Orthogonal Features Based EEG Signals Denoising Using Fractional and Compressed One-Dimensional CNN Autoencoder

Subham Nagar and Ahlad Kumar, *Senior Member, IEEE*

***Abstract*— This paper presents a fractional one-dimensional convolutional neural network (CNN) autoencoder for denoising the Electroencephalogram (EEG) signals which often get contaminated with noise during the recording process, mostly due to muscle artifacts (MA), introduced by the movement of muscles. The existing EEG denoising methods make use of decomposition, thresholding and filtering techniques. In the proposed approach, EEG signals are first transformed to orthogonal domain using Tchebichef moments before feeding to the proposed architecture. A new hyper-parameter (α) is introduced which refers to the fractional order with respect to which gradients are calculated during back-propagation. It is observed that by tuning α, the quality of the restored signal improves significantly. Motivated by the high usage of portable low energy devices which make use of compressed deep learning architectures, the trainable parameters of the proposed architecture are compressed using randomized singular value decomposition (RSVD) algorithm. The experiments are performed on the standard EEG datasets, namely, Mendeley and Bonn. The study shows that the proposed fractional and compressed architecture performs better than existing state-of-the-art signal denoising methods.**

***Index Terms*— EEG signal denoising, convolutional neural networks, autoencoder, Tchebichef moments, compression.**

I. Introduction

ELECTROENCEPHALOGRAM (EEG) is the recording of electrical activity inside the human brain [1]. It is during the recording process that the EEG signals often get contaminated with various types of artifacts, due to muscle activity, eye movements and heart rhythms, which are measured by electromyogram (EMG), electrooculogram (EOG) electrocardiogram (ECG) signals, respectively [2]. Among these, the Electromyogram/muscle artifact (EMG/MA) is one such type of noise that is generally found to bes challenging to eliminate, mainly due to its high amplitude and its broad frequency and anatomical distributions [3].

Different approaches have been reported in the existing literature to remove muscle artifacts (MA) from the contaminated EEG signals. Numerous decomposition techniques like wavelet transform [4], empirical mode decomposition (EMD) [5], ensemble empirical mode decomposition (EEMD) [6] have also been employed to achieve good results. Canonical correlation analysis has also been successfully used with the above decomposition techniques [7], [8]. Sweeny *et al.* [7] has used both EMD and EEMD with canonical correlation analysis. Recently, variational mode decomposition (VMD) was proposed [9], to yield superior results.

The application of machine learning and deep learning architectures have also been found to be very effective in the area of EEG signals [10]. Temporal spectral based fusion network [11] and attention based networks [12] have been used in EEG domain. Denoising auto-encoder produced a major breakthrough in the problem of signal denoising [13]. However, the effect of compression was not studied on signal denoising. Pruning was proposed in [14] to remove the least important trainable weights from the neural network. Calculation of the low-rank approximation of weight matrices simultaneously reduced storage and time complexity during the training and testing phases [15]. The concept of randomized singular valued decomposition (RSVD) was introduced in [16], which represented a faster way of calculating low-rank approximations. Its effectiveness was also explored in [17]. In this paper, the RSVD algorithm is used for compressing the trainable weight matrices of the proposed architecture.

The discrete cosine transform (DCT) coefficients of the input images was used with neural networks for achieving high speed deep learning architectures [18] and also for improving it's image denoising performance [19]. Recently, orthogonal moment domain, has also been recently explored to address common problems in image processing [20]. One such kind of orthogonal moments, namely, Tchebichef moments (TM) exhibit an essential property of energy compaction, that led to promising results in denoising images [21]. This research finding has motivated us to exploit the advantage of feeding these TM based orthogonal features to the

Manuscript received 6 October 2021; revised 1 March 2022 and 9 July 2022; accepted 21 August 2022. Date of publication 24 August 2022; date of current version 2 September 2022. *(Corresponding author: Ahlad Kumar.)*

Subham Nagar is with DA-IICT, Gandhinagar, Gujarat 382007, India (e-mail: subhamnagar@gmail.com).

Ahlad Kumar is with the Department of Information and Communication Technology, DA-IICT, Gandhinagar, Gujarat 382007, India (e-mail: ahlad_kumar@daiict.ac.in).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNSRE.2022.3201197.

Digital Object Identifier 10.1109/TNSRE.2022.3201197

proposed one-dimensional convolution neural network (CNN) architecture.

Fractional calculus is now getting popular for solving significant problems in the image processing domain like image denoising [22] and texture enhancement [23]. It has also found a place in the neural networks where gradients are calculated using differentiation with respect to a particular fractional order (α) and has given a performance boost in classification problems as compared to conventional neural networks [24]. We have also used fractional calculus in the back-propagation phase of the proposed architecture for improved performance in the case of EEG signal denoising.

This paper is organized as follows. Section II provides some mathematical preliminaries that are useful in understanding the underlying mechanics of our architecture. Section III and IV propose the workflow of our fractional one-dimensional CNN and it's compressed form respectively. Experimental results are provided in Section V that contains the details of the datasets used, data preparation for training, performance metrics and evaluation of our proposed model under compression and in frequency domain, followed by discussion on the results obtained. Section VI concludes this work.

II. Mathematical Preliminaries

A. Tchebichef Moments (TM)

Let $x(n)$ be an EEG signal with $n = 1, 2, \ldots, \mathcal{N}$. The relationship between the noisy signal $y(n)$ and the original signal $x(n)$ corrupted by noise is given as follows:

$$y(n) = x(n) + \zeta(n) \tag{1}$$

where $\zeta(n)$ is the muscle artifacts (MA) noise. This paper proposes a deep learning architecture which recovers an estimate of the original signal from its noisy observation $y(n)$.

The Tchebichef moments of order p for a signal $x(n)$ of length $\mathcal{N}$ samples is given by [25]:

$$T_p(x) = \sum_{n=0}^{\mathcal{N}-1} t_p(n;\mathcal{N})x(n) \tag{2}$$

with $p, n = 0, 1, 2 \ldots . \mathcal{N}-1$ and $x(n)$ is of dimension $1 \times N$. For simplicity, $t_p(n)$ has been used to represent $t_p(n;\mathcal{N})$ which is the orthonormal Tchebichef polynomials given by

$$t_p(n) = \beta_1(2n + 1 - \mathcal{N})t_{p-1}(n) + \beta_2 t_{p-2}(n) \tag{3}$$

where the variables β_1 and β_2 are given by

$$\beta_1 = \frac{1}{p}\sqrt{\frac{4(p^2-1)}{\mathcal{N}^2 - p^2}} \tag{4}$$

$$\beta_2 = \frac{1-p}{p}\sqrt{\frac{2p+1}{2p+3}}\sqrt{\frac{\mathcal{N}^2-(p-1)^2}{\mathcal{N}^2-p^2}} \tag{5}$$

The initial conditions for the recurrence relations are

$$t_0(n) = \frac{1}{\sqrt{\mathcal{N}}} \tag{6}$$

and

$$t_1(n) = (2n + 1 - \mathcal{N})\sqrt{\frac{3}{\mathcal{N}(\mathcal{N}^2-1)}} \tag{7}$$

The set of TMs upto order p in matrix form is given as

$$T_p(X) = XQ^T \tag{8}$$

where $X = [x(0),\ x(1),\ x(2), \ldots,\ x(\mathcal{N}-1)]$ and

$$Q = \begin{bmatrix} t_0(0) & \ldots & t_0(\mathcal{N}-1) \\ \vdots & \ddots & \vdots \\ t_{p-1}(0) & \ldots & t_{p-1}(\mathcal{N}-1) \end{bmatrix} \tag{9}$$

Here, Q is the Tchebichef polynomial matrix upto order p. The original one-dimensional signal X can be reconstructed from the set of Tchebichef moments using the following equation

$$X = T_p(X)Q \tag{10}$$

B. Compression Using RSVD

The compression of the kernel and weight matrices used in the proposed architecture is carried out using low rank approximation of these matrices. It is done using the RSVD technique, which decomposes the original matrix $A \in \mathbb{R}^{n\times m}$ into a smaller randomized subspace $B \in \mathbb{R}^{c\times m}$, where $c < n$. For kernel matrix $K \in \mathbb{R}^{n\times c\times f}$, where n, c and f refer to the number of filters, number of channels and feature dimension respectively, we reshape it into a $2D$ matrix A of the form $\mathbb{R}^{n\times m}$, where $m = c * f$.

For calculating the low rank approximation, let $O \in \mathbb{R}^{m\times(r+p)}$ be a normally distributed random matrix, where r is the rank to be approximated, p denotes the number of additional projections such that $r + p < n$. We define Q_i as the orthogonal basis after i iterations, where $i = 1, 2, 3 \ldots, k$ and k denotes the number of subspace iterations. The value of Q_0 is set using the following equation

$$Q_0 = AO \tag{11}$$

The recurrence relation for calculating the orthogonal basis Q_i is given by

$$G_\mathrm{i} = qr(A^\mathrm{T}Q_{\mathrm{i}-1}) \tag{12}$$

$$Q_\mathrm{i} = qr(AG_\mathrm{i}) \tag{13}$$

where $qr()$ is the function for the QR decomposition operation which factorizes a matrix into an orthogonal matrix and an upper triangular matrix. Here, we just take the orthogonal matrix part. The above recurrence equations are applied over k iterations, after which, we get the value Q_k. The condensed matrix B can be calculated as

$$B = Q_\mathrm{k}{}^\mathrm{T}A \tag{14}$$

The SVD decomposition of this condensed matrix is

$$[U_r, S_r, V_r] = \mathrm{SVD}(B) \tag{15}$$

$$U_\mathrm{r} = Q_\mathrm{k}U_\mathrm{r} \tag{16}$$

where, $U_r \in \mathbb{R}^{n\times r}$, $V_r \in \mathbb{R}^{m\times r}$ are the matrices with orthonormal columns and $S_r \in \mathbb{R}^{r\times r}$ is a diagonal matrix.

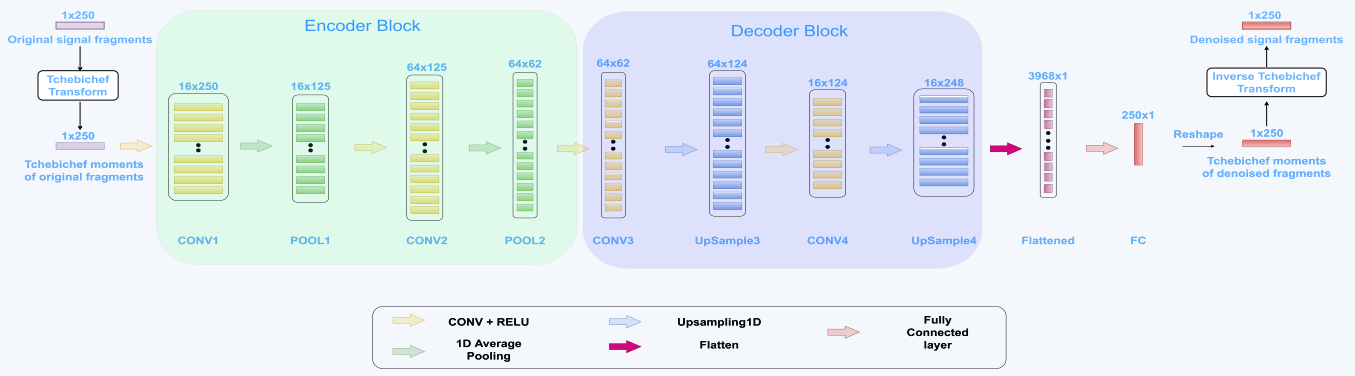

Fig. 1. Architecture of the proposed fractional CNN auto-encoder.

C. Fractional Order Processing

Unlike the integer order derivatives, various definitions have been proposed for fractional order derivatives. The three most commonly used fractional order derivatives are, namely, Grunwald Letnikov (G-L), Riemann-Liouville (R-L) and Caputo derivatives [24]. We have used Caputo fractional derivative (CFD) of a function $f(u)$ with order α, defined as follows:

$$ {}_{u_0}^{C}D^{\alpha}{}_{u} f(u) = \frac{1}{\Gamma(n-\alpha)} \int_{u_0}^{v} \frac{f^{(n)}(y)}{(u-y)^{(1+\alpha-n)}} dy \qquad (17) $$

where $n-1 < \alpha < n$, $n \in \mathbb{N}^{+}$, u_0 is the initial value and $\Gamma(\cdot)$ denotes the Gamma function. The CFD is found to be consistent with the integer order derivatives used in neural networks, because of which this derivative is applied in several engineering problems [24]. This motivated us to employ it during the back-propagation of our proposed model. Let α be the fractional order for which the derivative needs to be calculated and $f(x) = (x-a)^k$ be a polynomial function of degree k. The Caputo fractional derivative is given by [26]:

$$ \frac{d^{\alpha} f(x)}{dx^{\alpha}} = \frac{\Gamma(k+1)(x-a)^{k-\alpha}}{\Gamma(k-\alpha+1)}) \qquad (18) $$

For simplicity of the notation, the fractional derivative $\frac{\partial^{\alpha} f(x)}{\partial x^{\alpha}}$ is denoted as $D_x^{\alpha} f(x)$ and will be used in calculating gradients of the proposed architecture.

III. Proposed Architecture

The proposed fractional CNN based architecture for denoising EEG signals is shown in Fig. 1. The model is based on the encoder-decoder architecture. The encoding of the EEG signal is carried out and the information is represented in the compressed form as latent vectors. This is followed by up-sampling (decoder) operation that recovers the information present in the EEG signal from the latent space. This can be observed from Fig. 1, where each of the first two convolutional layers followed by average pooling layers constitutes the encoder block while each of the last two convolutional layers followed by up-sampling layers constitutes the decoder block. Here, the up-sampling layers are used for recovering structural details present in the EEG signals.

The original EEG signal fragments are transformed into TMs (orthogonal) space $T_{\mathcal{N}}(\mathbf{X})$ using Eq. 8 where $\mathbf{X}$ is the original signal fragment of dimension $\mathcal{N}$. Similarly, the transformation of the noisy signal $T_{\mathcal{N}}(y)$ has also been done, which is then fed as an input to the first convolutional layer of the proposed architecture. The architecture has four convolutional layers (CONV), two average pooling layers, two upsampling layers and a flattened layer, which is connected to the fully connected (FC) layer. The first convolutional layer (CONV1) has 16 filters. The next two convolutional layers (CONV2, CONV3) have 64 filters each while CONV4 has 16 filters. Rectified linear units (ReLU) have been employed as activation functions for convolutional hidden layers. There are 250 neurons in the fully connected layer (FC). All the convolutional layers are having kernel of dimension 1×3 and kernel stride is taken as 1. The average pooling layers have kernel dimension of 2 and a stride of 2, while the up-sampling layers have the up-sampling factor as 2. The padding ensures the output dimension to be same as that of the input. Next, the workflow of the architecture that involves forward and proposed fractional backward propagation will be discussed.

A. Forward Propagation

1) Convolutional Layer: The input-output relationship for the convolutional layer of the architecture is given as

$$ S^{[i]}_{m,j} = \sum_{n=0}^{N_C^{[i-1]}-1} \sum_{p=0}^{F^{[i]}-1} I^{[i-1]}_{n,j+p} K^{[i]}_{m,n,p} + b^{[i]}_{m} \qquad (19) $$

where $m = 0 \ldots N_F^{[i]} - 1$, $j = 0 \ldots N_W^{[i]} - 1$. Here, $I^{[i-1]}$ is the input of dimension $N_C^{[i-1]} \times N_W^{[i-1]}$, being fed to the i^{th} convolutional layer with $N_C^{[i-1]}$ representing the number of channels and $N_W^{[i-1]}$ being the feature dimension. For the i^{th} convolutional layer, we have the trainable kernel $K^{[i]}$ of size $N_F^{[i]} \times N_C^{[i-1]} \times F^{[i]}$, where $F^{[i]}$ is the kernel filter dimension and $N_F^{[i]}$ denotes the number of kernel filters. The matrix $b^{[i]}$ denotes the bias for this layer of dimension $N_F^{[i]} \times 1$. $S^{[i]}$ is the output of the convolution layer and is of size $N_F^{[i]} \times N_W^{[i]}$, where the output feature dimension $N_W^{[i]}$ is given by

$$ N_W^{[i]} = (N_W^{[i-1]} - F^{[i]} + 2 * g^{[i]}) + 1 \qquad (20) $$

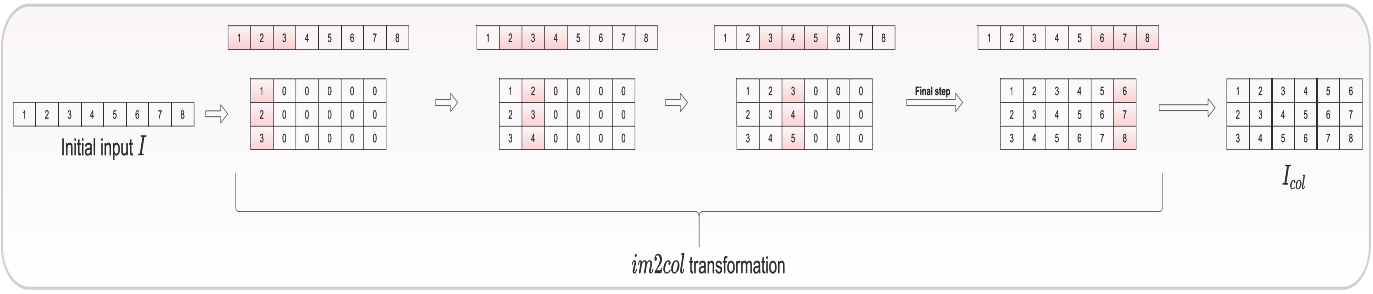

Fig. 2. Illustration of *im2col* transformation on 1D input.

with $g^{[i]}$, denoting the padding size. In this paper, Tchebichef vector of the noisy signal $T_{\mathcal{N}}(y)$ is taken as the input features denoted by $I^{[0]}$ (Eq. 19). This is fed to the first convolutional layer of the architecture, with $N_C^{[0]} = 1$ and $N_W^{[0]} = \mathcal{N}$.

For faster implementation of the above approach, we use the method of matrix multiplication to represent convolution operation given in Eq. 19. For this, input $I^{[i-1]}$ is converted into matrix form using the following transformation:

$$I_{col}^{[i-1]} = im2col(I^{[i-1]}) \tag{21}$$

where the dimension of $I_{col}^{[i-1]}$ is $(N_C^{[i-1]} \times F^{[i]} \times N_W^{[i]})$, with $N_W^{[i]} = (N_W^{[i-1]} - F^{[i]} + 2 * g^{[i]}) + 1$

Here, $im2col$ refers to the technique in which input of size $1 \times F^{[i]}$ is taken and stack it in the form of columns of a matrix. The pictorial illustration of $im2col$ has been shown in Fig. 2. Now, Eq. (19) gets modified as

$$S^{[i]} = K^{[i]}(*)I_{col}^{[i-1]} + b^{[i]} \tag{22}$$

where (*) denotes the matrix multiplication, $b^{[i]}$ is the bias matrix of size $(N_F^{[i]} \times 1)$.

Each convolutional layer output $S^{[i]}$ is fed as an input to the rectified linear unit activation function (ReLU), which gives $C^{[i]}$ as the output governed by the following expression

$$C_{m,j}^{[i]} = ReLU(S_{m,j}^{[i]}) \tag{23}$$

2) Average Pooling Layer: Average pooling operation is applied on the activated feature maps of the i^{th} convolutional layer $C^{[i]}$ having dimension of $N_F^{[i]} \times N_W^{[i]}$. In this study, pooling filter size $P_f^{[i]} = 2$ and stride $P_s^{[i]} = 2$ is taken. The pooling output $P^{[i]}$ of dimension $N_F^{[i]} \times N_P^{[i]}$ is given as

$$P_{m,k}^{[i]} = \frac{C_{m,2k}^{[i]} + C_{m,2k+1}^{[i]}}{2} \tag{24}$$

where $m = 0 \ldots N_F^{[i]} - 1$ and $k = 0 \ldots N_P^{[i]} - 1$ with $N_P^{[i]} = \frac{N_W^{[i]}}{2}$.

3) Up-Sampling Layer: Up-sampling operation increases the resolution of the activated feature maps, i.e, the i^{th} convolutional layer $C^{[i]}$. The output of the up-sampling layer $U^{[i]}$ is defined as

$$U_{m,j}^{[i]} = C_{m,\lfloor j/2 \rfloor}^{[i]} \tag{25}$$

where $m = 0 \ldots N_F^{[i]} - 1$ and $j = 0 \ldots N_U^{[i]} - 1$ with $N_U^{[i]} = 2 * N_W^{[i]}$ as the up-sampling factor $U_f^{[i]} = 2$. The dimension of $U^{[i]}$ is $N_F^{[i]} \times N_U^{[i]}$.

4) Flattened Layer: Before we feed the features to the FC layer, it needs to be flattened into a one-dimensional format. This process can be represented by the following equation

$$\mathcal{F} = flatten(R) \tag{26}$$

where R is the input for this layer and $\mathcal{F}$ is the flattened output which can be used in the FC layer.

5) FC Layer: Now, the forward propagation can be represented by

$$\hat{T}_{\mathcal{N}}(x) = W(*)\mathcal{F} + B \tag{27}$$

where $\mathcal{F}$, W and B denote the output of the flattened layer, weight matrix and bias respectively. Here, $\hat{T}_{\mathcal{N}}(x)$ is the estimated denoised signal which will be used in the formulation of the loss function discussed next.

B. Loss Function

The proposed loss function for the fractional CNN auto-encoder is given as

$$L = \frac{1}{2\mathcal{M}} \sum_{i=0}^{\mathcal{M}-1} \left(\hat{T}_{\mathcal{N}}^{(i)(x)} - T_{\mathcal{N}}^{(i)(x)} \right)^2 + \frac{\lambda}{2} \left(\sum_{j=1}^{E} \left\| K^{[j]} \right\|_2^2 + \|W\|_2^2 \right) \tag{28}$$

where, $\mathcal{M}$ represents the number of training samples, E is the number of convolutional layers and λ represents the regularization parameter. For back-propagation, the derivative of loss with respect to $\hat{T}_{\mathcal{N}}(x)$ is calculated as

$$\frac{\partial L}{\partial \hat{T}_{\mathcal{N}}(x)} = \frac{1}{\mathcal{M}} \sum_{i=0}^{\mathcal{M}-1} \left(\hat{T}_{\mathcal{N}}^{(i)(x)} - T_{\mathcal{N}}^{(i)(x)} \right) \tag{29}$$

Eq. (29) is used during back-propagation process which is discussed next.

C. Fractional Back-Propagation

The proposed back-propagation technique comes with the advantage in the form of an extra hyper-parameter, i.e, fractional order α which can be tuned to obtain the best denoising performance and also plays an important role in training the architecture.

1) FC Layer: The value obtained using Eq. (29) is used as input to the FC layer. Using Eqs. (27) and (28), the fractional gradients of weights, i.e, $D_W^\alpha L$ is calculated as follows:

$$D_W^\alpha L = \left(\frac{\partial L}{\partial \hat{T}_{\mathcal{N}}(x)}\right) D_W^\alpha \hat{T}_{\mathcal{N}}(x) + \frac{\lambda}{2} D_W^\alpha W^2 \tag{30}$$

Using Eq. (18), the fractional gradients present on the right hand side of Eq. (30) are calculated as follows

$$D_W^\alpha \hat{T}_{\mathcal{N}}(x) = \mathcal{F}\frac{W^{1-\alpha}}{\Gamma(2-\alpha)} \tag{31}$$

$$\frac{\lambda}{2} D_W^\alpha W^2 = \lambda \frac{W^{2-\alpha}}{\Gamma(3-\alpha)} \tag{32}$$

where, $\Gamma(\cdot)$ denotes the gamma function. Substituting these values obtained in Eq. (30) results in

$$D_W^\alpha L = \left(\frac{\partial L}{\partial \hat{T}_{\mathcal{N}}(x)}\right) \mathcal{F}\frac{W^{1-\alpha}}{\Gamma(2-\alpha)} + \lambda \frac{W^{2-\alpha}}{\Gamma(3-\alpha)} \tag{33}$$

Similarly, the fractional gradient with respect to the bias B is given by

$$D_B^\alpha L = \frac{\partial L}{\partial \hat{T}_{\mathcal{N}}(x)} D_B^\alpha \hat{T}_{\mathcal{N}}(x) = \frac{\partial L}{\partial \hat{T}_{\mathcal{N}}(x)} \frac{B^{1-\alpha}}{\Gamma(2-\alpha)} \tag{34}$$

2) Flattened Layer: This layer flattens the input during forward propagation as mentioned in Eq.(26). So during backward propagation, we can reshape the output gradient $\frac{\partial L}{\partial \mathcal{F}}$ in the shape of $R^{[i]}$ to get the input gradient $\frac{\partial L}{\partial R}$

3) Up-Sampling Layer: For backpropagating the gradients, the successive gradients from $\frac{\partial L}{\partial U^{[i]}}$ are added and assigned to each element in $\frac{\partial L}{\partial C^{[i]}}$. This can be represented using the following equation:

$$\frac{\partial L}{\partial C_{m,j}^{[i]}} = \frac{\partial L}{\partial U_{m,2j}^{[i]}} + \frac{\partial L}{\partial U_{m,2j+1}^{[i]}} \tag{35}$$

where the up-sampling factor $U_f^{[i]} = 2$. The dimension of $\frac{\partial L}{\partial C^{[i]}}$ is $N_F^{[i]} \times N_W^{[i]}$ where $N_W^{[i]} = N_U^{[i]}/2$. The gradient $\frac{\partial L}{\partial C^{[i]}}$ for the m^{th} feature map can also be represented in a matrix form as in (36), shown at the bottom of the page, where, $\frac{\partial L}{\partial U_{m,k}^{[i]}}$ refers to the k^{th} element of m^{th} feature map in $\frac{\partial L}{\partial U^{[i]}}$.

4) Average Pooling Layer: Similar to the back-propagation for Up-sampling layer, $\frac{\partial L}{\partial C^{[i]}}$ is calculated from the pooling gradient $\frac{\partial L}{\partial P^{[i]}}$. Considering the fact that $P^{[i]}$ is the average pooling output, here the operation will be slightly different. Each element in $\frac{\partial L}{\partial P^{[i]}}$ is divided by the pooling size $P_f^{[i]}$ and proportionally back-propagate the error gradients to the input. This can be represented using the following equation:

$$\frac{\partial L}{\partial C_{m,j}^{[i]}} = \frac{1}{2}\frac{\partial L}{\partial P_{m,\lfloor j/2 \rfloor}^{[i]}} \tag{37}$$

where, $P_f^{[i]} = 2$. The gradient $\frac{\partial L}{\partial C^{[i]}}$ for the m^{th} feature map calculated in Eq. (37) can also be represented in a matrix form as in (38), shown at the bottom of the page, where, $\frac{\partial L}{\partial P_{m,k}^{[i]}}$ refers to the k^{th} element of m^{th} feature map in $\frac{\partial L}{\partial P^{[i]}}$. The dimension of $\frac{\partial L}{\partial C^{[i]}}$ is $N_F^{[i]} \times N_W^{[i]}$, where $N_W^{[i]} = 2 * N_P^{[i]}$.

5) Convolutional Layer: In this layer, backward propagation of the errors is carried out to calculate the fractional gradients for the kernel $D_{K^{[i]}}^\alpha L$ and bias $D_{b^{[i]}}^\alpha L$ matrices. The output gradient $\frac{\partial L}{\partial C^{[i]}}$ is used to calculate these gradients. Using Eqs. (22) and (28) we obtain

$$D_{K^{[i]}}^\alpha L = \frac{\partial L}{\partial S^{[i]}} D_{K^{[i]}}^\alpha S^{[i]} + \frac{\lambda}{2} D_{K^{[i]}}^\alpha (K^{[i]})^2 \tag{39}$$

$$D_{b^{[i]}}^\alpha L = \frac{\partial L}{\partial S^{[i]}} D_{b^{[i]}}^\alpha S^{[i]} \tag{40}$$

Here, the gradient $\frac{\partial L}{\partial S^{[i]}}$ can be calculated using element-wise multiplication of $\frac{\partial L}{\partial C^{[i]}}$ and $\frac{\partial C^{[i]}}{\partial S^{[i]}}$ and is given as

$$\frac{\partial L}{\partial S^{[i]}} = \frac{\partial L}{\partial C^{[i]}} \frac{\partial C^{[i]}}{\partial S^{[i]}} \tag{41}$$

Using Eq. (23), the value of $\frac{\partial C^{[i]}}{\partial S^{[i]}}$ can be obtained as follows

$$\frac{\partial C^{[i]}}{\partial S^{[i]}} = \begin{cases} 1, & \text{if } S^{[i]} > 0 \\ 0, & \text{if } S^{[i]} \leq 0 \end{cases} \tag{42}$$

Substituting the value of $\frac{\partial C^{[i]}}{\partial S^{[i]}}$ in Eq. (41), results in the following expression

$$\frac{\partial L}{\partial S^{[i]}} = \begin{cases} \frac{\partial L}{\partial C^{[i]}}, & \text{if } S^{[i]} > 0 \\ 0, & \text{if } S^{[i]} \leq 0 \end{cases} \tag{43}$$

$$\frac{\partial L}{\partial C_m^{[i]}} = \left[\frac{\partial L}{\partial U_{m,0}^{[i]}} + \frac{\partial L}{\partial U_{m,1}^{[i]}} \; \frac{\partial L}{\partial U_{m,2}^{[i]}} + \frac{\partial L}{\partial U_{m,3}^{[i]}} \cdots \frac{\partial L}{\partial U_{m,W_u^{[i]}-2}^{[i]}} + \frac{\partial L}{\partial U_{m,W_u^{[i]}-1}^{[i]}}\right] \tag{36}$$

$$\frac{\partial L}{\partial C_m^{[i]}} = \frac{1}{2}\left[\frac{\partial L}{\partial P_{m,0}^{[i]}} \; \frac{\partial L}{\partial P_{m,0}^{[i]}} \; \frac{\partial L}{\partial P_{m,1}^{[i]}} \; \frac{\partial L}{\partial P_{m,1}^{[i]}} \cdots \frac{\partial L}{\partial P_{m,N_W^{[i]}-1}^{[i]}} \; \frac{\partial L}{\partial P_{m,N_P^{[i]}-1}^{[i]}}\right] \tag{38}$$

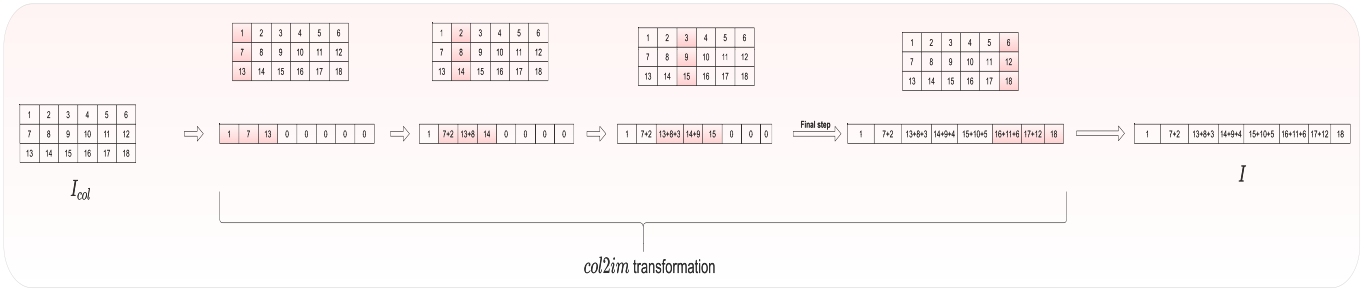

Fig. 3. Illustration of *col2im* transformation.

Using Eqs. (22) and (18), the individual terms of Eq. (39) can be written in the following way

$$D^{\alpha}_{K^{[i]}} S^{[i]} = I^{[i-1]}_{col} \left(\frac{(K^{[i]})^{1-\alpha}}{\Gamma(2-\alpha)} \right) \tag{44}$$

$$\frac{\lambda}{2} D^{\alpha}_{K^{[i]}} K^{[i]2} = \lambda \frac{(K^{[i]})^{2-\alpha}}{\Gamma(3-\alpha)} \tag{45}$$

Substituting the above results in Eq. (39) the fractional kernel gradient is given as

$$D^{\alpha}_{K^{[i]}} L = \begin{cases} \dfrac{\partial L}{\partial C^{[i]}} I^{[i-1]}_{col} \left(\dfrac{(K^{[i]})^{1-\alpha}}{\Gamma(2-\alpha)} \right) + \lambda \dfrac{(K^{[i]})^{2-\alpha}}{\Gamma(3-\alpha)}, & \text{if } S^{[i]} > 0 \\ \lambda \dfrac{(K^{[i]})^{2-\alpha}}{\Gamma(3-\alpha)}, & \text{if } S^{[i]} \leq 0 \end{cases} \tag{46}$$

Similarly, using Eqs. (41) and (43) the fractional gradient with respect to the bias is given as

$$D^{\alpha}_{b^{[i]}} L = \begin{cases} \dfrac{\partial L}{\partial C^{[i]}} \left(\dfrac{(b^{[i]})^{1-\alpha}}{\Gamma(2-\alpha)} \right), & \text{if } S^{[i]} > 0 \\ 0, & \text{if } S^{[i]} \leq 0 \end{cases} \tag{47}$$

Next, for back-propagating the errors to the previous layers such as average pooling or up-sampling layer, the input gradient $\frac{\partial L}{\partial I^{[i-1]}}$ needs to be calculated. For this, first we need to calculate the gradient $\frac{\partial L}{\partial I^{[i-1]}_{col}}$. It's value can be obtained using Eq. (22) and is given as

$$\frac{\partial L}{\partial I^{[i-1]}_{col}} = \frac{\partial L}{\partial S^{[i]}} (K^{[i]}) \tag{48}$$

with the dimension being $N^{[i-1]}_C \times F^{[i]} \times N^{[i]}_W$. Now, the value of $\frac{\partial L}{\partial I^{[i-1]}}$ is obtained as follows

$$\frac{\partial L}{\partial I^{[i-1]}} = col2im \left(\frac{\partial L}{\partial I^{[i-1]}_{col}} \right) \tag{49}$$

with the dimension being $N^{[i-1]}_C \times N^{[i-1]}_W$. Here, the inverse transformation $col2im$ (Fig. 3) is operated on $\frac{\partial L}{\partial I^{[i-1]}_{col}}$. Next, we use $\frac{\partial L}{\partial I^{[i-1]}}$ as $\frac{\partial L}{\partial U^{[i-1]}}$ or $\frac{\partial L}{\partial P^{[i-1]}}$ depending on whether the previous layer is an average pooling layer or the up-sampling layer. Once the gradients of all the trainable parameters are obtained, parameter update is carried out using gradient descent of learning rate η as follows

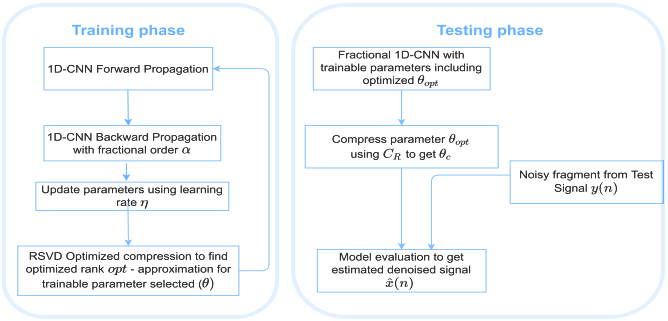

Fig. 4. Workflow of the proposed compressed fractional 1D-CNN.

$$K^{[i]} \leftarrow K^{[i]} - \eta D^{\alpha}_{K^{[i]}} L \tag{50}$$

$$W \leftarrow W - \eta D^{\alpha}_{W} L \tag{51}$$

$$b^{[i]} \leftarrow b^{[i]} - \eta D^{\alpha}_{b^{[i]}} L \tag{52}$$

$$B \leftarrow B - \eta D^{\alpha}_{B} L \tag{53}$$

We have mathematically proved the convergence of the proposed algorithm, the details of which can be found in the supplementary material.

IV. Compressed Architecture

The flowchart for the compressed version of the fractional based CNN architecture is shown in Fig. 4. The training phase consists of forward and backward propagation of the fractional architecture discussed in Sec. III. This is followed by updating the trainable weights using fractional gradient descent. Next, the compression of the trained weights is carried out using RSVD function discussed in *algorithm* 1. The inputs required for the calculation of RSVD are θ and r. Here, r denotes the rank of the matrix whereas $\theta \in \{K^{[i]}, W\}$ denotes the trainable set consisting of kernels $K^{[i]}$, which are used in convolution layers (CONV) and W is the inter-connection weight between the flattened and FC layers. The optimized rank (opt) is calculated as the rank at which 90% of variance is covered for the singular vectors obtained using SVD decomposition given in Eq. (15). This is carried out using $check\_optimized\_rank$. The above description about the compression procedure carried out during the training

process is summarized in *algorithm* 1 and the steps are repeated till the convergence is achieved.

Algorithm 1 RSVD Compression for Obtaining Optimized Kernels and Weight Matrices

procedure $RSVD\_Opt\_Compression(\theta, r)$:
calculate $[U_r, S_r, V_r]$ = RSVD (θ, r) ▷ from Eqs. (11)-(16)
$opt \leftarrow$ check_optimized_rank(S_r)
$\theta_{opt} \leftarrow U_{\text{opt}} S_{\text{opt}} V_{opt}^T$
return θ_{opt}
end procedure

During the testing phase shown in Fig. 4, the architecture has the optimized trainable parameter θ_{opt}, obtained using *algorithm* 1. Next, the compression of the θ_{opt} based on the compression rate (C_R) is performed using *algorithm* 2 resulting in θ_c which is a rank r approximation of the θ_{opt}. Finally, the compressed parameter θ_c is used for denoising the EEG signals. The process is summarized in *algorithm* 2. Here, the compression rate C_R is varied from 5% to 95% for our observations and this is done for various values of fractional order α ranging from 1 to 1.5.

Algorithm 2 Evaluating Denoising Performance on Testing Dataset Using Compressed Kernels and Weight Matrices

Input: Trained parameter $\theta_{opt} \in \{K^{[i]}, W\}$
Initialize compression rate C_{R}
$r = (1 - C_{\text{R}}) * rank(\theta_{opt})$
$[U_r, S_r, V_r]$ = RSVD (θ_{opt}, r) ▷ from Eqs.(11)-(16)
$\theta_c \leftarrow U_r S_r V_r^T$
Estimated denoised signal $\hat{x}(n)$ using forward propagation ▷ from Eqs. (22)-(27)
return $\hat{x}(n)$

V. Results and Discussion

In this section, several experiments are conducted to validate the efficiency of our proposed architecture. Firstly the standard datasets on which evaluations are conducted is presented, followed by experimental results including comparison with existing MA removal methods. Next, a detail study in which the performance of the architecture after compressing the kernel weights using low rank approximation is examined. All the experiments are performed on TESLA K80 GPU.

A. EEG Datasets

Now, we evaluate the architecture on two publicly available databases, i.e., Mendeley [27] and epileptic Bonn [28] database. Mendeley database contains clean EEG recordings of 40 subjects, each having 19 channels and sampled at 200 Hz. The epileptic Bonn database contains five different sets of databases, each representing a particular subject. The subjects Z and O contains EEG recordings of five healthy subjects with eyes open and closed, respectively, subjects N and F contain inter-ictal recordings from seizure patients and S represents the seizure-EEG signals. These EEG signals are sampled at 173.61 Hz. For creating MA-contaminated EEG signals, we take the help from examples of electromyograms database [29], which has clean EMG signals recorded from healthy subjects, and patients with myopathy and neuropathy. The noisy signals are created by randomly mixing the clean EEG signals with EMG signals after re-sampling all the signals at 200 Hz.

B. Data Preparation and Performance Metrics

For Mendeley database, we took 1026 signals, each of 2000 samples, from which the training and testing data were created after splitting the data into 80% training and 20% testing set. Bonn database has five subjects each having 100 signals. We take those 100 signals and make a 80%-20% train-test split across all the subjects, i.e., 80 signals are taken from each of the five subjects for training, while remaining 20 for testing. Accordingly, the contaminated signals are generated by randomly mixing EMG signals with the original EEG ones. For comparison with existing MA removal methods, we take the subject 'Z' from Bonn database (represented by Bonn(Z)) for comparison while the whole testing set is taken from Mendeley database.

The next step involves creating fragments of 250 samples from both the noisy and original EEG signals. The number of signals after taking each fragments is just 8 times more, so it is not enough for training any deep neural network. To combat this, we perform data augmentation by randomly choosing a point from a particular signal to take 250 fragments and repeat this process for few number of iterations such that finally, we have around 20000 fragments for training. All these fragments are then transformed into orthogonal domain using TMs. After normalizing the fragments using *standardscalar* function from *sklearn* [30] library, the resulting noisy Tchebichef vectors are fed as an input to the proposed architecture.

The evaluation of the proposed architecture is performed using different performance metrics such as signal-to-noise ratio (SNR), correlation coefficient (CC), percentage root mean square difference (PRD), root mean square error ($RMSE$) and mean absolute error (MAE) as mentioned in [9].

To validate the effectiveness of the proposed architecture, comparison of the results are carried out with the existing MA removal methods, namely, Wavelet based EEG denoising [4], EEMD [6], EMD-CCA [7], EEMD-CCA [7], EEMD-MCCA [8] and VMD [9]

C. Denoising Performance

To validate the denoising performance of the architecture, a comparative analysis is carried out with the existing MA removal methods that have given good results on both the databases. It can be observed from Table I, that the performance metrics for the proposed fractional CNN architecture outperforms all of the existing methods. Our model gives an improvement of 8.5% and 12.7% in SNR values for

TABLE I
COMPARISON OF PROPOSED ARCHITECTURE WITH EXISTING MA REMOVAL METHODS ON MENDELEY AND BONN(Z) DATABASE

Methods	Database	SNR	CC	PRD	RMSE	MAE
EEMD-CCA [7]	Mendeley	7.20	0.90	45.00	0.11	0.08
	Bonn (Z)	8.20	0.93	39.00	0.11	0.08
EEMD [6]	Mendeley	7.46	0.90	43.61	0.11	0.08
	Bonn (Z)	8.44	0.93	38.01	0.10	0.08
EMD-CCA [7]	Mendeley	4.46	0.81	59.56	0.15	0.11
	Bonn (Z)	5.45	0.85	53.81	0.15	0.10
Wavelet [4]	Mendeley	7.10	0.90	45.51	0.11	0.08
	Bonn (Z)	7.70	0.92	41.52	0.11	0.08
EEMD-MCCA [8]	Mendeley	7.26	0.90	44.78	0.11	0.08
	Bonn (Z)	8.26	0.93	38.94	0.11	0.08
VMD [9]	Mendeley	8.00	0.92	41.46	0.10	0.07
	Bonn (Z)	9.00	0.95	36.13	0.09	0.07
Proposed Method	Mendeley	**8.68**	**0.93**	**37.65**	**0.09**	**0.06**
	Bonn (Z)	**10.15**	**0.97**	**31.59**	**0.08**	**0.06**

Mendeley and Bonn(Z), respectively, when compared to the recently introduced variational mode decomposition (VMD) method.

The optimal hyper-parameters for the convolutional neural network were selected only after parameter tuning. It was found that with batch size of 64, learning rate (η) of 0.0005, regularization parameter (λ) of 0.00001 and training for 300 epochs was found to be optimal for Mendeley database. For the Bonn database, training is performed using 200 epochs while the other hyper-parameters remains the same.

Apart from the standard hyper-parameters listed in the aforementioned paragraph, the proposed architecture provides an extra hyper-parameter α that can be tuned to boost the denoising performance. The fractional order (α) is used in calculating the weight gradients in back-propagation discussed in Sec. III-C. Experiments are conducted, where α value is varied from 1 to 1.6 in steps of 0.1 and the optimal one is taken for final evaluation. The performance metrics are represented in Table II after calculating the average over all the test data. It can be seen that for both the Mendeley and Bonn(Z) database, $\alpha = 1.2$ gives the best denoising results. Compared to the traditional integer order CNN with $\alpha = 1$, fractional CNN with $\alpha = 1.2$ gives around 7% and 4% performance boost in denoising for Mendeley and Bonn database respectively.

The original, noisy and the denoised EEG signals in spatial domain are visualized on left side of the Figs. (5) and (6) for Mendeley and Bonn(Z) database, respectively. The corresponding signals in Tchebichef domain are plotted on the right side. It can be observed from Figs. 5-6(d)-(f) that by transforming signals into orthogonal space using TMs exhibits sparse behaviour. These sparse signals are used as input feature vectors that helps in accelerating the training process as the architecture now requires fewer number of input coefficients to work upon.

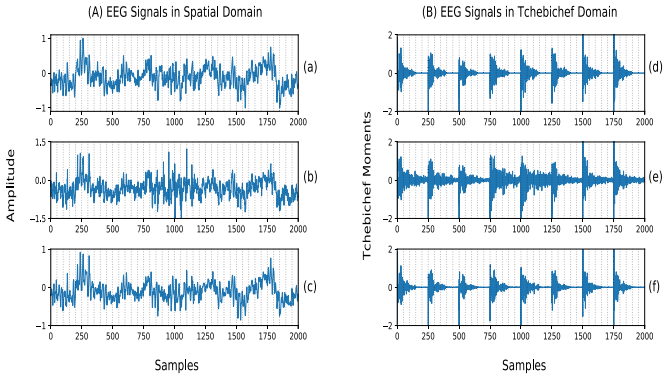

Fig. 5. EEG signals visualizations for Mendeley database. (A) Spatial domain: (a) Original signal (b) Noisy signal (c) Denoised signal; (B) Tchebichef moment: (d)-(f) Corresponding signals in Tchebichef domain.

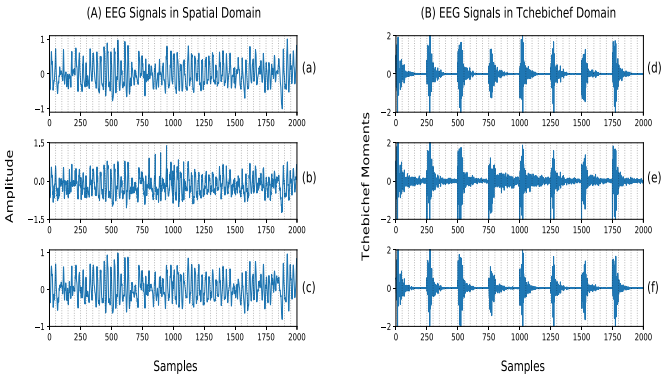

Fig. 6. EEG signals visualizations for Bonn(Z) database. (A) Spatial domain: (a) Original signal (b) Noisy signal (c) Denoised signal; (B) Tchebichef moment: (d)-(f) Corresponding signals in Tchebichef domain.

D. Compression Analysis

In this section, denoising results are presented when the fractional auto-encoder is trained with RSVD compression carried out on trainable weights (see Sect. IV). The optimized rank for the weights matrix in FC layer and kernel matrix in CONV2 and CONV3 layer is selected such that 90% of singular values can be retained. Table III shows the original and the optimized rank for CONV2, CONV3 and FC layers calculated using *algorithm* 1. It can be seen that for

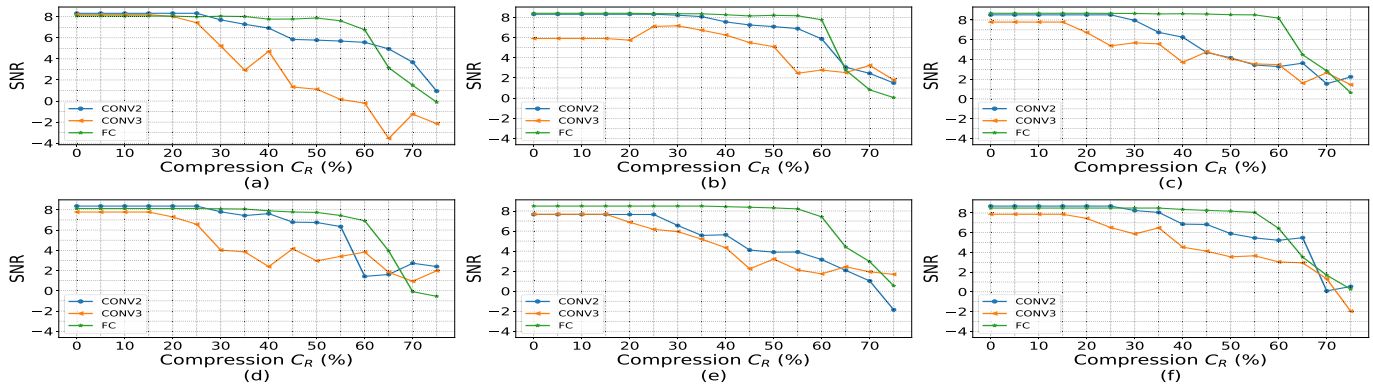

Fig. 7. SNR performance of the proposed architecture on Mendeley database after compression. Fractional order (α) used are: (a) 1, (b) 1.1, (c) 1.2, (d) 1.3, (e) 1.4, (f) 1.5.

TABLE II
PERFORMANCE OF PROPOSED MODEL ON MENDELEY AND BONN(Z) DATABASE

α	Database	SNR	CC	PRD	RMSE	MAE
1	Mendeley	8.07	0.92	40.37	0.10	0.07
	Bonn (Z)	9.87	0.96	32.28	0.08	0.06
1.1	Mendeley	8.41	0.93	38.80	0.10	0.07
	Bonn (Z)	9.85	0.96	32.46	0.08	0.06
1.2	Mendeley	**8.68**	0.93	**37.65**	0.09	0.06
	Bonn (Z)	**10.15**	0.97	**31.59**	0.08	0.06
1.3	Mendeley	8.32	0.92	39.14	0.10	0.07
	Bonn (Z)	9.16	0.95	35.25	0.09	0.07
1.4	Mendeley	8.49	0.93	38.49	0.09	0.06
	Bonn (Z)	10.09	0.97	31.67	0.08	0.06
1.5	Mendeley	8.50	0.92	38.48	0.10	0.07
	Bonn (Z)	8.85	0.94	36.62	0.10	0.08
1.6	Mendeley	7.50	0.92	43.36	0.11	0.08
	Bonn (Z)	8.33	0.93	38.67	0.11	0.08

TABLE III
LAYER WISE OPTIMIZED RANKS AFTER TRAINING THE PROPOSED ARCHITECTURE WITH RSVD COMPRESSION FOR MENDELEY AND BONN(Z) DATABASE

Database	CONV2		CONV3		FC	
	Original Rank	Optimized Rank	Original Rank	Optimized Rank	Original Rank	Optimized Rank
Mendeley	48	36	64	53	250	187
Bonn(Z)	48	36	64	53	250	211

Bonn(Z) database, the optimized rank for FC layer is higher as compared to that of Mendeley database, which signifies that this layer will be more sensitive to compression for Bonn(Z) database.

The CONV2, CONV3 and FC layers are individually compressed using the optimized ranks obtained in Table III and the performance of the model is evaluated using *algorithm* 2. Here, CONV1 layer is not compressed as it is directly interacting with the input. Fig. 7 shows the effect of layer-wise compression on denoising performance evaluated in terms of SNR using the proposed model at various fractional orders.

It can be observed that conventional CNN auto-encoder network shown in Fig. 7(a) with $\alpha = 1$ outperforms the existing state-of-the-art methods if CONV2 layer is compressed up-to 25%. However, as shown in Fig. 7(c), the proposed architecture at $\alpha = 1.2$ gives higher SNR compared to the conventional CNN even when the CONV2 layer is compressed by 30%. Likewise for FC layer compression, the architecture at $\alpha = 1.2$ gives better SNR values compared to other state of the art methods even at 60% compression, while the conventional CNN outperforms other methods only upto 35% compression. Moreover, the proposed architecture at $\alpha = 1.5$ (Fig. 7(f)) gives good performance under compression upto 55% and 35% of FC and CONV2, respectively. Similar analysis is for the Bonn(Z) database and the results are shown in Fig. 8. It can be observed that the best SNR is obtained compared to the existing methods at $\alpha = 1.2$. This performance is guaranteed even when the CONV2, CONV3 and FC layers are compressed by 25%. For this database, it can be seen that FC layer is more sensitive to compression. This is because the optimized rank (see Table III) is 15% of the original rank, as a result the compression at early stages leads to drop in

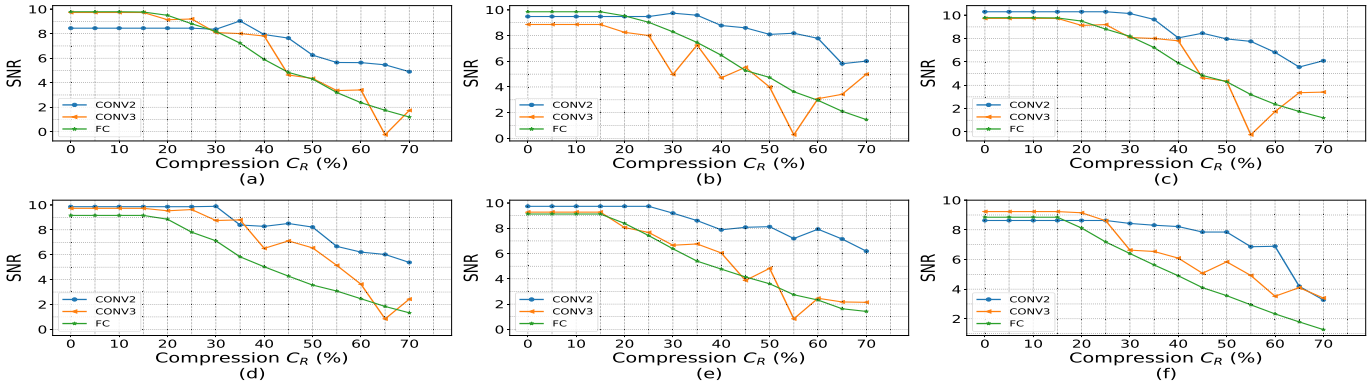

Fig. 8. SNR performance of the proposed architecture on Bonn(Z) database after compression. Fractional order (α) used are: (a) 1, (b) 1.1, (c) 1.2, (d) 1.3, (e) 1.4, (f) 1.5.

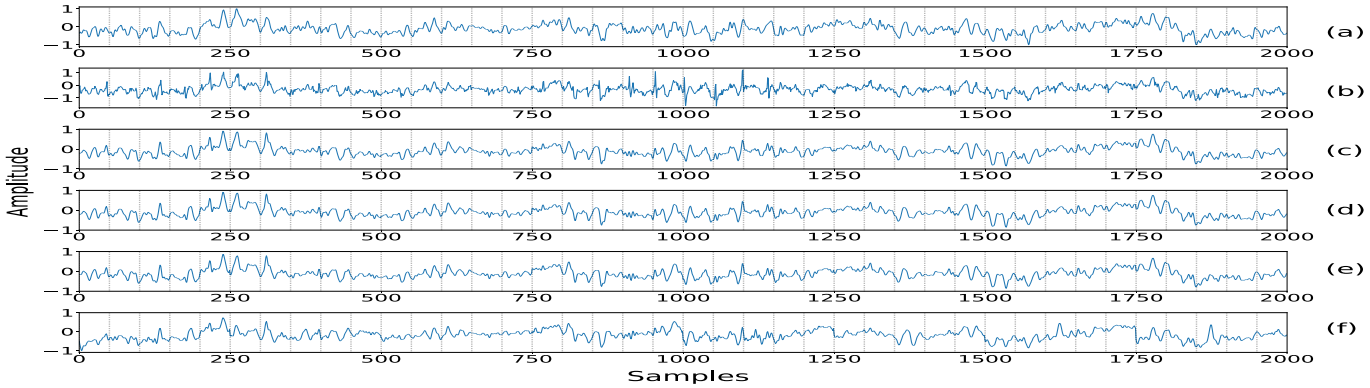

Fig. 9. EEG signals visualization under compression at various rates: (a) Original EEG signal, (b) Noisy EEG signal; Denoised signals: (c) without compression, (d) 25% compression, (e) 50% compression, (f) 75% compression.

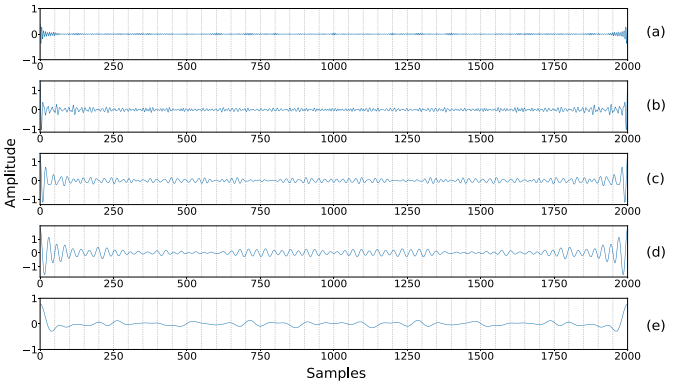

Fig. 10. Five frequency bands of original EEG signal from Bonn database, (a) Gamma band, (b) Beta band, (c) Alpha band, (d) Theta band, (e) Delta band.

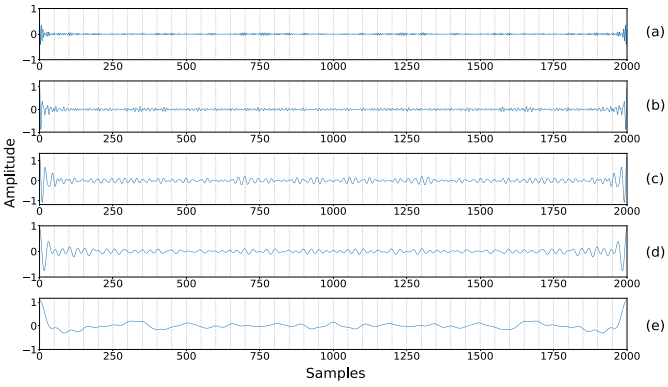

Fig. 11. Five frequency bands of original EEG signal from Mendeley database, (a) Gamma band, (b) Beta band, (c) Alpha band, (d) Theta band, (e) Delta band.

the SNR performance. This is not in the case of Mendeley database as optimized rank is 25% of the original rank. For further qualitative analysis, Fig. 9 shows the denoised signals produced by the architecture at $\alpha = 1.2$ when CONV2 layer is subjected to various compression rates. It can be seen that even at 50% compression the reconstructed signal resembles the original one at most of the points. Lastly, two things can be concluded from this study. Firstly, using hyper-parameter α can boost the SNR values of the EEG signals significantly. Secondly, by compressing the weights of the architecture does not degrade the SNR performance too much and its still better than the existing methods. Compressing the architecture provides an advantage that it consumes less memory space and is suitable for edge computing devices.

E. EEG Quality in Frequency Domain

We classify the EEG waveform into five frequency bands,i.e, delta (0.5-4 Hz), theta (4-8 Hz), alpha (8-14 Hz),

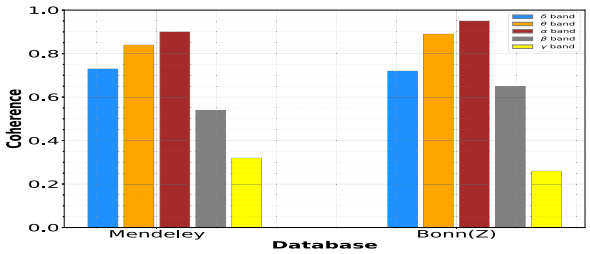

Fig. 12. Coherence with respect to five frequency bands for both Mendeley and Bonn(Z) database.

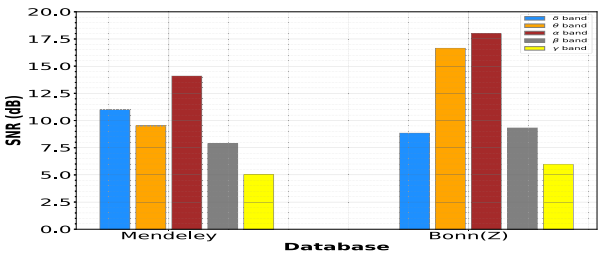

Fig. 13. SNR with respect to five frequency bands for both Mendeley and Bonn(Z) database.

beta (14-30 Hz) and gamma (over 30 Hz) [31]. Fig. 10 and 11 displays the five frequency bands for original EEG signal from the Bonn and Mendeley database respectively. The same is also done for the respective noisy and denoised EEG signals. Coherence and SNR is then calculated by taking the original and denoised EEG signals for the respective frequency bands. Fig. 12 depicts the coherence between the original and the denoised EEG signal. We get the best results for alpha frequency band, while it decreases for higher frequency bands like beta and gamma. The same trend can be obtained in Fig. 13, where SNR is calculated for original and denoised signals at respective frequency band for both the databases.

VI. Conclusion

A fractional and compressed one-dimensional CNN auto-encoder, which uses orthogonal features in the form of Tchebichef moments has been proposed. The proposed method gives superior results in denoising MA contaminated signals when compared to existing MA removal methods with the best result observed at $\alpha = 1.2$. Moreover, increasing the compression ratio (C_R) for the weights of the architecture, it beats the existing methods when evaluated using the performance metrics. Another important observation is that a compressed fractional architecture at $\alpha = 1.2$ performs better than the conventional CNN auto-encoder without compression, i.e., with 60% compression of FC layer for Mendeley database and 30% compression of the CONV2 layer for the Bonn(Z) database. Compressing the architecture not only makes its occupies less memory foot-print but also delivers superior performance compared to other methods. Qualitative analysis of the signals is presented at various stages of compression, which showed that the denoised signals are close to the original signals. Future work involves deployment of the compressed architecture on portable low energy devices.

Acknowledgment

The authors would like to thank the support received from Dr. Bakul Gohel at DA-IICT in providing relevant information on frequency domain based metrics used in EEG denoising application.

References

[1] F. C. Morabito *et al.*, "Enhanced compressibility of EEG signal in Alzheimer's disease patients," *IEEE Sensors J.*, vol. 13, no. 9, pp. 3255–3262, Sep. 2013.

[2] X. Jiang, G.-B. Bian, and Z. Tian, "Removal of artifacts from EEG signals: A review," *Sensors*, vol. 19, no. 5, p. 987, Feb. 2019.

[3] X. Chen *et al.*, "Removal of muscle artifacts from the EEG: A review and recommendations," *IEEE Sensors J.*, vol. 19, no. 14, pp. 5353–5368, Jul. 2019.

[4] N. Mammone, F. La Foresta, and F. C. Morabito, "Automatic artifact rejection from multichannel scalp EEG by wavelet ICA," *IEEE Sensors J.*, vol. 12, no. 3, pp. 533–542, Mar. 2012.

[5] N. E. Huang *et al.*, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time," *Proc. Roy. Soc. London A, Math., Phys. Eng. Sci.*, vol. 454, pp. 903–995, Mar. 1998.

[6] Z. Wu and N. E. Huang, "Ensemble empirical mode decomposition: A noise-assisted data analysis method," *Adv. Adapt. Data Anal.*, vol. 1, no. 1, pp. 1–41, 2008.

[7] K. T. Sweeney, S. F. McLoone, and T. E. Ward, "The use of ensemble empirical mode decomposition with canonical correlation analysis as a novel artifact removal technique," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 1, pp. 97–105, Jan. 2013.

[8] X. Chen, C. He, and H. Peng, "Removal of muscle artifacts from single-channel EEG based on ensemble empirical mode decomposition and multiset canonical correlation analysis," *J. Appl. Math.*, vol. 2014, pp. 1–10, Jun. 2014.

[9] M. Saini, U. Satija, and M. D. Upadhayay, "Effective automated method for detection and suppression of muscle artefacts from single-channel EEG signal," *IET Healthcare Technol. Lett.*, vol. 7, no. 2, pp. 35–40, Apr. 2020.

[10] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[11] Y. Li, L. Guo, Y. Liu, J. Liu, and F. Meng, "A temporal-spectral-based squeeze-and-excitation feature fusion network for motor imagery EEG decoding," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1534–1545, 2021.

[12] E. Eldele *et al.*, "An attention-based deep learning approach for sleep stage classification with single-channel EEG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 809–818, 2021.

[13] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory DSP]," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 145–154, Jan. 2011.

[14] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28. Red Hook, NY, USA: Curran Associates, 2015, pp. 1–9.

[15] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proc. Brit. Mach. Vis. Conf. (BMVA)*, 2014, pp. 1–12.

[16] N. B. Erichson, S. Voronin, S. L. Brunton, and J. N. Kutz, "Randomized matrix decompositions using R," *J. Stat. Softw.*, vol. 89, no. 11, pp. 1–47, 2019.

[17] E. Clark, S. L. Brunton, and J. Nathan Kutz, "Multi-fidelity sensor selection: Greedy algorithms to place cheap and expensive sensors with cost constraints," *IEEE Sensors J.*, vol. 21, no. 1, pp. 600–611, Jan. 2021.

[18] X. Zou, X. Xu, C. Qing, and X. Xing, "High speed deep networks based on discrete cosine transformation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 5921–5925.

[19] J. Zhao, R. Xiong, J. Xu, F. Wu, and T. Huang, "Learning a deep convolutional network for subband image denoising," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 1420–1425.

[20] H. Wu and S. Yan, "Computing invariants of Tchebichef moments for shape based image retrieval," *Neurocomputing*, vol. 215, pp. 110–117, Nov. 2016.

[21] A. Kumar, M. O. Ahmad, and M. N. S. Swamy, "Tchebichef and adaptive steerable-based total variation model for image denoising," *IEEE Trans. Image Process.*, vol. 28, no. 6, pp. 2921–2935, Jun. 2019.

[22] J. Yu, L. Tan, S. Zhou, L. Wang, and M. A. Siddique, "Image denoising algorithm based on entropy and adaptive fractional order calculus operator," *IEEE Access*, vol. 5, pp. 12275–12285, 2017.

[23] Y.-F. Pu, J.-L. Zhou, and X. Yuan, "Fractional differential mask: A fractional differential-based approach for multiscale texture enhancement," *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 491–511, Feb. 2010.

[24] J. Wang, Y. Wen, Y. Gou, Z. Ye, and H. Chan, "Fractional-order gradient descent learning of BP neural networks with Caputo derivative," *Neural Netw.*, vol. 89, pp. 19–30, May 2017.

[25] R. Mukundan, "Some computational aspects of discrete orthonormal moments," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1055–1059, Aug. 2004.

[26] C. Bao, Y. Pu, and Y. Zhang, "Fractional-order deep backpropagation neural network," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–10, Jul. 2018.

[27] M. A. Klados and P. D. Bamidis, "A semi-simulated EEG/EOG dataset for the comparison of EOG artifact rejection techniques," *Data Brief*, vol. 8, pp. 1004–1006, Sep. 2016.

[28] 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," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 64, no. 6, pp. 1–8, Nov. 2001.

[29] A. Goldberger *et al.*, "PhysioBank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[30] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[31] M. Abo-Zahhad, S. M. Ahmed, and S. N. Abbas, "A new EEG acquisition protocol for biometric identification using eye blinking signals," *Int. J. Intell. Syst. Appl.*, vol. 7, no. 6, pp. 48–54, May 2015.