Scalp EEGs. In: Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA). 1–6, Orlando, Florida, USA (2018). doi: 10.1109/ICMLA.2018.00118

27. Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition (CVPR). IEEE. 1–9, Boston, Massachusetts, USA (2015)

28. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Proceedings of the International Conference on Learning Representations (ICLR). 1–14, San Diego, California, USA (2015)

29. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 770–778, Las Vegas, Nevada, USA (2016)

30. Radford, A., Metz, L., Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: Proceedings of the International Conference on Learning Representations (ICLR). San Juan, Puerto Rico (2016)

31. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Nets. Proc. Conf. Neural Inf. Process. Syst. 2672–2680 (2014). doi:10.1017/CBO9781139058452

32. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9, 1735–80 (1997). doi:10.1162/neco.1997.9.8.1735

33. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv Prepr. arXiv1412.3555. 1–9 (2014)

34. Natus Medical: Nicolet® NicVue Connectivity Solution, https://neuro.natus.com/products-services/nicolet-nicvue-connectivity-solution

35. Harati, A., Lopez, S., Obeid, I., Jacobson, M., Tobochnik, S., Picone, J.: The TUH EEG Corpus: A Big Data Resource for Automated EEG Interpretation. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium. 1–5, Philadelphia, Pennsylvania, USA (2014). doi: 10.1109/SPMB.2014.7002953

36. Lopez, S., Golmohammadi, M., Obeid, I., Picone, J.: An Analysis of Two Common Reference Points for EEGs. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium. 1–4, Philadelphia, Pennsylvania, USA (2016). doi: 10.1109/SPMB.2016.7846854

37. Hirsch L., J., Laroche S., M., Gaspard N., T., Gerard E., F., Svoronos A., ..., Drislane F., W.: American Clinical Neurophysiology Society's Standardized Critical Care EEG Terminology: 2012 version. J. Clin. Neurophysiol. 30, 1–27 (2013). doi:10.1097/WNP.0b013e3182784729

38. Shah, V., von Weltin, E., Lopez, S., McHugh, J.R., Veloso, L., Golmohammadi, M., Obeid, I., Picone, J.: The Temple University Hospital Seizure Detection Corpus. Front. Neuroinform. 12, 83 (2018). doi:10.3389/fninf.2018.00083

39. Shah, V., Anstotz, R., Obeid, I., Picone, J.: Adapting an Automatic Speech Recognition System to Event Classification of Electroencephalograms. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium (SPMB). p. 1. , Philadelphia, Pennsylvania, USA (2018). doi: 10.1109/SPMB.2016.7846854

40. Harati, A., Golmohammadi, M., Lopez, S., Obeid, I., Picone, J.: Improved EEG Event Classification Using Differential Energy. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium. 1–4, Philadelphia, Pennsylvania, USA (2015). doi: 10.1109/SPMB.2015.7405421

41. Swisher, C.B., White, C.R., Mace, B.E., Dombrowski, K.E.: Diagnostic Accuracy of Electrographic Seizure Detection by Neurophysiologists and Non-Neurophysiologists in the Adult ICU Using a Panel of Quantitative EEG Trends. J. Clin. Neurophysiol. 32, 324–330 (2015). doi:10.1097/WNP.0000000000000144

42. Kubota, Y., Nakamoto, H., Egawa, S., Kawamata, T.: Continuous EEG monitoring in ICU. J. Intensive Care. 6, 39 (2018). doi:10.1186/s40560-018-0310-z

43. Nihon Kohden Corporation, https://us.nihonkohden.com/products/eeg-1200

44. Picone, J.: Signal modeling techniques in speech recognition. Proc. IEEE. 81, 1215–1247 (1993). doi:10.1109/5.237532

45. Thodoroff, P., Pineau, J., Lim, A.: Learning Robust Features using Deep Learning for Automatic Seizure Detection. In: Machine Learning and Healthcare Conference (2016)

46. Mirowski, P., Madhavan, D., Lecun, Y., Kuzniecky, R.: Classification of patterns of EEG synchronization for seizure prediction. Clin. Neurophysiol. 120, 1927–1940 (2009). doi:10.1016/j.clinph.2009.09.002

47. Subasi, A.: EEG signal classification using wavelet feature extraction and a mixture of expert model. Expert Syst. Appl. 32, 1084–1093 (2007). doi:10.1016/j.eswa.2006.02.005

48. Jahankhani, P., Kodogiannis, V., Revett, K.: EEG Signal Classification Using Wavelet Feature Extraction and Neural Networks. IEEE John Vincent Atanasoff 2006 Int. Symp. Mod. Comput. 120–124 (2006). doi:10.1109/JVA.2006.17

49. Da Rocha Garrit, P.H., Guimaraes Moura, A., Obeid, I., Picone, J.: Wavelet Analysis for Feature Extraction on EEG Signals. In: NEDC Summer Research Experience for Undergraduates, Dept. of Electrical and Computer Engineering, Temple University. p. 1. , Philadelphia, Pennsylvania, USA (2015).

50. Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., Stolcke, A.: The Microsoft 2017 Conversational Speech Recognition System. In: Proceedings

of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 1–5, Calgary, Canada (2017)

51. Picone, J.: Continuous Speech Recognition Using Hidden Markov Models. IEEE ASSP Mag. 7, 26–41 (1990). doi:10.1109/53.54527

52. Huang, K., Picone, J.: Internet-Accessible Speech Recognition Technology. In: Proceedings of the IEEE Midwest Symposium on Circuits and Systems. p. III-73-III-76. , Tulsa, Oklahoma, USA (2002)

53. Parker, D., Picone, J., Harati, A., Lu, S., Jenkyns, M., Polgreen, P.: Detecting paroxysmal coughing from pertussis cases using voice recognition technology. PLoS One. 8, e82971 (2013). doi:10.1371/journal.pone.0082971

54. Lu, S., Picone, J.: Fingerspelling Gesture Recognition Using A Two-Level Hidden Markov Model. In: Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (ICPV). 538–543, Las Vegas, Nevada, USA (2013)

55. Obeid, I., Picone, J.: Machine Learning Approaches to Automatic Interpretation of EEGs. In: Sejdik, E. and Falk, T. (eds.) Signal Processing and Machine Learning for Biomedical Big Data. p. 30. Taylor & Francis Group, Boca Raton, Florida, USA (2018). doi: 10.1201/9781351061223

56. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the International Conference on Machine Learning (ICMLA). 1096–1103, New York, New York, USA (2008)

57. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol. J. Mach. Learn. Res. 11, 3371–3408 (2010). doi:10.1111/1467-8535.00290

58. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing System. 153–160, Vancouver, B.C., Canada (2007)

59. Hinton, G.E., Osindero, S., Teh, Y.-W.: A Fast Learning Algorithm for Deep Belief Nets. Neural Comput. 18, 1527–1554 (2006). doi:10.1162/neco.2006.18.7.1527

60. van der Maaten, L., Postma, E., van den Herik, J.: Dimensionality Reduction: A Comparative Review. (2009)

61. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. 77, 125–141 (2008). doi:10.1007/s11263-007-0075-7

62. Zinkevich, M., Weimer, M., Smola, A., Li, L.: Parallelized Stochastic Gradient Descent. In: Proceedings of Neural Information Processing Systems. 2595–

2603, Vancouver, British Columbia, Canada (2010)

63. Golmohammadi, M., Harati Nejad Torbati, A.H., de Diego, S., Obeid, I., Picone, J.: Automatic Analysis of EEGs Using Big Data and Hybrid Deep Learning Architectures. Front. Hum. Neurosci. 13, 76 (2019). doi:10.3389/fnhum.2019.00076

64. Saon, G., Sercu, T., Rennie, S., Kuo, H.-K.J.K.J.: The IBM 2016 English Conversational Telephone Speech Recognition System. In: Proceedings of the Annual Conference of the International Speech Communication Association. 7–11 (2016)

65. Lopez, S.: Automated Identification of Abnormal Adult Electroencephalograms, epartment of Electrical and Computer Engineering, Temple University. 58, Philadelphia, Pennsylvania, USA. url: https://digital.library.temple.edu/digital/collection/p245801coll10/id/463223/rec/1, (2017)

66. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. 521, 436–444 (2015)

67. Nair, V., Hinton, G.E.: Rectified Linear Units Improve Restricted Boltzmann Machines. Proc. Int. Conf. Mach. Learn. 807–814 (2010). doi:10.1.1.165.6419

68. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning for Audio, Speech and Language Processing. p. 6. , Atlanta, Georgia, USA (2013)

69. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res. 15, 1929–1958 (2014). doi:10.1214/12-AOS1000

70. Kingma, D.P., Ba, J.L.: Adam: a Method for Stochastic Optimization. In: Proceedings of the International Conference on Learning Representations. 1–15, San Diego, California, USA (2015)

71. Jelinek, F.: Statistical Methods for Speech Recognition. The MIT Press, Boston, Massachusetts, USA (1997)

72. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York, New York, USA (2011)

73. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. 20, 273–297 (1995). doi:10.1007/BF00994018

74. Bahl, L., Brown, P., de Souza, P., Mercer, R.: Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. 49–52, Tokyo, Japan (1986)

75. Pandey, P.: Deep Generative Models, https://towardsdatascience.com/deep-generative-models-25ab2821afd3

76. Day, M.Y., Tsai, C.C., Chuang, W.C., Lin, J.K., Chang, H.Y., ... Fergus, R.: NIPS 2016 Tutorial: Generative Adversarial Networks. EMNLP. (2016). doi:10.1007/978-3-319-10590-1_53

77. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved Techniques for Training GANs. In: Proceedings of Neural Information Processing Systems (NIPS). 1–9, Barcelona, Spain (2016)

78. Yang, S., López, S., Golmohammadi, M., Obeid, I., Picone, J.: Semi-automated annotation of signal events in clinical EEG data. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium (SPMB). IEEE. 1–5, Philadelphia, Pennsylvania, USA (2016). doi: 10.1109/SPMB.2016.7846855

79. Lang, K.J., Waibel, A., Hinton, G.E.: A time-delay neural network architecture for isolated word recognition. Neural Networks. 3, 23–43 (1990). doi:10.1016/0893-6080(90)90044-L

80. Levy, A., Lindenbaum, M.: Sequential Karhunen-Loeve basis extraction and its application to images. Proc. IEEE Trans. Image Process. 9, 1371–1374 (2000). doi:10.1109/ICIP.1998.723422

81. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In: International Conference On Learning Representations (ICLR). 1–14, San Juan, Puerto Rico (2016)

82. Golmohammadi, M., Ziyabari, S., Shah, V., Obeid, I., Picone, J.: Gated Recurrent Networks for Seizure Detection. In: Obeid, I. and Picone, J. (eds.) Proceedings of the IEEE Signal Processing in Medicine and Biology Symposium. IEEE. 1–5, Philadelphia, Pennsylvania, USA (2017). doi: 10.1109/SPMB.2017.8257020

83. Hermans, M., Schrauwen, B.: Training and Analyzing Deep Recurrent Neural Networks. Adv. Neural Inf. Process. Syst. 190–198 (2013). doi:http://dl.acm.org/citation.cfm?id=2999611.2999633

84. Graves, A., Mohamed, A., Hinton, G.: Speech Recognition With Deep Recurrent Neural Networks. Int. Conf. Acoust. Speech, Signal Process. 6645–6649 (2013). doi:10.1109/ICASSP.2013.6638947

85. Krauss, G.L., Fisher, R.S.: The Johns Hopkins Atlas of Digital EEG: An Interactive Training Guide. Johns Hopkins University Press, Baltimore, Maryland, USA (2011)

86. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Proceedings of the International Conference on Machine Learning (ICML). 448–456, Lille, France (2015)

87. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the Importance of Initialization and Momentum in Deep Learning. In: Proceedings of the

International Conference on Machine Learning (ICML). 1139–1147, Atlanta, Georgia, USA (2013)

88. Shah, V., von Weltin, E., Ahsan, T., Obeid, I., Picone, J.: A Cost-effective Method for Generating High-quality Annotations of Seizure Events. J. Clin. Neurophysiol. (2019) (in review). url: www.isip.piconepress.com/publications/unpublished/journals/2017/jcn/ira

89. Fiscus, J., Ajot, J., Garofolo, J., Doddingtion, G.: Results of the 2006 Spoken Term Detection Evaluation. In: Proceedings of the SIGIR 2007 Workshop: Searching Spontaneous Conversational Speech. 45–50, Amsterdam, Netherlands (2007)

90. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: a classification perspective, https://www.amazon.com/Evaluating-Learning-Algorithms-Classification-Perspective/dp/1107653118, (2014)

91. Shah, V., Picone, J.: NEDC Eval EEG: A Comprehensive Scoring Package for Sequential Decoding of Multichannel Signals, https://www.isip.piconepress.com/projects/tuh_eeg/downloads/nedc_eval_eeg/

92. Shah, V., Golmohammadi, M., Obeid, I., Picone, J.: Objective evaluation metrics for automatic classification of EEG events. J. Neural Eng. 1–21 (2018) (in review). url: www.isip.piconepress.com/publications/unpublished/journals/2018/iop_jne/metrics/

93. Liu, A., Hahn, J.S., Heldt, G.P., Coen, R.W.: Detection of neonatal seizures through computerized EEG analysis. Electroencephalogr. Clin. Neurophysiol. 82, 32–37 (1992). doi:10.1016/0013-4694(92)90179-L

94. Navakatikyan, M.A., Colditz, P.B., Burke, C.J., Inder, T.E., Richmond, J., Williams, C.E.: Seizure detection algorithm for neonates based on wave-sequence analysis. Clin. Neurophysiol. 117, 1190–1203 (2006). doi:http://dx.doi.org/10.1016/j.clinph.2006.02.016

95. Fiscus, J.G., Chen, N.: Overview of the NIST Open Keyword Search 2013 Evaluation Workshop. , Bethesda, Maryland, USA (2013)

96. Sundermeyer, M., Ney, H., Schluter, R.: From Feedforward to Recurrent LSTM Neural Networks for Language Modeling. IEEE/ACM Trans. Audio, Speech, Lang. Process. 23, 517–529 (2015). doi:10.1109/TASLP.2015.2400218

97. Bottou, L., Lecun, Y.: Large Scale Online Learning. Adv. Neural Inf. Process. Syst. 217–225 (2004). doi:https://papers.nips.cc/paper/2365-large-scale-online-learning

98. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA Neural Networks Mach. Learn. (2012)

99. Duchi, J., Hazan, E., Singer, Y.: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. J. Mach. Learn. Res. 12, 2121–2159 (2011). url: https://dl.acm.org/citation.cfm?id=2021068

100. Zeiler, M.D.: ADADELTA: An Adaptive Learning Rate Method. arXiv. abs/1212.5, 1–6 (2012)

101. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In: Proceedings of the International Conference on Learning Representations (ICLR). 1–22, Banff, Canada (2014)

102. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, Massachusetts, USA (2016)

103. Perez, L., Wang, J.: The Effectiveness of Data Augmentation in Image Classification using Deep Learning. arXiv. (2017)