Methodology

Daubechies kernel induced convolutional neural network (DbXNet)

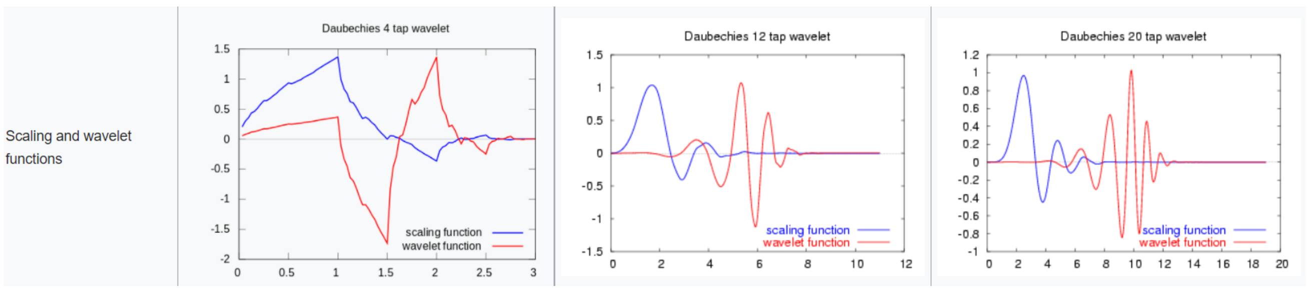
1. Wavelet Transform Head

The architecture has been modified such that for the first convolutional layer, instead of the traditional Xavier normal or Xavier uniform initialisation function [8] a wavelet-based function has been used to initialise the kernel. The motivation behind the use of wavelets in this case was to discriminate between EEG signals obtained from healthy and epileptic individuals during both epileptic as well as seizure free interludes with eyes open and closed. Wavelet transforms [7] have the ability to describe a signal in both time and frequency domains as follows:

$$[W_\psi f](a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \overline{\psi\left(\frac{x-b}{a}\right)} f(x)dx$$

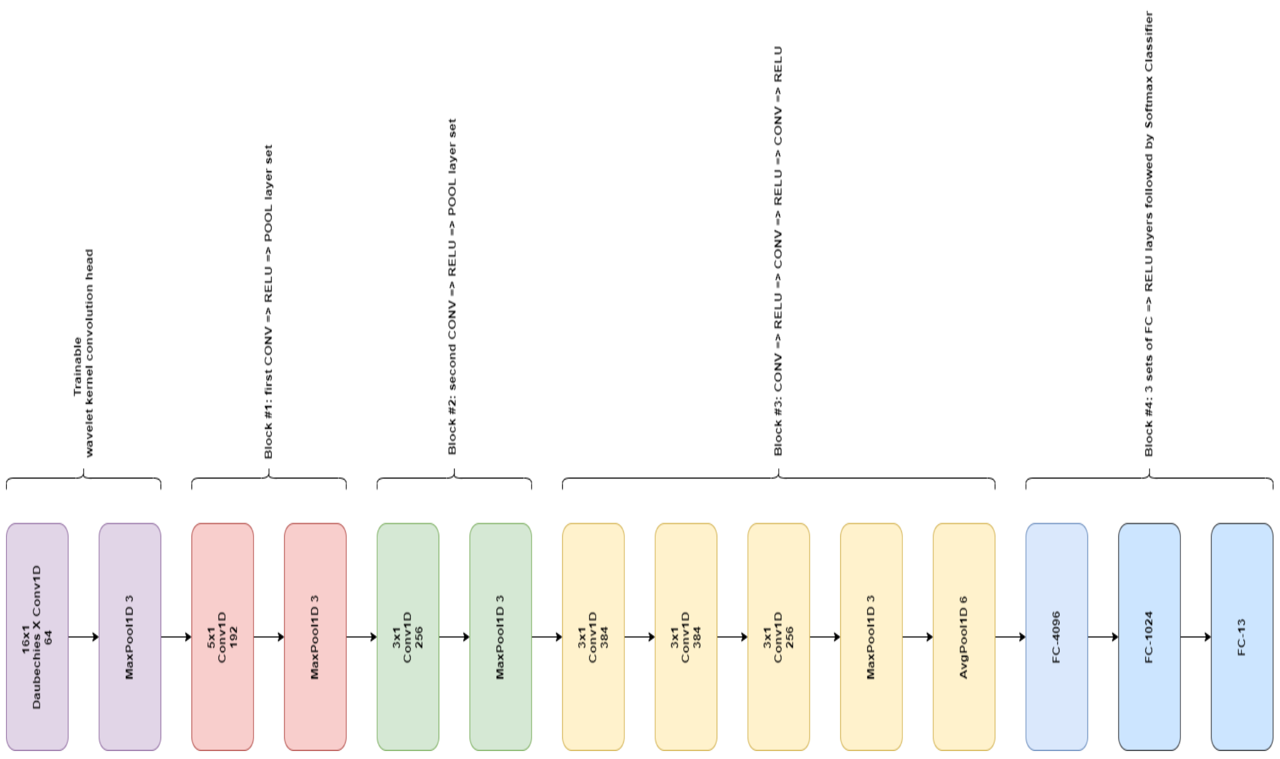
where, $\overline{\psi\left(\frac{x-b}{a}\right)}$ is the selected mother wavelet, a and b being the scaling and shifting factors respectively and f(x) is the input signal. Thus, wavelets have been used to extract the salient features of the different frequency bands of EEG signals.

The Daubechies family of mother wavelets were chosen to be used individually with varying degrees of performance. Classical Back propagation algorithm has been used to update the kernel weights after each epoch.



2. CNN Backbone

A Convolutional Neural Network (CNN/ConvNet) is an improvisation on the traditional ANN that replaces all but the last layer with specialised convolution layers. First introduced by Yann LeCun in 1997 [1], CNNs have since seen widespread adoption in object detection, signal and image classification etc. CNN closely resembles with the natural brain and works on the principle of regularized multi-layer perception. When compared with the other popular feed forward deep learning algorithms such as artificial neural network (ANN), CNN is usually complimented as the superior one because of its end-to-end learning architecture for extracting features and classification purpose. The architecture of a CNN model follows a hierarchical approach, which compositely contains an input layer, consecutive hidden layers in between and an output layer in the end. The hidden layers in the CNN architecture comprises of several convolution and pooling layers. Being sequentially embedded with each other, these operating layers serves the purpose of extracting significant attributes from the signal inputs.


Trainable wavelet kernel convolution head
Block #1: first CONV => RELU => POOL layer set
Block #2: second CONV => RELU => POOL layer set
Block #3: CONV => RELU => CONV => RELU => CONV => RELU
Block #4: 3 sets of FC => RELU layers followed by Softmax Classifier
16x1 Daubechies X Conv1D 64
MaxPool1D 3
5x1 Conv1D 192
MaxPool1D 3
3x1 Conv1D 256
MaxPool1D 3
3x1 Conv1D 384
3x1 Conv1D 384
3x1 Conv1D 256
MaxPool1D 3
AvgPool1D 6
FC-4096
FC-1024
FC-13

a. Alexnet

For our study the CNN backbone we have chosen to implement is the Alexnet. AlexNet, proposed by Alex Krizhevsky [2] is a popular CNN architecture, which has been the winner of ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), 2012. The network has been trained on the ILSVRC database, containing over 1.2 million images corresponding to 1000 different class labels. The AlexNet architetcture contains roughly 650,000 neurons and 60 million parameters. The initial hidden layers of the AlexNet architecture have a depth of 8 layers, containing 5 convolutional layers and 3 max-pooling layers, respectively. The first two convolution layers make use of 96 and 256 numbers of kernels with sizes of $11 \times 11 \times 3$ and $48 \times 5 \times 5$, respectively. A max-pooling layer, using filter of size 3×3, follows each of these two layers. The rest of the convolution layers i.e. third, fourth and the fifth ones are directly connected with each other and use 384, 384, and 256 numbers of kernels with size of the filters being $256 \times 3 \times 3$, $192 \times 3 \times 3$ and $192 \times 3 \times 3$, respectively. Another max-pooling layer, using filter of size 3×3, follows the fifth convolution layer and links the succeeding 2 fully connected layers namely Fc6 and Fc7, containing 4096 neurons each. The fully connected layers produce a high dimensional feature vector, which is utilized as input to the terminating classification layer with softmax activation that is capable of differentiating between images of 1000 data labels. The AlexNet architecture employs the dropout regularization method in classification layer and ReLU activation functions in the convolution layers. For this study, the last fully connected layer i.e. Fc7 was utilized to extract deep neural features from the AlexNet CNN model. A brief structure of AlexNet is shown below in Figure 2

Convolution layers: These layers are considered as the building blocks of a CNN model as it stresses upon heavy computations to produce the initial high dimensional feature maps from the image input. Convolution layers are consisted of a fixed number of independent filters i.e. most commonly known as kernels or edge detectors [3]. These filters are generally two-dimensional arrays of weights, which are sequentially processed to perform convolution on the input data. The choice of several convolution layer parameters such as filter size, filter count, stride, padding etc. are very much important in training process since they heavily affect the performance of the CNN model. The filters are convoluted transversely through the enitre input image on a feed-forward basis to construct the high-level feature maps.

Activation: The convoluted signals are normalised using batch normalisation and passed through a ReLU (Rectified Linear Unit) activation function that suppresses all negative feature elements to 0 thereby improving the feature learning ability of the network.

$C_i' = \mathrm{ReLU}\ (C_i)$

Where ReLU activation function is defined as:

$\mathrm{ReLU}(x) = x \text{ for } x \geq 0$

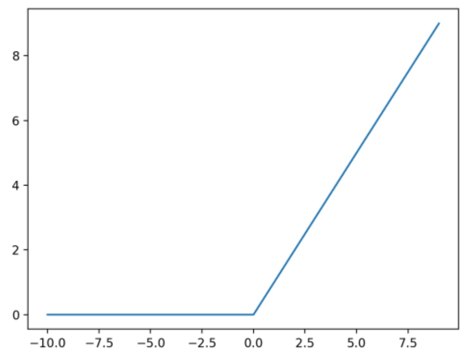


Fig. 3 ReLU function

Pooling layers: In the CNN architecture, pooling layers are merged in succession with the convolution layers to reduce dimension and computation complexity of the network and also to avoid the model overfitting and loss of useful information [4]. The dimension reduction is performed with help of applying non-linear down sampling on the previously generated high dimensional feature output. Usually, a series of non-intersecting rectangular sub-regions are generated from the regions covered by the feature maps and from these sub-regions; pooling operations are done with help of various operators such as maximum pooling, average pooling, global pooling etc. Hence, the spatial volume of the output is reduced and also by reducing the network parameters, the training time of the 2 CNN model is shortened.

Dropout Layer: Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel [5]. During training, some number of layer outputs are randomly ignored or “dropped out.” This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different “view” of the configured layer. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. This conceptualization suggests that perhaps dropout breaks-up situations where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust.

Fully connected layers: The final wrapping in the CNN architecture is termed as the fully connected layers. The purpose of this layer is to produce attributes corresponding to the feature output of the previous pooling layers and differentiate between different labels of the input data. The fully connected layer flattens the feature outputs of the preceding convolution and pooling layers into 1-dimensional feature vectors. A softmax classification layer is also included in the fully connected layer, which allocates class labels to each category of the input data, depending upon scores calculated by the previous operating layers [6].

3. <u>Training Procedure</u>

The Deep Learning framework proposed here has been written completely in Anaconda distribution of the Python 3.7 programming language using the libraries TensorFlow and Keras and has a depth of 29 layers This specific implementation of the DbXNet architecture has been trained, validated and tested on the PQD Dataset over an NVIDIA GTX 1050Ti GPU with 4GB of VRAM .

Gradient descent learning has been used as the optimization function for the Back propagation algorithm to update the wavelet kernel weights with the minimization of the loss function. The loss function used is 'categorical cross-entropy'. The training epoch is set to 100 with an initial learning rate of 0.001 has been chosen with a momentum of 0.9. A polynomial function

$$\alpha = LRinit x(1-(e/maxEpochs))) \qquad (16)$$

has been used as the learning rate scheduler where, α is the learning rate for epoch e, LRinit is the initial learning rate and maxEpochs is the total number of training epochs. A minibatch of size 20 has been chosen to train the proposed network. K-Fold Stratified cross validation has been applied for testing and training the model with the number of folds chosen as 4

i. Optimal Initial Learning Rate

The learning rate is a very important hyper-parameter for training any neural network. Until recently the optimal learning rate was chosen using a trial-and-error method. However, we have used the method proposed by Krishna Katyal on his GitHub project on Malaria Detection [7]. This method involves doing a trial run to train the neural network using a low learning rate, but then increasing it exponentially with each batch while recording the loss for every value of the learning rate. We then plot loss against learning rate. The optimum value of learning rate is the point where the learning rate is highest, and the slope of the loss vs learning rate curve is negative (i.e., loss is descending).

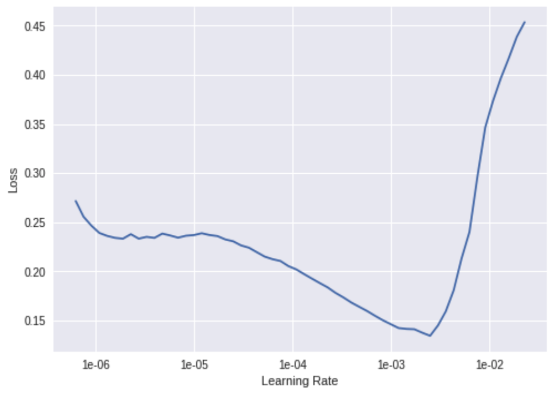


Fig.9.Loss vs Learning Rate curve

Based on the curve plotted above the optimised initial learning rate was chosen to be 0.001

ii. K-Fold Cross Validation

The K-fold cross validation is a statistical method used to estimate the skill of machine learning models. The method consists of a parameter k that represents the number of groups that a given data sample will be split into. The general procedure is as follows:
The dataset is shuffled randomly and split into k groups. Each group is taken as the test set while the remaining k-1 groups are training set. The model is then fitted on training set and evaluated on test set. The evaluation score is retained while the model is discarded and the skill of the model is summarised using the retained evaluation scores. This procedure is

repeated for all the groups. Thus K-fold cross validation ensures that every observation from the original dataset has the chance of appearing in both training and test set and is a very useful technique for assessing model effectiveness, mitigating the overfitting problem and determining hyper parameters that will result in least test error. In this instance a 4-fold cross validation has been performed (i.e., k=4).

iii. Back Propagation in Wavelet Kernels

To calculate the error $\delta i,k$ at the output layer the component wise difference of the target value and the output value is taken.

$\delta i,k = Ti,k - Oi,k \qquad \forall k$

where, $\delta i,k$, Ti,k and Oi,k are the k-th component of i-th error vector, target vector and output vector respectively, and the cross-entropy loss function is defined as:

$J = (-1/k) \sum [i=1: k] \{Ti . \ln (Oi)\}$

The error after each forward pass is back-propagated from the output layer to the previous layers by node updation of each layer followed by updation of the weights of the kernels in every layer. This procedure is known as the back-propagation (BP) algorithm and is based on the stochastic gradient descent technique. The BP process consists of the following two steps:

a) Node Updation

The errors calculated from each neuron of the Dense layer and propagated from the output layer to the input layer through the Dense layer

$\delta fc = \sum [i=1: N]\ \delta i,k\ . w$

b) Weights and Biases Updation

Weights and biases are updated by the minimization of the cross-entropy loss function via gradient descent principle [6]

$$\left.\begin{aligned} w_{mn} &\leftarrow w_{mn} - \alpha \frac{\partial J}{\partial w_{mn}} \\ b_k &\leftarrow b_k - \alpha \frac{\partial J}{\partial b_k} \end{aligned}\right\}$$

where, wmn is the weight connected between neuron m of (k-1) th layer to neuron n of k-th layer and α is the learning rate. The value of is different for non-linear layers. From Output to the Dense layer, the non-linearity is SoftMax activation function. Hence, weights are updated as:

$wmn \leftarrow wmn + \alpha(Tn - On).On\,(1 - On)O\mathrm{m}$

where, On is the output obtained from n neuron at k-th layer and Om is the output obtained from m neuron at (k – 1)-th layer. Since the max- pooling layers are linear, the weights of the wavelet kernels are updated as:

$wmn \leftarrow wmn + \alpha\,(Tn - On).\ Om$

Since, the weights are updated according to the BP algorithm and the updation rule is based on the gradient, there is a possibility of the network being trapped in a local minima. In order to reduce this possibility, a momentum value of 0.9 is added and the weight adaptation rule becomes:

$wmn \leftarrow wmn + \alpha(Tn - On).Om + \gamma\Delta wmm$

where, γ is the momentum value and Δwmn is the weight difference from node m to n. However, since there is no guarantee that the trapping to local minima can be completely avoided with this method, better initialization of the layer filters is a key factor in better learning or training of the network. By using wavelets that match the shape of the input signal in the first convolution layer, the kernels are better initialized. Therefore, they are more likely to reach the optimum value quicker (i.e., in lesser iterations) and achieve values for higher performance metrics like accuracy (ACC), specificity (SPE), sensitivity (SEN), positive predictive value (PPV) and negative predictive value (NPV). Similarly using wavelets that are not a good match for the input signals the kernels are poorly initialised resulting in higher loss and poor performance metrics as has been found in experimentation. Hence, proper selection of wavelet kernels and their combinations are necessary for better performance of the proposed neural network.

Reference

1. Y. Lecun and Y. Bengio, Convolutional Networks for Images, Speech and Time Series
2. Krizhevsky A, Sutskever I and Hinton G E 2012 Imagenet classification with deep convolutional neural networks: Proc. Advances in Neural Information Processing Systems (Lake Tahoe, USA, December.2012) pp 1097–1105
3. Zhang C, Cheng X, Liu J, He J and Liu G, 2018 Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status, J. Control Sci. Eng. 2018 1–9
4. Sengur A, Akbulut Y, Guo Y and Bajaj V 2017 Classification of amyotrophic lateral sclerosis disease based on convolutional neural network and reinforcement sample learning algorithm, Health Inf. Sci. Syst. 5(1) 1–7
5. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R, 2014 Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 2014
6. Sengur A, Gedikpinar M, Akbulut Y, Deniz E, Bajaj V and Guo Y 2017 27 DeepEMGNet: an application for efficient discrimination of ALS and normal EMG signals: Proc. International Conference Mechatronics (Australia, February.2017) (Springer, Cham) pp 619–625
7. P.M. Bentley and J.T.E. McDonnell,” Wavelet transforms: an introduction”, Electronics & Communication Engineering Journal August 1994
8. X. Glorot, and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proc. 13th Int. Conf. Art. Intel. Stat. Sardinia, Italy, May 13-15, 2010, pp. 249-256.