Epileptic Seizure Detection from EEG Signals with Autoencoded Features

Tuan Nguyen

March 26, 2018

1 Definition

1.1 Project Overview

The domain of interest of this project belongs to physiological data such as electroencephalography (EEG), magnetoencephalography (MEG), electrocardiography (ECG) and the recorded signals from wearable devices. This project focuses on the EEG signals, which capture activities of brain neurons during a period of time. Different kinds of EEG data has been recorded from humans, for instance from those at rest, sleep [9], during some periods of specific cognitive activities, or from patients with diseases such as Alzheimer's, Parkinson's, depression and epileptic seisures ([1] and references thereof).

This project studies the classification problem on an EEG data set recorded from healthy volunteers and patients having epileptic seizures [1]; solutions for this problem could be used to assist with detecting patients with the disease from their brain activity signals. Previous work on this data set focused mainly on extracting hand-crafted features to be used for classification. As examples, Nigam and Graupe [10] extracted the spike amplitudes and frequency of the signals, feeding them into a neural network for classification; Guler et al. [4] applied wavelet transformation to the signals to extract features, and classification was performed using a neuro-fuzzy system; Kannathal et al. [7] extracted entropy-based features for classification. On another data set, Wulsin et al. [13] learned features of second-long segments of EEG signals using Deep Belief Networks [5] and classified them into "clinically significant" EEG classes. To the best of my knowledge, there was

no previous work reporting the results of using autoencoders for the data set that this project is interested in. The purpose of this work, therefore, is to explore the use of traditional and denoising autoencoders ([2], [12]) in learning the features representing the EEG signals and then use them to classify unseen brain activity signals into different classes of patients. For this classification purpose, the current implementation was limited to the use of multi-layer fully-connected neural networks where their hidden layers were used to fine-tune the feature learned by the autoencoders, and a softmax layer served as the ouput layer discriminating input signals into classes. To the end, I present extensive experiments that show the great benefits of using denoising autoencoders in detecting epileptic seizure from EEG signals.

1.2 Problem Statement

This project aims to classify 1-second long EEG segments into one of the three classes: (1) healthy volunteers, (2) patients with epileptic seizures disease during seizure-free periods, and (3) patients with the disease during active seizure periods. It can be formally defined as follows.

- **Input:** training set $X = \langle \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \rangle$ of N samples, where $\mathbf{x}_i$ is a vector of M electrical voltage values recorded in one second by an electrode at a specific point and time; target vector $y = \langle y_1, y_2, ..., y_N \rangle$ where $y_i \in \{1, 2, 3\}$ is the type of volunteers which the sample $\mathbf{x}_i$ belong to. In the data set [1], the targets $1, 2, 3$ respectively correspond to sets of volunteers $A + B$, $C + D$ and E.

- **Output:** a hypothesis h classifying a sample of M electrical voltage values into one of the volunteer classes $\{1, 2, 3\}$.

The formation of the training set X and target y is presented in more details in Section 2.1. This project explores the benefits of using autoencoders in learning a representation $f(\mathbf{x}_i)$ of signal segments $\mathbf{x}_i \in X$ that can be helpful for our classification task at hand; the two types of autoencoders considered in this work are traditional [2] and denoising autoencoder [12]. The extracted features are then tested with popular learning models (Decision Trees, K-Kearest Neighbors, Support Vector Machines and Feedforward Multi-layer Neural Networks) for classification (see Section 2.2).

1.3 Metrics

The quality of the returned hypothesis is evaluated on an independent test set using the accuracy measure, which is the percentage of samples classified into their correct classes. The test set is created from randomly chosen 20% of the given data set. Although, as suggested by Wulsin et al. [13], the classification time is considered important for applications monitoring EEG signals of patients, the testing set is small enough that the prediction time performed by the multi-layer neural network classifiers is much less than 1 second and therefore will not be discussed further in the result section. (I also regret to note that my implementation did not initially put in tensors for recording the F_1 score, which should also be a meaningful measure to examine.)

2 Analysis

2.1 Data Exploration and visualization

The data set used in this study is provided with the work by Andrzejak et al. [1], recording brain electrical activity of five sets of human volunteers:

- Set A: healthy volunteers in relaxed state with eyes open.
- Set B: healthy volunteers in relaxed state with eyes closed.
- Set D: epilepsy patients during seizure-free periods; the electrodes were implanted to record brain activity from within the epileptogenic zone.
- Set C: epilepsy patients during seizure-free periods; the electrodes were implanted from the opposite hemisphere of the brain (compared to those in Set D).
- Set E: same patients with sets C and D but activity were recorded from all electrodes during active seizures.

For each set of volunteers above, the data set contains 100 single-channel EEG segments of 4096 signal micro-Voltage values in time domain of 23.6-sec duration. For this project each of these segments is divided into smaller segments of 1-second durations, which are considered stationary for EEG data [10]. These 1-second durations together define the training set X of 178

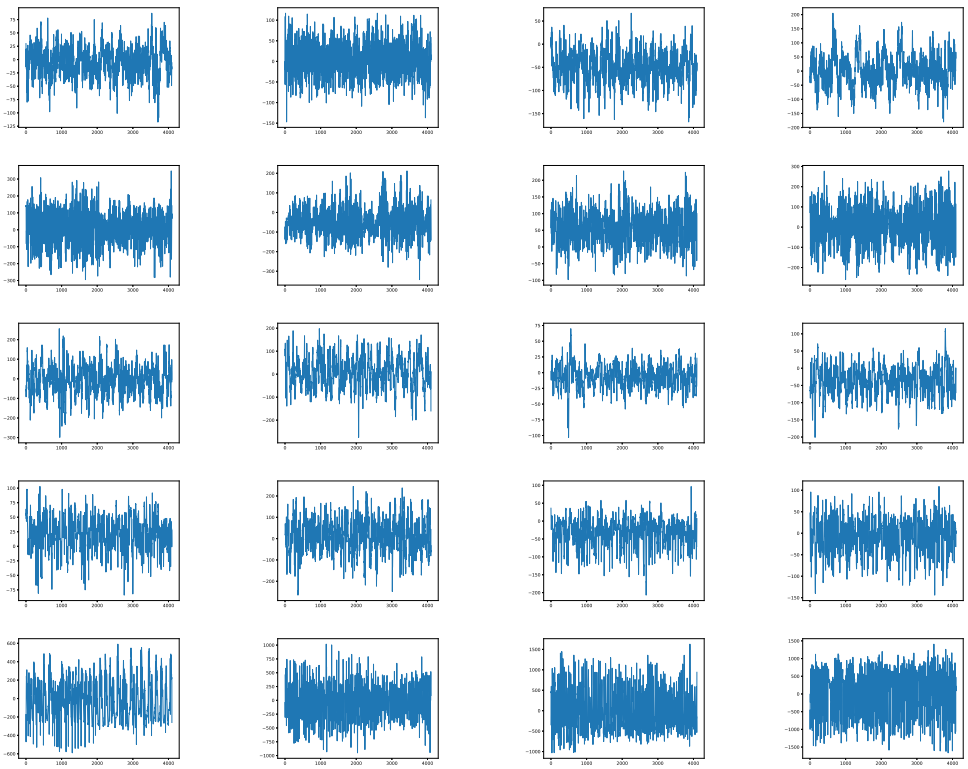

Figure 1: Images of brain activity signals from patients of classes A (1st row), B(2nd row), C(3rd row), D(4th row) and E(5th row).

dimensions, and the three sets of volunteers $A + B$, $C + D$ and E define the target vector y in the problem statement above.

Figure 1 shows four randomly chosen 23.6-second segments of each class from which we can only observe very slightly differences in patterns of the signals in each class. However, by looking at the shape of each class more closely it is interesting to notice some very distinguished segments that belong to epileptic seizure patients in both seizure-free or seizure-active periods (i.e., class C, D and E), as shown in Figure 2.

Figure 3 shows the chart of common statistics of signal values within each class, and the detailed numbers are shown in Table 1. The data points of class E, the category of patients in their seizure-active periods, have the smallest average value (-4.74) but spread the most among the classes with standard deviation 341.15.

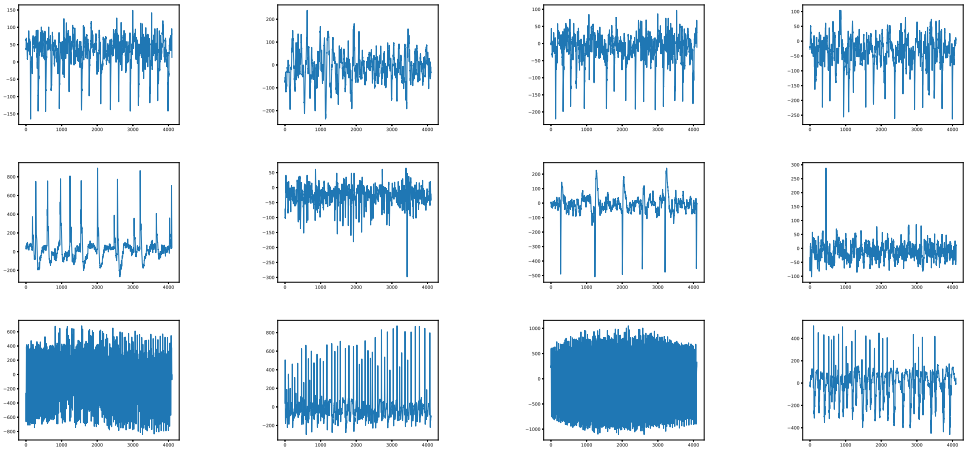

Figure 2: Some distinguished signals from patients of classes C (1st row), D(2nd row), E(3rd row).

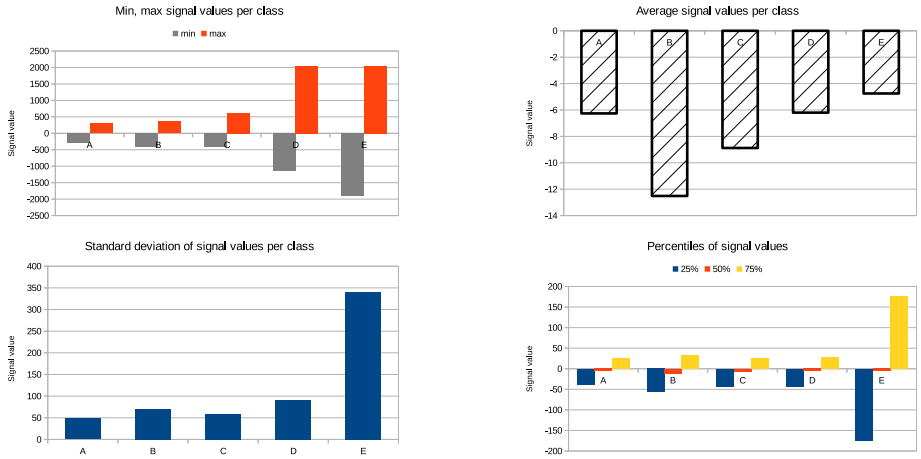

Figure 3: Some statistics of the data set per class: min, max (top left), mean (top right), standard deviation (bottom left) and percentiles (bottom right).

Class	Min	Max	Mean	Stddev	25%	50%	75%
A	-288	294	-6.26	48.33	-38	-6	25
B	-424	360	-12.51	70.68	-57	-12	32
C	-412	623	-8.88	59.38	-43	-7	26
D	-1147	2047	-6.20	90.34	-43	-6	29
E	-1885	2047	-4.74	341.15	-175	-5	177

Table 1: Statistics of the data set per class.

2.2 Algorithms and Techniques

This project aims to use a type of autoencoders ([2], [12], [3], [8]) that can learn to extract features of the training set X for the purpose of classifying new sample of EEG segments. I propose to first try traditional stacked autoencoders [2]; the other autoencoders will be explored depending on the performance of the first one on this data set.

2.2.1 Training autoencoders

An *traditional autoencoder* [12] is a fully connected feed-forward neural network which consists of an input layer, one hidden layer and an output layer such that the input and output layers have the same number of neurons. The computation performed by the hidden layer, specified by a weight matrix $\mathbf{W}$, bias vector $\mathbf{b}$ and nonlinear activation function s, acts as an *encoder* that maps an input vector $\mathbf{x}$ to a representation $f_\theta(\mathbf{x})$ defined as follows:

$$f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}).$$

The parameter set $\theta = (\mathbf{W}, \mathbf{b})$ is part of what to be adjusted. The encoding $f_\theta(\mathbf{x})$ is then mapped back to vector $\mathbf{z}$ in the same space with $\mathbf{x}$ through the computation of the output layer, representing the *decoder* part of the autoencoder:

$$\mathbf{z} = g_{\theta'}(f_\theta(\mathbf{x})),$$

where $g_{\theta'}$ can be either a linear or a non-linear function whose parameter θ' is the other part of what to be learned. With real-valued input signals $\mathbf{x}$, the output $\mathbf{z}$ in general is interpreted not as the exact reconstruction of $\mathbf{x}$ but

the mean of a distribution $P(\mathbf{x}|\mathbf{z})$. Squared error function between $\mathbf{x}$ and $\mathbf{z}$, and thus the objective function

$$L(X) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i} - \mathbf{z_i})^2$$

is to be optimized during the training of the autoencoders.

In order for autoencdoers not to produce trivial encoding and decoding functions (e.g., to avoid the identity function from input x to itself), certain types of constraints need to be imposed. In traditional autoencoder, a constraint could be that the number of neurons of the hidden layer is smaller than the dimension of the inputs x_i's. In so-called *denoising autoencoders* [12], the inputs are made "dirty" before being fed into the encoder—by attempting to reconstruct the original input from its corrupted version, denoising autoencoders might learn representations that are more essential to the inputs. In this work, the input signals are made corrupted using one of the following two ways:

- Adding into the inputs random noises generated by a Gaussian distribution with zero mean and σ standard deviation $N(0, \sigma^2)$.
- Disable a random number of neurons at the input layer of the autoencoders, which can be implemented using dropout techniques [11] with dropout rate denoted by ρ $(0 < \rho < 1)$.

Since the purpose of the above autoencoders is to learn representations of x_i by bring the ouput z as close as possible to $\mathbf{x}$. The reconstruction error between input vector $\mathbf{x}$ and its reconstructed $\mathbf{z}$ is defined with the squared error function $(\mathbf{z} - \mathbf{x})^2$

2.2.2 Stacking up autoencoders

The first (lowest) autoencoder on the stack is trained to minimize the difference, defined for instance as the mean squared error, between the input batches of EEG segments and their output activations. The output activation of the hidden layer becomes the input to the second autoencoder on the stack, which is trained also to minimize the difference between its input and output layers. This process of adding more autoencoders continues in that fashion. Finally, the output layer of the entire stack is the output of the first

autoencoder. The stacked autoencoders is therefore symmetrical with regard to the middle hidden layer: the first lower half acts as the encoder while the second higher half of the stack as the decoder. The output activations of the middle layer thus act as encoded features of the training set of EEG segments.

Depending on the performance of the above autoencoder in classifying the target classes of volunteers, I might also plan to train a stack of denoising autoencoders [12] to extract features for the data set. At high level, this stack is much like the traditional one except for the Gaussian noises added into the input layer, forcing the decoder to learn more meaningful features, potentially improve the performance in classification.

After the feature learning phase above, the output of the coding layer the network is considered the features representing the training set X. They are then used as input features of popular classification algorithms. This work considers the following classifiers: Decision Trees, K-Kearest Neighbors, Support Vector Machines and Feedforward Multi-layer Neural Networks. The hypothesis here is that the features learned automatically using autoencoders, together with one of these classifiers, provides comparable performance compared to the previous work (see Section **??**).

2.3 Benchmark

Several previous work has been developed for the classification problem on the same data set [1]; the main contribution of most of these work, however, were on designing various hand-crafted features for the EEG signals. They might also be different from each other in the target classes of interest. As examples, Nigam and Graupe [10] used two features based on the spike information of the signals for further learning and detecting patients in set E (thus, binary classification problem). Kabir et al. [6] extracted statistical features based on "optimum allocation technique" to be used with logistic model trees for classifying volunteers into the three classes that are of interest of this project. Guler et al. [4] employed Lyapunov exponents based features for classification using recurrent neural networks where the target classes are three sets A, D and E. On another data set, Wulsin et al. [13] learned features of second-long segments of EEG signals using Deep Belief Networks [5] and classified them into "clinically significant" EEG classes. To the best of my knowledge, there was no previous work reporting the results of using autoencoders for the data set that this project is interested in.

The proposed method by Kabir et al. [6] on the data set of interest in this project outperforms many the state-of-the-art approaches using the same data set; among many other measures, the total accuracy of their approach was 95.3%. Since its target classes are the same with those of interest in this project, and their work represents a class of work using hand-crafted features, in my opinion it is reasonable to use the performance of their work as the baseline for the approach to be explored in this project. The hypothesis is that by using autoencoders, it is possible to learn a set of features automatically for classifying EEG signals with comparable results. The next section discusses measures to evaluate performance of classifiers in this domain.

3 Methodology

3.1 Data Preprocessing

The data set [1] contains 100 channels (or vector of voltage values) for each of the five sets of volunteers. Each channel is a vector of 4097 values that are EEG signals recorded in 23.6 seconds. The data set will be preprocessed as follows.

- Each channel is divided into 23 segments with equal length, each contains 178 values. These segments are marked with the target value 0, 1 or 2 for the classes $A + B$, $C + D$ and E of the original channel.
- The resulting set of segments are splitted into training and test set (using StratifiedShuffleSplit function from Scikit-Learn).
- The resulting set of segments in the training set are then normalized so that the values of EEG signals are in between $[0, 1]$.

The data set was divided into three parts: training set (60%), validation and testing sets (20% each). The testing set was set aside and not used in any ways during training or validating candidate models.

The training data was scaled into $[-1, 1]$ as follows (the formula):... This is to fit to ELU activation function.

The resulting scaling coefficients were then used to scale the testing and validation sets. This way we made sure that there were no information from the validation or testing sets used in any ways during training (data snooping).

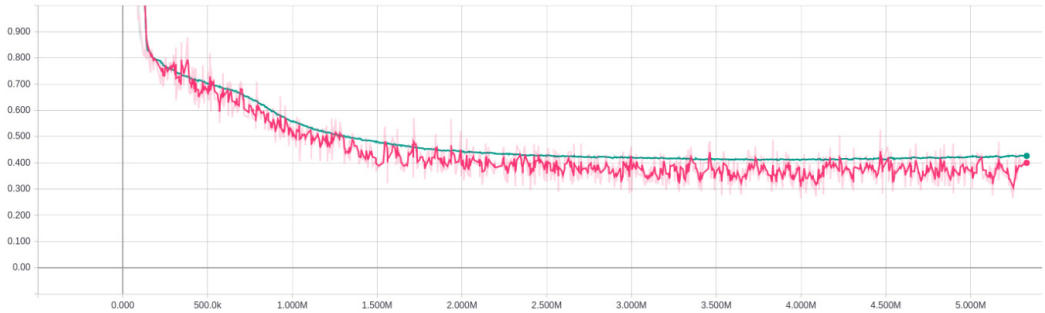

Figure 4: Cross-entropy error (Y-axis) on the training (red) and validation (green) sets of a 2-layer, 50 neurons each, fully connected network, called A_0, with randomly initialized weights and biases. Notice that the validation curve started to go up after about 3.8 millions of batch updates, indicating the overfitting of the network.

3.2 Implementation

3.3 Refinement

4 Results

A. Randomly initialized vs. pretrained weights and biases: We first examine the benefit of using weights and biases pretrained in unit autoencoders to initialize a multi-layer neural networks. Figure 4 shows the learning curves for the training set (in red) and the validation set (in green) for a stack of two unit denoising autoencoders that were not pretrained. In other words, the weights and biases of the 2-layer fully connected network with 50 neurons per layer, called A_0, were initialized randomly. Dropout technique with the rate 0.33 (for input layer) and 0.5 (for hidden layers) are used to cope with overfitting. During the training of this network, the best model was saved at step $3,878,215$ of batch updates, after which the network started to overfit to the training data (i.e., the validation curve began to go up).

Figure 5 shows the learning curves for a similar stack as above but the two unit denoising autoencoders were pretrained to reconstruct their inputs, and the resulting weights and biases were used to initialize the stack before being fine-tuned. We could observe that the network did not overfit even after more than 18 millions batch updates, which is much better than A_0. The cross-entropy error of this network on the validation set can also be seen as

Network	Train set	Validation set	Test set
A_0	89.59%	87%	87.52%
A_1	93.17%	88.96%	90.34%

Table 2: Accuracy by networks A_0 and A_1 on training, validation and test sets.

Figure 5: Cross-entropy error (Y-axis) on the training (red) and validation (green) sets of a 2-layer fully connected network, called A_1, whose weights and biases were initialized with pretrained values.

much better than from the one with randomly initialized weights and biases: less than 0.3 vs. greater than 0.4. This by definition results in a difference between A_1 and A_0 in terms of prediction accuracy on the validation, and importantly, the *test* sets: 90.34% vs. 87.52%, respectively. The accuracy errors by the two networks on the various sets are shown in Table 2.

Although the network A_1 had a wider accuracy gap between its training and validation sets, it is important to stress that this gap did not appear to be widening, which was not the case for the network A_0. If the training were to be allowed on A_1, we would expect that its generalization capacity continues to be enhanced; the network A_0, however, overfitted early and would not have any benefits from further training.

B. Traditional vs. denoising autoencoders: We now compare two types of autoencoders, traditional and denoising, in terms of reconstructing orginal inputs and, more importantly, of learning representations for our classification problem. The denoising autoencoders here were those used to pretrain weights/biases of A_1, which learned their unique features by reconstruct-

ing the inputs from their corrupted version. The traditional autoencoders had the same configuration with the other type (i.e., number of neurons, dropout rates for the hidden layer), except that they attempted to reconstruct the inputs from themself instead of their corrupted version. To test their representations in classifying the target signals, we constructed a 2-layer fully-connected neural network, called B_0, whose weights and biases were initialized from the two pretrained traditional autoencoders.

Table 3 shows the reconstruction errors by two autoencoders of the above types used to construct the layers of networks A_1 and B_0. On both the training and validation sets, the traditional autoencoders which processed the inputs directly resulted in more accurate reconstructed signals, i.e. with smaller errors, than those produced by the denoising counterpart; and this is true for autoencoders at two layers of the networks.

Autoencoder	Layer 1		Layer 2	
	Train	Validation	Train	Validation
Traditional	0.0015	0.0016	0.00051	0.00052
Denoise	0.0019	0.0021	0.0011	0.0011

Table 3: Reconstruction errors by traditional and denoise autoencoders used at layer 1 and 2 of networks B_0 and A_1, respectively.

The outperformance of traditional autoencoders in reconstructing the input signals, however, did not transform to its classification performance. Figure 6 shows the cross-entropy errors by the network B_0: it was expressive enough and achieved 97.13% accuracy on the training set, however overfitted early after step $1,675,620$ of batch updates. The accuracy of the resulting best model on the validation and testing sets were 90.47% and 91.78%, respectively. (The corresponding statistics by A_1 was presented in the previous result section.) Both the about 7% performance gaps on between the training and the two independent sets, and the $> 1\%$ gap between the validation and test sets indicate that this model suffers from high variance. The network A_1, which used features extracted through reconstructing corrupted inputs, as shown in Figure **??**, provided a comparable and more reliable accuracy performance on unseen data.

Perhaps one explanation for the above difference could be that since the two networks were the same except for the clean vs. the corrupted inputs

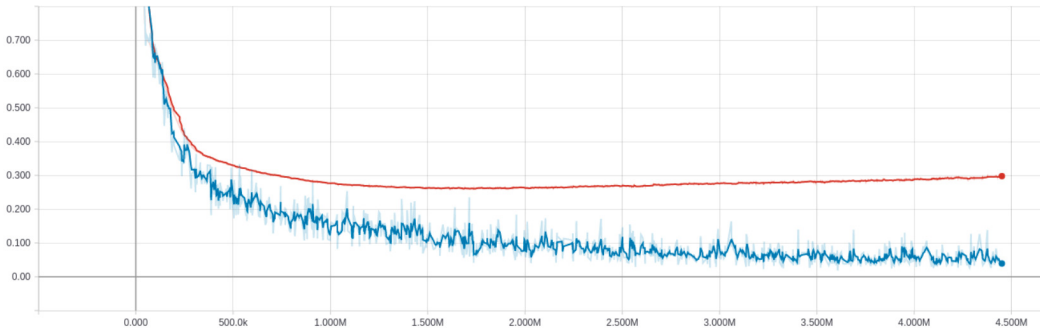

Figure 6: Cross-entropy error (Y-axis) on the training (green) and validation (red) sets of a 2-layer fully connected network, called B_0, with inputs not being corrupted during training.

that they processed, the noises introduced into the inputs helped the network A_1 learned features more essential to the original inputs; when used for classification task, these features resulted in much less overfitting model compared to those simply learned from an identity function of the clean inputs. Although it is hard for humans to make sense the features learned by the two networks to represent brain activity signals (in contrast to interpretable features in image domains), it is interesting to see what the neurons were "excited" about after the learning process. Figure 7 shows the weights of six randomly chosen neurons learned by the first traditional autoencoder used in network B_0: we could observe that the weights did not appear to learn any interesting patterns after reconstructing the original inputs.

For the same set of neurons, Figure 8 shows their weights learned by the denoising autoencoder used in the first hidden layer of network A_1 after reconstructing the inputs from their corrupted version. As opposed to those in Figure 7, these weights appear to reflect some specific patterns that the neurons were adapted to. Similar contrasts could also be observed for autoencoders at the second layers of the networks B_0 and A_1.

Comparision on networks of 75-neuron layers: Table 4 shows the accuracy performances of networks B_1 and B_2 with 2 layers fully connected networks, each having 75 neurons. They differed in exactly the same way B_0 and A_1 were distinguished: B_1 reconstructed the original signals from clean inputs, whereas B_2 did that from the corrupted inputs. We can again observe from the accuracy gaps that by reconstructing the inputs from their corrupted version, the denoising autoencoders extracted features that helped

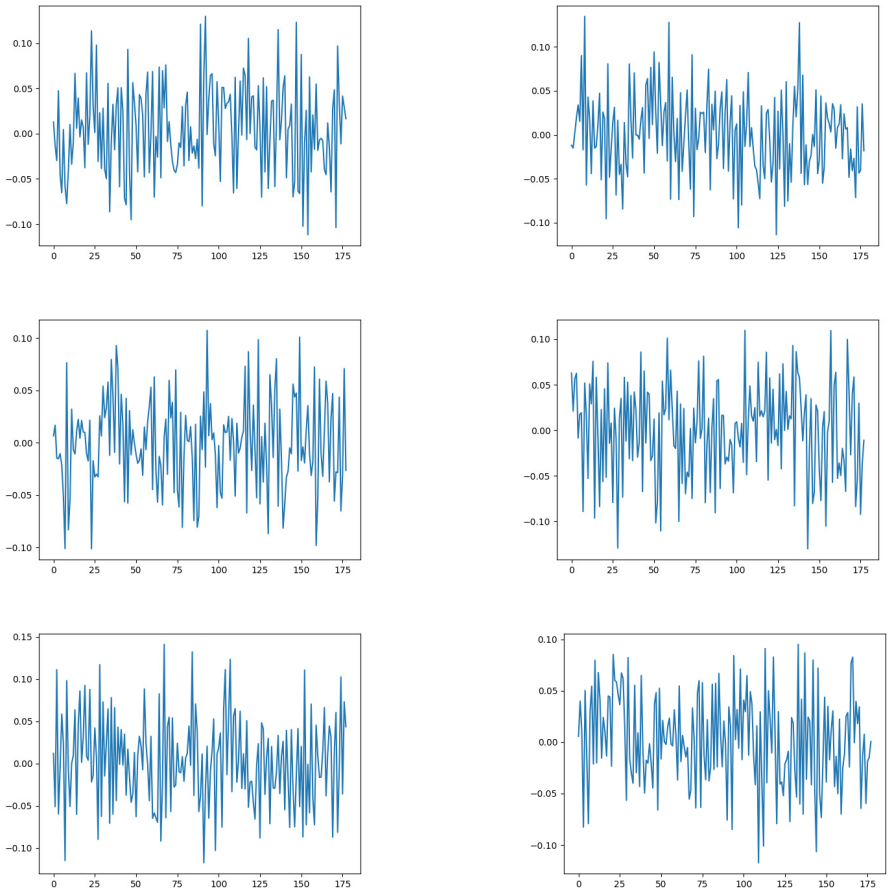

Figure 7: Weights of six neurons, chosen randomly, learned by the first traditional autoencoder used in network B_0 after reconstructing the original (uncorrupted) inputs.

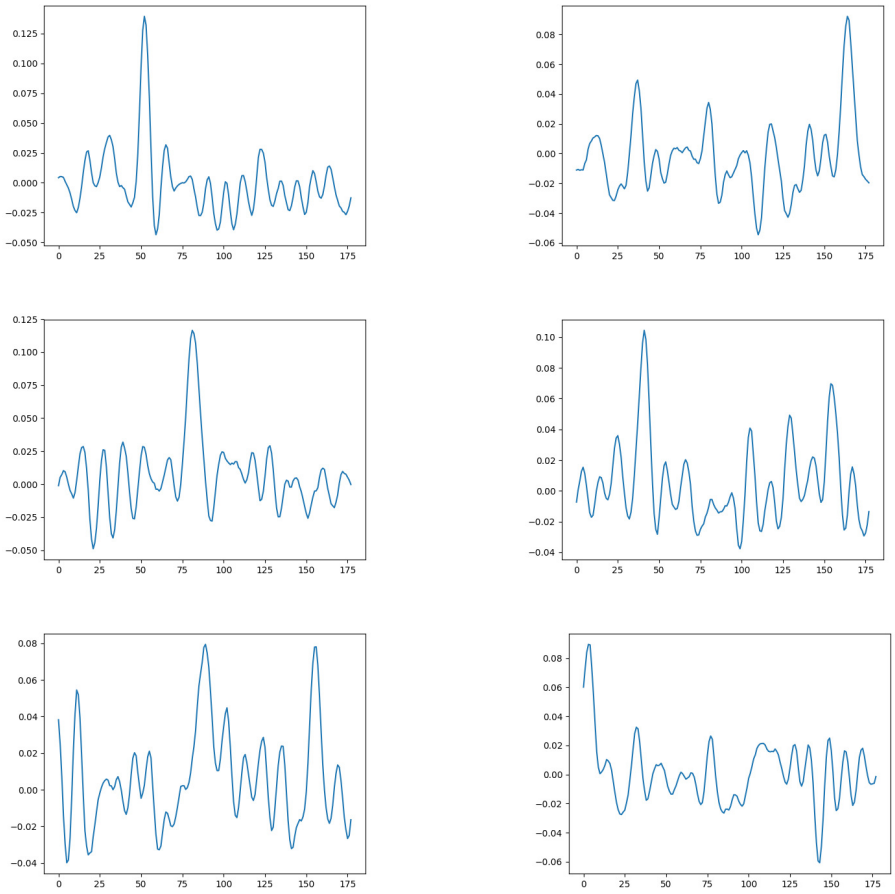

Figure 8: Weights of the corresponding six neurons in Figure 7 learned by the first denoising autoencoder used in network A_1 after reconstructing the corrupted inputs.

Network	Train set	Validation set	Test set
B_1	97.3%	89.61%	90.22%
B_2	94.28%	89.99%	90.69%

Table 4: Accuracy by networks with 2 layers of 75 neurons B_1 and B_2 on training, validation and test sets. B_1 reconstructed the original signals from themselves, and B_2 did so from the corrupted inputs.

Network	Train set	Validation set	Test set
C_0	93.84%	90.43%	90.52%
A_1	93.17%	88.96%	90.34%

Table 5: Accuracy by networks C_0 and A_1 representing the effects of using Gaussian noises and dropout to corrupt the inputs for learning representations.

the network classifier learned a model that was more reliable in predicting the target classes.

B1. The recovered signals:

C. Gaussian noisy inputs vs. dropout inputs:
I found that using Gaussian noises to corrupt the inputs of autoencoders seemed to work equally well compared to using the dropout technique. Figure 9 shows the learning curves on the training and validation sets produced by a network, called C_0, that is similar to A_1 except for inputs being corrupted using Gaussian noises of 0.1 standard deviation (instead of dropout with rate 0.33 used by A_1 and its autoencoders.) The best model by C_0 was saved at step $17,575,820$ of batch updates, after which the performance on the validation set started to decrease indicating that the model began to overfit to the training data. The accuracy performance of C_0, compared to A_1, is summarized in Table 5.

D. Denoising autoencoders with dropout: number of neurons and layers *Number of hidden neurons:* Table 6 shows the accuracy performance on the three sets of 2-hidden-layer networks with different number of hidden

Figure 9: Cross-entropy error (Y-axis) on the training (red) and validation (green) sets of a network, called C_0, that is similar to A_1 except for the inputs being corrupted using Gaussian noises instead of dropout.

Network	#Neurons/layer	Train set	Validation set	Test set
A_1	50	93.17%	88.96%	90.34%
B_2	75	94.27%	89.99%	90.69%
D_0	100	95.68%	90.56%	91.52%
D_1	250	99.42%	90.95%	91.95%

Table 6: Accuracy by networks of two hidden layers with different number of neurons per layer.

neurons. We can see that increasing the size of the hidden layers from 50 to 75 and to 100 improved the accuracy up to 1%; unfortunately the performance gaps between training and validation/test sets also got wider, increasing the variances of the bigger networks. With 250 hidden neurons (network D_1), the network achieved 99.42% accuracy on the training set but did not improve much on the independent test set (91.95%). As shown in Figure 10, this network overfitted to the training set at step $6,073,855$ of batch updates. For that many hidden neurons, perhaps a more agressive dropout would be needed to cope with overfitting.

Number of hidden layers: As shown above for 2-hidden-layer networks, increasing number of hidden neurons could help with improving the accuracy performance (though not much unless more agressive ways were used to deal with overfitting). The number of hidden layers is also another hyperparamter

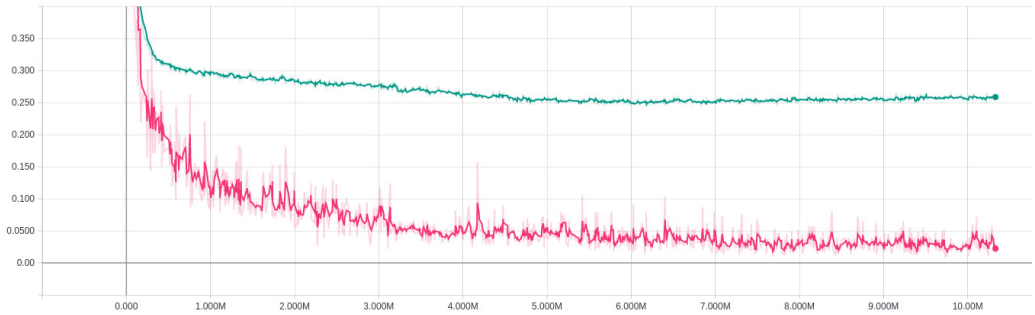

Figure 10: Cross-entropy error (Y-axis) on the training (red) and validation (green) sets of 2-hidden-layer network D_1 which has 250 hidden neurons per layer.

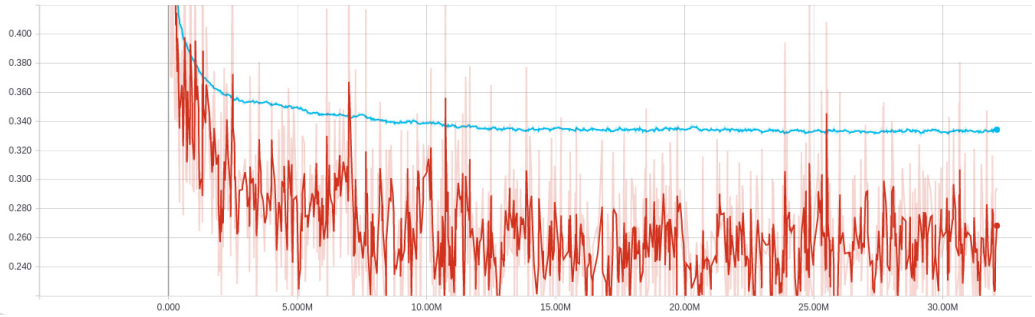

Figure 11: Cross-entropy error (Y-axis) on the training (red) and validation (green) sets of a network, called D_2, that is similar to A_1 but has 1 hidden layer.

that determines the expressiveness of a neural network. With our small training set of about 6900 examples, we might not need much deeper neural network. Figure 11 and 12 shows the learning curves by networks of 1 and 3 hidden layers, each of which has 50 hidden neurons, and Table 7 shows the accuracy performance on the training, validation and testing sets of these networks compared to their counterpart that has 2 hidden layers. We can observe that their performance are quite close with A_1, which has 2 hidden layers, being slightly better than the others on an independent test set.

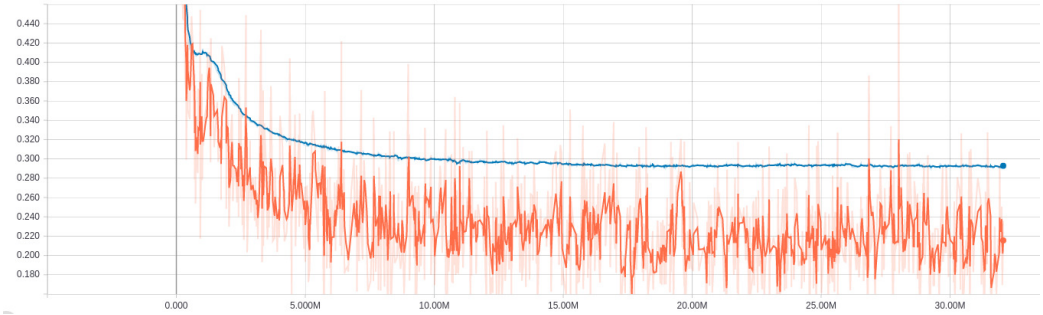

Figure 12: Cross-entropy error (Y-axis) on the training (red) and validation (green) sets of a network, called D_3, that is similar to A_1 but has 3 hidden layers.

Network	#Hidden layers	Train set	Validation set	Test set
D_2	1	93.34%	89.26%	89.43%
A_1	2	93.17%	88.96%	90.34%
D_3	3	92.98%	88.56%	89.69%

Table 7: Accuracy by networks of 1, 2 and 3 hidden layers with 50 hidden neurons per layer.

5 Conclusion

Future work includes considering sparse autoencoder[3] and variational autoencoders [8]

References

[1] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[3] Y.-l. Boureau, Y. L. Cun, et al. Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, pages 1185–1192, 2008.

[4] N. F. Güler, E. D. Übeyli, and I. Güler. Recurrent neural networks employing lyapunov exponents for eeg signals classification. *Expert systems with applications*, 29(3):506–514, 2005.

[5] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[6] E. Kabir, Siuly, and Y. Zhang. Epileptic seizure detection from eeg signals using logistic model trees. *Brain informatics*, 3(2):93–100, 2016.

[7] N. Kannathal, M. L. Choo, U. R. Acharya, and P. Sadasivan. Entropies for detection of epilepsy in eeg. *Computer methods and programs in biomedicine*, 80(3):187–194, 2005.

[8] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] M. Längkvist, L. Karlsson, and A. Loutfi. Sleep stage classification using unsupervised feature learning. *Advances in Artificial Neural Systems*, 2012:5, 2012.

[10] V. P. Nigam and D. Graupe. A neural-network-based detection of epilepsy. *Neurological Research*, 26(1):55–60, 2004.

[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[12] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[13] D. Wulsin, J. Gupta, R. Mani, J. Blanco, and B. Litt. Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement. *Journal of neural engineering*, 8(3):036015, 2011.