Sig-R^{2}ResNet: Residual Network with Signal Processing-refined Residual Mapping, Auto-tuned L_1-Regularization with Modified Adam Optimizer for Time Series Classification

Arijit Ukil, Soma Bandyopadhyay, Arpan Pal
Embedded Systems and Robotics, Research and Innovation
Tata Consultancy Services,
Kolkata, India
{arijit.ukil, soma.bandyopadhyay, arpan.pal}@tcs.com

***Abstract*—Time Series Classification (TSC) is becoming a challenging and important problem to solve specifically due to the advent of sensor-based applications and Internet of Things (IoT). Residual mapping displays easier and evidently near-optimal learning. In this paper, we extend this notion by incorporating fine-grained refining of the residual learning through augmentation of feature space using gamut of signal processing transformations. Our proposed Sig-R^{2}ResNet refines the learning process by introducing newer representation through signal processing primitives without distorting the residual mapping channel, along with an auto-tuning L_1 regularization. We further adapt the learning rate decay through learning over the trend of validation loss and modify the network parameter update process of Adam optimizer. The proposed method- Sig-R^{2}ResNet can be viewed as an informal game where training experience is augmented through unsupervised signal processing features while the model growth is controlled by a regularization process and smoother learning convergence is achieved by validation loss dependent learning rate. One of the novelties is that the training signal dynamics control the enhancement of representation complexity, regularization and learning rate adaptation. We perform extensive experiments with diverse datasets from publicly available UCR time series database and demonstrate empirical evidences that our method consistently outperforms the existing benchmark results (creating 59.10% new benchmark results) as well as shows significantly better classification outcome than the current baselines and state-of-the-art algorithms like ResNet, BOSS, COTE.**

Keywords— Sensor signal, feature augmentation, ResNet, analytics, signal processing, regularization, learning rate

I. Introduction and our Contribution

Time series signals play a major role for developing array of applications in Internet of Things (IoT), ranging from healthcare, transportation, retail and many other domains [27, 29]. It is agreed upon among the researchers and practitioners that Time Series Classification or TSC is a major challenge in IoT applications. Deep residual learning has shown tremendous success in image and video analytics (visual applications). Deep residual learning was introduced to solve the learning degradation problem when the depth of the deep network is increased [1- 2]. The problem is to provide the layer-wise recursive learning of $\mathcal{H}_{l+1}(\mathcal{X}) = \mathcal{H}_l(\mathcal{X}) + \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right)$, where $\mathcal{G}_l$ is the non-linear neural network (say, convolution network), $\mathcal{H}_l(\mathcal{X})$ is the residual function and $\mathcal{H}_0(\mathcal{X}) = 0, \mathcal{G}_0\left(\mathcal{H}_0(\mathcal{X})\right) = \mathcal{X}$, $\mathcal{X}$ is the input time series. It is described in [3], individual layers in residual networks attempt to modify the learnt representation from the previous layers. It does not learn new representation. However, learning by purely modifying the input $\mathcal{X}$ throughout the learning process may be incomplete. The model has the higher chance to suffer from the internal co-variance shift problem [4] when l^{th} layer irrelevant feature $\mathcal{H}_l(\mathcal{X})$ is encountered in the residual function $\mathcal{H}_{l+1}(\mathcal{X}) = \mathcal{H}_l(\mathcal{X}) + \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right)$. Such potential inferior model learning is due to the strong coupling of $\mathcal{H}_l(\mathcal{X})$ with $\mathcal{X}$.

Inspired by the observations that mere modification of representation space is insufficient as put forward in [3], we feel that refining the input representation along with enrichment of the learning through augmented representation space would provide the necessary impetus to the residual learning process. Our motivation stems from the fact that most of the works (exception [5] or may be few others) of ResNet-based deep learning research (including the seminal work on ResNet [1]) focuses on solving visual application problems. In order to address the TSC problem, we find that baseline ResNet model for time series signals is not sufficient for ensuring consistently good performance outcome [5]. In this paper, we propose that unsupervised feature space integrated residual mapping enriches the learning process by introducing newer representation. The presence of intricate and diverse patterns in practical time series signals demand rich feature space exploration that covers not only temporal, but also spectral as well as time-frequency characterization. Additionally, we need to ensure that network growth is made restricted as a function of the time series signal dynamics. It is shown in [6] that regularization is of utmost importance to construct a better-learned network. We apply L_1 regularization [20] for regulating the model complexity introduced by feature space augmentation and learning rate adaptation is introduced for smoother gradient descent. The prime motivations of choosing L_1 regularization is the possibility of sparser and interpretable model [20]. Moreover, we propose Mod-Adam, a modified adaptive moment estimation (Adam optimizer [24]) for updating network weights. In Mod-Adam, through the inspection of the validation loss, the learning rate value is

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

directly decayed to enable smoother learning convergence. Mod-Adam indirectly helps to minimize the generalization loss.

In Sig-R^{2}ResNet, the representation space augmentation tries to minimize the training error through introducing model complexity, dynamic regularization and learning rate adaptation attempt to minimize the validation error by regulating the model growth and enabling smoother learning convergence respectively. The obtained empirical evidences on publicly available UCR (University of California, Riverside) time series datasets [7, 8] prominently demonstrate the significant performance gain of Sig-R^{2}ResNet against state-of-the-art algorithms.

Our proposed method has the following distinct parts:

1. *Enriching the residual learning with new representation through unsupervised feature space augmentation that refines the residual channel of Residual Network to enhance the model complexity as well as improving the training loss:* The unsupervised signal processing feature set $\Omega = [\omega_1, \omega_2, \omega_3, .., \omega_\beta]$ transforms the input time series signals $\mathbb{R}^T \rightarrow \Omega \in \mathbb{R}^\beta$ to β-dimension feature vectors and incorporating the external feature vectors Ω refine the residual map. The process of concatenation of additional signal processing features eases the learnability in high-dimensional space.

2. *Regularizing the deep network exploiting the time series signal dynamics to attempt the minimization of generalization loss:* Stochasticity of practical time series signals (examples: accelerometer, engine noise, household energy consumption signal from smart energy meter, etc.) induces us to develop signal dynamics explored L_1 regularization factor setting. We propose a L1 regularization method which automatically computes the L_1 regularization factor as a function of the training set distribution. An appropriate setting of L_1 regularization relaxes the model to underplay the higher learnability with the consequence of complex representation space as an outcome in the process of enrichment of residual learning by signal processing feature space augmentation.

3. *Continuous adaptation of learning rate for smoother convergence with generalization loss:* The update of parameters (ω) at the ξ^{th} epoch in a typical deep neural network follow the following equation:

$$\omega_\xi \leftarrow \omega_{\xi-1} - \alpha \nabla_{\omega_{\xi-1}} f(\omega_{\xi-1})$$
$$\omega_\xi \leftarrow \omega_{\xi-1} - \alpha g(\omega_{\xi-1})$$
$$g(\omega_{\xi-1}) = \nabla_{\omega_{\xi-1}} f(\omega_{\xi-1})$$

Where, $f(\omega_{\xi-1})$ is a stochastic objective function with parameters $\omega_{\xi-1}$, α is the learning rate and $\nabla_{\omega_{\xi-1}} f(\omega_{\xi-1})$ denotes the gradient w.r.t $f(\omega_{\xi-1})$. The well-known Adam optimization that provides combined advantages of established stochastic optimization methods AdaGrad [25] and RMSProp [26]. Adam method updates of the network parameters by changing the $g(\omega_{\xi-1})$, $g(\omega_{\xi-1}) \xrightarrow{Adam} g_{Adam}(\omega_{\xi-1})$ which in effect, impacts the values of ω_ξ [24]. However, the learning rate parameter α is made fixed (typically, $\alpha = 0.001$ [24]). We propose Mod-Adam, that dynamically modifies α based on the trend of validation loss over the epochs. Thus, the parameter updation of Mod-Adam is: $\omega_\xi \leftarrow \omega_{\xi-1} - \alpha_\xi^{Mod-Adam} g_{Adam}(\omega_{\xi-1})$. Mod-Adam considers the validation loss while calculating ω_ξ for providing better and intelligent path to convergence by indirectly leading to a path of generalization loss minimization.

II. PRIOR WORKS AND BACKGROUND

Dynamic Time Wrapper based distance function with Nearest Neighbor classifier (1NN- DTW) is the baseline approach to solve TSC problems [9- 10]. Gradually, better learning methods were developed. For example, Collective of Transformation-based Ensembles (COTE) [11] algorithm maps the time series to shapelets along with 35 classification ensembles. Another algorithm BOSS is a truncated Discrete Fourier Transform-based substructure analysis [12].

Understandably, researchers have shown interests to study the deep architectures. ResNet is introduced in [5] to address the TSC problem as the baseline deep neural network. ResNet-inspired architecture has historically demonstrated excellent performance (ResNetXt [14]) in various classification problems (mainly in visual applications). The crux of success is the addition of identity mapping of the input signal such that the non-linear mapping function (say, Rectified Liner Unit - ReLU) acts on the residual part. However, time series signals are more stochastic than the image or visual signals. With the augmentation of the feature space that captures the intrinsic patterns (in different transformed domain like wavelet, spectral), Sig-R^{2}ResNet attempts to enhance the capability of residual learning through a refined representation channel. Dynamic regularization controls the model growth and adaptive learning rate improves learnability. Thus, a competitive model learning is evolved. In fact, we have shown in our previous work that signal processing features, termed as Signal Properties based Generic Features (SPGF) have a promising role for TSC problems [13]. SPGF features have further demonstrated that fusing with autoencoder based feature extraction (TimeNet [17]) results in considerable performance gain. This work indeed is a motivation for us to develop Sig-R^{2}ResNet.

However, we feel that the major issue in the learning algorithms for TSC is the absence of a single strong method over diverse set of datasets. For example, in the standard UCR time series database [7- 8], the existing benchmark results on 44 diverse datasets are contributed by more than 15 different algorithms and none of the algorithms have reported consistently dependable performance with respect to the benchmark results. In this paper, we show that Sig-R^{2}ResNet is a strong contender of being a strong learning algorithm for TSC problems through empirical evidences over assorted UCR time series datasets.

III. SIG-R^{2}RESNET: REFINING THE RESIDUAL MAPPING AND REGULARIZING THE NETWORK CAPACITY

In order to describe Sig-R^{2}ResNet architecture and related algorithms, we illustrate few relevant definitions.

Definition 1. Time series signal $\mathcal{X} = [x_1, x_2, x_3, \dots, x_T] \in \mathbb{R}^T$ is an ordered set of real values.

Definition 2. Training dataset $\mathcal{T}_{Train} = [\mathcal{X}_{Train}, \mathcal{L}_{Train}]$, where training dataset $\mathcal{T}_{Train}$ consists of M number of training

instances, $\mathcal{X}_{Train} = [\mathcal{X}^1_{Train}, \mathcal{X}^2_{Train}, \dots, \mathcal{X}^M_{Train}]$, and each of the training instance consists of T number of samples, $\mathcal{X}^m_{Train} \in \mathbb{R}^T, \forall m, m = [1,2, \dots, M]$, $\mathcal{L}_{Train}$ denotes the lables, $\mathcal{L}_{Train} = [\mathcal{L}^1_{Train}, \mathcal{L}^2_{Train}, \mathcal{L}^3_{Train}, \dots, \mathcal{L}^M_{Train}]$, $\mathcal{L}^M_{Train} \in [1, \mathcal{C}]$, $\forall m$, labels correspond to one of the $\mathcal{C}$ classes, $\mathcal{L}^m_{Train} \in \mathbb{Z}$.

Definition 3. Unsupervised feature generation $\Omega = [\omega_1, \omega_2, \omega_3, .., \omega_\beta]$ transforms each of the training instances $\mathcal{X}^m_{Train} \in \mathbb{R}^\beta$ and converts to β dimension vectors: $\mathbb{R}^T \rightarrow \mathbb{R}^\beta$.

Definition 4. Deep Residual Network accepts: $M \times \beta$ unsupervised feature vectors, $M \times$ T training samples, and $M\times$ 1 training labels. Subsequently, Sig-R^{2}ResNet generates training model by accepting these three inputs.

Sig-R^{2}ResNet consists of two main components: 1. Refining the residual mapping by introducing the unsupervised features to boost training performance for high training accuracy and auto-tuned regularizations to tune the network parameters for improving the generalization accuracy.

A. Unsupervised Features using Signal Processing Transformations

We depict few examples of practical time series signals in Figure 1.

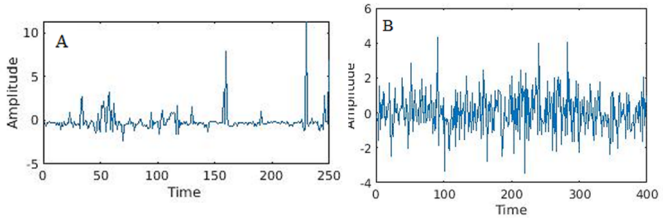

Fig. 1. Examples of time series signals: (A)- Computers, (B)-FordB from UCR time series database [7-8].

It is known and observed that time series signals are usually noisy and non-stationary with high degree of stochasticity. For the purpose of analyzing time series, particularly sensor time series signals, the classification tasks need to essentially understand the intricate details of the time series from different domains like statistical, spectral and wavelet. Hence, Sig-R^{2}ResNet learns through residual networks with refinement using signal processing transformations.

1) Signal processing transformations for low-level representation: In order to extract faithful and near-complete depiction of feature space, we propose a set of $\Omega = [\omega_1, \omega_2, \omega_3, .., \omega_\beta]$ features to augment the representation space through exploiting the temporal, statistical, wavelet and spectral properties of $\mathcal{X}$ [15-16, 18]. Such feature space Ω captures the long-term and short-term descriptions of $\mathcal{X}$. The complete feature space in Ω consist of two types: long-term features and short-term features. The output of each of the transformations including long-term features and short-term features. Each of the feature is a scalar and the complete feature space is a β dimension vector. In other words, there are total β number of transformations.

- Long-term features are constructed by 1. directly applying statistical functions on $\mathcal{X} = [x_1, x_2, x_3, \dots, x_T], x_T \in \mathbb{R}^T$ to get the temporal features: $\mathbb{R}^T \xrightarrow{statistical} \mathbb{R}$, (temporal features consists of different statstistical functions like mean, median, kurtosis and others along with information theoretic measures like Shannon entropy), and 2. first transforming $\mathcal{X}$ to spectral or wavelet domains, $\mathbb{R}^T \xrightarrow{spectral\ or\ wavelet} \mathbb{R}^T$ and subsequently statistical features are applied $\mathbb{R}^T \xrightarrow{statistical} \mathbb{R}$. Few examples of the long-term features are: $kurtosis(\mathcal{X})$, $max\big(Fast\ Fourier\ Transform\ co-efficient(\mathcal{X})\big)$, $median\big(Daubechies\ wavelet\ \ detailed\ co-efficient(\mathcal{X})\big)$ (For Daubechies wavelet transformation, please refer [23]).
- For short-term description, we segment $\mathcal{X} = \mathcal{X}_{segmented} = (\overbrace{x^1, x^2, \dots, x^s}^{segment\ \#\ 1}, \overbrace{x^{s+1}, x^{s+2}, \dots, x^{2s}}^{segment\ \#\ 2}, \dots)$ into h number of segments, whereas each of the segments consists of s number of samples: $\mathbb{R}^T \xrightarrow{segmentation} \mathbb{R}^{h\times s}$. Each of the segments (consisting of s number of samples) is again transformed and we get from first-level transformation: $\mathbb{R}^{h\times s} \xrightarrow{first-level\ transformation} \mathbb{R}^h$. Each of the resultant transformation outputs are further transformed using different descriptive statistics like measures of central tendencies (mean, median, etc…) and measures of dispersion (like standard deviation) to obtain aggregate information: $\mathbb{R}^h \xrightarrow{descriptive\ statistics} \mathbb{R}$. Few examples of the short-term features are: $median\left(Kurtosis(\mathcal{X}_{segmented})\right)$: Kurtosis of each of the segments is calculated and mean operation is performed over it, $\left(standard\_deviation\left(median\left(Detailed\ \ Daubechies\ wavelet\ \ co-efficient(\mathcal{X}_{segmented})\right)\right)\right)$: First, the median of the detailed co-efficients of Daubechies wavelet transformation from each of the segments are obtained and successively, standard deviation is computed. Moreover, we intend to emphasize that in presence of noise in $\mathcal{X}$ (which is obvious in time series signals) as well as in l^{th} residual layer $\mathcal{H}_l(\mathcal{X})$ short-term features have the ability to minimize the signal distortion. For example, total number of segments in $\mathcal{X}$ be 20 and three of which are got distorted due the presence of noise. Features like $\left(median\left(Kurtosis(\mathcal{X}_{segmented})\right)\right)$ has minimum impact due to such distortion.

B. Refining the residual map through augmenting feature space using signal processing transformations

The empirical evidences [1, 5] supported by some theoretical understanding of the advantage of the residual networks [2, 3, 21] lead us to safely assume that residual mapping based learning or ResNet-like learning network is a worthy candidate to attempt for solving classification tasks. Although little evidences are emerged other than the initial works by Wang. et al. [5], we have felt that ResNet is a suitable time series classification architecture owing to its inherent capability of

resisting the learning degradation problems in deep networks [1, 21] and norm-preserving gradient in the backward path [21]. In ResNet, the layers in the deep networks are reformulated to learn the residual function by adding an identity loop. Below, we depict in Figure 2, the typical architecture and construction of ResNet, where each of the residual blocks consist of number of layers with convolution blocks (conv) with Batch Normalization (BN) and Rectified Linear Unit (ReLU). From a closer look at the residual blocks and overall architecture of ResNet (from Figure 3), a residual learning block characteristically acts as a modification of the input representation. The residual mapping through identity connection has the potential to ease out the learning problems owing to the spurious local optima encountered in many learning networks [22].

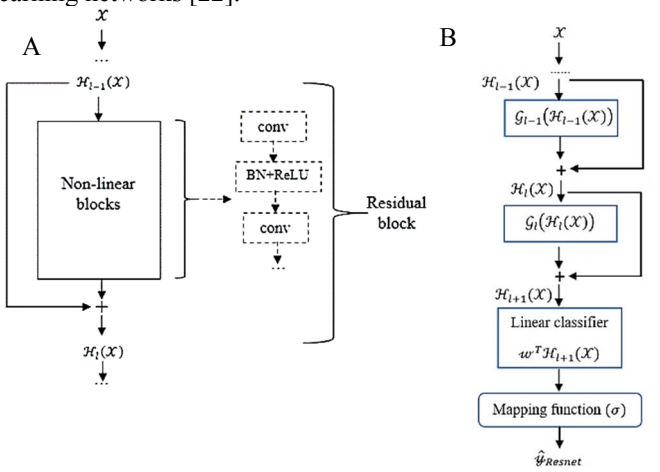

Fig. 2. ResNet functional architecture, (A). Representative residual block with convolution blocks with batch normalization (BN) and Rectified Liner Unit (ReLU). (B). Generic construction of Residual Networks (ResNet).

Let, $\mathcal{G}_l$ be the non-linear neural network (here, convolution network) module of the l^{th} level. The l^{th} residual block output is $\mathcal{H}_{l+1}(\mathcal{X})$.

$$\mathcal{H}_{l+1}(\mathcal{X}) = \mathcal{H}_l(\mathcal{X}) + \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right) \quad (1)$$

Recursive nature of Eqn. (1) leads to:

$$\mathcal{H}_{l+1}(\mathcal{X}) = \sum_{l=0}^{L} \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right) \quad (2)$$
$$\mathcal{H}_0(\mathcal{X}) = 0, \mathcal{G}_0\left(\mathcal{H}_0(\mathcal{X})\right) = \mathcal{X}$$

Over the representation of $\mathcal{H}_{l+1}(\mathcal{X})$, the given input $\mathcal{X}$ is ultimately fed to a linear classifier with $w \in \mathbb{R}^n$. Let, $\hat{y}$ be the predicted outcome:

$$\hat{y}_{Resnet} = \sigma\left(w^T \mathcal{H}_{l+1}(\mathcal{X})\right)$$
$$= \sigma\left(w^T \sum_{l=0}^{L} \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right)\right)$$

$\sigma(.)$ denotes the mapping function from classifier outputs to the predicted labels ($\hat{y}$). Typically, $\sigma(.)$ Is a linear classifier like softmax.

In Eqn. (1), we note that $\mathcal{H}_{l+1}(\mathcal{X})$ is an additive outcome unlike the conventional deep neural network where transfer function is multiplicative: $\prod w_i\, func(\mathcal{X}_i)$. From Eqn. (1):

$$\mathcal{H}_{l+1}(\mathcal{X}) = \mathcal{H}_l(\mathcal{X}) + \sum_{l=0}^{L} \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right)$$

From the backpropagation chain rule for the cost function $\mathcal{E}$:

$$\left.\frac{\partial \mathcal{E}}{\partial \mathcal{H}_l}\right|_{ResNet} = \frac{\partial \mathcal{E}}{\partial \mathcal{H}_{l+1}} \cdot \frac{\partial \mathcal{H}_{l+1}}{\partial \mathcal{H}_l}$$
$$= \frac{\partial \mathcal{E}}{\partial \mathcal{H}_{l+1}}\left(1 + \frac{\partial\left(\sum_{l=1}^{L-1} \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right)\right)}{\partial \mathcal{H}_l}\right) \quad (3)$$

Let us consider that Sig-R^{2}ResNet consists of L number of layers. In Sig-R^{2}ResNet, augmented feature vectors $\Omega(\mathcal{X})$ are concatenated ($\|$) with $\mathcal{H}_L(\mathcal{X})$ as shown in the Sig-R^{2}ResNet functional architecture in Figure 3.

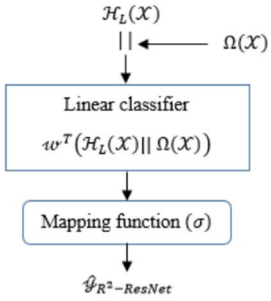

Fig. 3. Sig-R^{2}ResNet Functional Architecture for Residual Map Refinement. Here, '$\|$' denotes concatenation.

It is known and observed that the time series signals are usually non-stationary, sometimes noisy too. In order to extract faithful depiction of feature space Ω, we propose for designing Ω that augments the representation space through exploiting the temporal, statistical, wavelet and spectral properties of $\mathcal{X}$ [15, 16, 18]. The features in Ω can potentially identify the intrinsic patterns of the given time series.

From Eqn. (3):

$$\left.\frac{\partial \varepsilon}{\partial \mathcal{H}_L}\right|_{Sig-R^2 ResNet} = \frac{\partial \varepsilon}{\partial \mathcal{H}_L}\left(Cat\left(\left(1 + \frac{\partial\left(\sum_{l=1}^{L-1} \mathcal{G}_l\left(\mathcal{H}_l(\mathcal{X})\right)\right)}{\partial \mathcal{H}_l}\right), \Omega(\mathcal{X})\right)\right) \quad (4)$$

Where *Cat* denotes concatenation operation ($\|$). Eqn. (4) displays a nice property. Residual mapping is refined at the end of the learning process by embedded feature vectors $\Omega(\mathcal{X})$ so that residual learning completes first, then refined by newer representation $\Omega(\mathcal{X})$. Consequently, Sig-R^{2}ResNet does not distort the philosophy of residual mapping. Therefore, Sig-R^{2}ResNet retains the elegant properties of classical ResNet including the elimination of vanishing gradient problem. In fact, the fine-grained feature vectors of $\Omega(\mathcal{X})$ continuously refine the residual map for inducing the learning with newer representations. Therefore, we expect more capable residual learning without compromising with the associated advantages of residual network.

C. Auto-tuned L_1 Regularization

The risk of overfitting on training dataset when the model attempts to fit to a high-dimension feature vector can be minimized by reducing the complexity of the model through regularization. We expect the Sig-R^{2}ResNet model through augmented feature space by $\Omega(\mathcal{X})$ is overly complex in order to be able to capture the finer signatures in training time series

datasets. Hence, the regularization process requires to provide sparser solution and L_1 regularization (Lasso) [20] is the ideal choice. One of the salient aspects of L_1 regularization is its inherent contribution towards feature selection. The typical learning problem is to estimate a parametrized function with vector of weight parameters ω to the objective function J such that the regularized objective function $\tilde{J}$ becomes:

$$\tilde{J}(\omega; \mathcal{X}_{Train}, \mathcal{L}_{Train}) = J(\omega; \mathcal{X}_{Train}, \mathcal{L}_{Train}) + \lambda_1 \|\omega\|_1$$

Where, the hyperparameter $\lambda_1 \in (0, \infty]$ is the L_1 regularization factor. The value of λ_1 plays an important role to control the sparsity in the model. When $\lambda_1 \to 0$ or the value of λ_1 is too low, the regularization process would have no impact (overfitment problem), whereas, high value of λ_1 results in high amount of sparsity with large number of $\omega = 0$ (underfitment problem). In order to avoid both of the situations, we need to find λ_1 that is not predefined (or fixed) but depends on the training data distribution. Our intuition is that randomness or quantitatively the information uncertainty in a training set disturbs the learning process more than a regular training data distribution, where the information uncertainty is measured by Shannon entropy $Entr(.)$. For a discrete random variable X with outcomes $x_1, x_2, \dots, x_n$ and $pr(x_i)$ denotes the probability of x_i, $Entr(X) = \sum_{i=1}^{n} pr(x_i) \log_2 \left(\frac{1}{pr(x_i)}\right)$.

We intuite that when Shannon entropy among the training instances of $Entr(\mathcal{X}_{Train})$ is more dispersed, we require stricter regularization or more sparsity or higher λ_1 for restricting the impact of learnability due to the randomness in the training distribution. The value of λ_1 for training dataset $\mathcal{X}_{Train}$ is quantitavely estimated by the index of dispersion of the entropy of $\mathcal{X}_{Train}$ as described below. It is to be noted that for different types of datasets, λ_1 changes as per the following algorithm and λ_1 auto-tunes with the overall distribution of the training instances.

Algorithm 1- Finding λ_1

1: **Input**: $\mathcal{X}_{Train}$, scale factor: λ_{scale}
2: **Output**: λ_1
3: **Compute** $Entr(\mathcal{X}_{Train})$, where $\mathcal{X}_{Train}$ consists of number of instances, $\mathcal{X}_{Train} = [\mathcal{X}^1_{Train}, \mathcal{X}^2_{Train}, \dots, \mathcal{X}^M_{Train}]$, and each of the instances consist of T number of samples, $\mathcal{X}^m_{Train} \in \mathbb{R}^T$. We compute $Entr(\mathcal{X}^m_{Train})$, $m = 1,2,3, \dots M$, for each of the M number of training instances in $\mathcal{X}_{Train}$. $Entr(\mathcal{X}_{Train})$ is a vector that consists of M number of Shannon entropy values.
4: **Compute** $Entr_{var} = variance\left(Entr(\mathcal{X}_{Train})\right)$
5: **Compute** $Entr_{mean} = mean\left(Entr(\mathcal{X}_{Train})\right)$
6: **Index of dispersion:** $Entr_{ID} = \frac{Entr_{var}}{Entr_{mean}}$
7: $\lambda_1 = \lambda_{scale} \times Entr_{ID}$
8: **END**

D. Mod-Adam: Modified Adam Optimization for Learning Rate Adaptation

In order to get smoother convergence towards the optimal network parameters (ω), we need to adjust the learning rate to decay over the iterations or epochs. The well-known Adam optimization exponentially smoothens out the first-order gradient for incorporating momentum into the update process, $\omega_\xi \leftarrow \omega_{\xi-1} - \alpha g_{Adam}(\omega_{\xi-1})$. While, $g_{Adam}(\omega_{\xi-1})$ is dependent on the gradient response through first-order gradient-based optimization, whereas the learning rate α remains unchanged throughout the learning process. Adam optimization or the current state of the learning rate optimizer [24- 26] are oblivious to the validation loss. Mod-Adam updates the learning rate α by observing how the validation loss behaves over the epochs. The intuition is that when validation loss does not change over number of epochs, the stochastic optimization needs to be adjusted by slowing down α. The proposed Mod-Adam learning rate adaptation updates the weights as: $\omega_\xi \leftarrow \omega_{\xi-1} - \alpha_\xi^{Mod-Adam} \times \frac{\widehat{m_\xi}}{\sqrt{\widehat{\mu_\xi}} + \varrho}$, where $\alpha_\xi^{Mod-Adam}$ is the learning rate by Mod-Adam at ξ^{th} epoch, $\widehat{m_\xi}$ is the bias-corrected first moment estimate and $\widehat{\mu_\xi}$ is the bias-corrected second raw moment estimate from the Adam optimizer [24]. The decay rate of $\alpha_\xi^{Mod-Adam}$ depends on the distribution of $\mathcal{X}_{Train}$. Quantitatively, Shannon entropy provides the numerical estimate of the distribution of $\mathcal{X}_{Train}$. The value of $\alpha_\xi^{Mod-Adam}$ can vary at each of the instances in $\mathcal{X}_{Train} = [\mathcal{X}^1_{Train}, \mathcal{X}^2_{Train}, \mathcal{X}^3_{Train}, \dots, \mathcal{X}^M_{Train}]$, and may change after certain epochs in the learning process. The decay of $\alpha_\xi^{Mod-Adam}$ is exponential by factor α_{exp} (usually, $\alpha_{exp} = 0.5$) and dependent on the validation loss observation window ϑ. The observation window ϑ is computed from Shannon entropy $Entr(\mathcal{X}^m_{Train})$ of the training instance with respect to the minimum and maximum observation windows: ϑ_{min} and ϑ_{max} respectively. Observation windows are basically number of epochs, $\vartheta_{min}, \vartheta_{max} \in \mathbb{Z}_+$. We suggest $\vartheta_{max} = \left\lfloor \sqrt{Total\_number\_of\_epochs} \right\rfloor$. When the validation loss does not improve over the observation window $\vartheta \in \mathbb{Z}_+$, the learning rate decays by the factor α_{exp}. Thus, the learning rate $\alpha_\xi^{Mod-Adam}$ or the weight (network parameter) updation process responds to the trend in generalizability of the model and decays for smoother gradient descent.

Algorithm 2- Mod-Adam: Algorithm for finding $\alpha_\xi^{Mod-Adam}$ and $\omega_\xi^{Mod-Adam}$ (Showing for a single training instance $\mathcal{X}^1_{Train}$)

1: **Input**: $\mathcal{X}^1_{Train}$, minimum observation window ϑ_{min}, maximum observation window ϑ_{max} ($\vartheta_{max} = \left\lfloor \sqrt{Total\_number\_of\_epochs} \right\rfloor$), initial learning rate α_0 (usually, $\alpha_0 = 0.001$), α_{exp} (usually, $\alpha_{exp} = 0.5$), $\epsilon^{\xi-1}, \epsilon^{\xi}$ denotes validation loss at $(\xi - 1)^{\text{th}}$ and ξ^{th} epochs respectively.
2: **Output**: $\omega_\xi^{Mod-Adam}$ (updated network parameter at ξ^{th} epoch)
3: $\alpha_{\xi=0}^{Mod-Adam} = \alpha_0$
4: **Compute** $\psi^1 = Entr(\mathcal{X}^1_{Train})$
5: **Compute** $\hat{\vartheta}_{max} = \frac{\vartheta_{max}}{\psi^1}$ (Corrected maximum observation window)
6: **Compute** observation window: $\vartheta = \lfloor max(\hat{\vartheta}_{max}, \vartheta_{min}) \rfloor$
7: **IF** $(\epsilon^{\xi-1} - \epsilon^{\xi}) \geq 0$ at ξ^{th} epoch for consecutive ϑ number of epochs: $\alpha_\xi^{Mod-Adam} \leftarrow \left(\alpha_{\xi-1}^{Mod-Adam}\right)^{\alpha_{exp}}$ (learning rate updated at ξ^{th} epoch)
8: $\omega_\xi \leftarrow \omega_{\xi-1} - \alpha_\xi^{Mod-Adam} \times \frac{\widehat{m_\xi}}{\sqrt{\widehat{\mu_\xi}} + \varrho}$, $\widehat{m_\xi}$ is the bias-corrected first moment estimate and $\widehat{\mu_\xi}$ is the bias-corrected second raw moment estimate from the Adam optimizer [24]. For computing $\widehat{m_\xi}$, $\widehat{\mu_\xi}$ two hyperparameters are to be considered: β_1 and β_2; the values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varrho = 10^{-8}$ as described in [24].
9: **END**

IV. Experimental Evaluation

Sig-R^{2}ResNet for TSC consists of three residual blocks and each of the residual blocks consists of number of convolution networks of stride length= 1 with Batch Normalization (BN) and ReLU activation function. The first residual block consist of three convolution networks with kernel size [8, 5, 3], second residual block consists of six convolution networks with kernel size [8, 7, 6, 5, 4, 3] and the third residual block consists of of three convolution networks with kernel size [8, 5, 3]. Sig-R^{2}ResNet additionally consists of auto-tuned regularizer to dynamically vary the learning rate and to set the L_1 regularization factor as depicted in Algorithm 1. Learning rate is dynamically configured by our proposed Mod-Adam algorithm (Algorithm 2). Important network parameters are described in Table I. We have considered total 392 features (including long-term and short-term features) derived from the base transformations [13] that are concatenated at the fully connected network layer. Validation splitting from training dataset plays an important role in Sig-R^{2}ResNet particularly for Algorithm 2. We minimize the bias in validation set splitting by using five-fold cross-validation with 15 different random seeds. The reported performance is the mode of those settings. Thus, the human bias of fixed random seed setting in validation set splitting is eliminated to some extent.

TABLE I. Sig-R^{2}ResNet Hyperparameters

Total number of iterations or epochs	1500
λ_{scale}: L_1 regularization scale factor for finding λ_1	10^{-3}
ϑ_{min}: minimum observation window	2
β: Number of unsupervised features)	392

We present the most important results of this paper in Table II and Table IV. The results depict the performance over the test datasets from UCR time series database [7, 8]. The obtained results of Sig-R^{2}ResNet demonstrate a stellar achievement in solving TSC problems.

Data description: For fair comparison with relevant state-of-the-art, such that only reported (or published) test results are compared, we have divided the comparative study in two Tables. In Table II, total 30 UCR datasets are considered as reported in [13] and in second table (Table IV), 44 datasets are compared (as reported in similar baseline [5]). The complete description of all the datasets are publicly available [7, 8]. These datasets are from number of heterogeneous domains (mainly comprising of IoT applications) like medical (ECG), automobile (FordA), electrical (small kitchen) and many different others with number of training instances varying [20, 8926], number of samples at each of the training instances varying [60, 1092], and the number of classes varying [2, 37].

Performance metric: We use test accuracy as the performance benchmark following the reports in [13] and [5]. The model is trained by the training datasets and learned model is tested over the test datasets as per the datasets available in [7, 8], http://www.timeseriesclassification.com/dataset.php.

State-of-the-art algorithms for comparison: We first depict the performance comparison in Table II with our previous work [13], where total 30 number of UCR dataset is considered. We further compare Sig-R^{2}ResNet and have chosen the most successful state-of-the algorithms that attempt to solve the UCR time series datasets, viz. COTE [11], BOSS [12], ResNet [5], and the traditional baseline method 1-NN DTW [10]. We consider ResNet [5] as the most relevant baseline. Table IV demonstrates the test performance of Sig-R^{2}ResNet on the of 44 UCR datasets reported in ResNet [5]. Automatic Machine Learning (AutoML) is one of the current trends to construct trained model without human intervention [19]. We include AutoML as one of the contenders for comparison. The benchmark results as shown in Table IV is publicly available at: http://www.timeseriesclassification.com/dataset.php. The benchmark results in Table IV are the current publicly available best test performance over respective UCR datasets. Our experiment excludes any kind of hyperparameter tuning and the obtained results capture the pure merit of the proposed algorithm Sig-R^{2}ResNet.

Result summary from Table II: 1. Out of 30 UCR datasets, Sig-R^{2}ResNet demonstrates 63.33% best test accuracy among SPGF-C [13] and SPGF-TN-C [13], 2. SPGF-TN-C, which is a mixed domain feature set with SPGF (signal processing) and TimeNet [18] (pre-trained autoencoder) improves the overall test performance over signal processing feature based algorithm SPGF-C, showing the merit of deep learning feature and signal processing feature fusion, 3. The trend of learning method hybridization by incorporating signal processing features is unquestionably observed from the test results of Sig-R^{2}ResNet.

TABLE II. Comparison with state-of-the-art algorithhms (where fusion of signal processing features have been reported)

Dataset	SPGF-C [13]	SPGF-TN-C [13]	**Sig-R^{2}ResNet**
Synthetic	97.70	98.30	**100**
PhalangesOC	81.30	**81.60**	73.11
DistPhalOAG	82.70	**83.70**	81.76
DistPhalOC	84.30	**84.50**	82.91
DistPhalTW	77.70	**78.20**	75.12
MiddlePOAG	77.20	**77.80**	70.20
MiddlePOC	64.50	**66.70**	58.67
MiddletPTW	**63.90**	63.70	60.19
ProximalPOAG	84.90	84.90	**85.08**
ProximaPOC	86.60	87.60	**91.92**
ProximaPhalTW	80.00	80.70	**81.11**
Electric Devices	69.40	72.50	**72.83**
Medical Images	74.20	75.90	**78.82**
Swedish Leaf	**95.60**	**95.60**	92.87
Two patterns	77.90	**100**	**100**
ECG 5000	94.60	94.20	**99.29**
ECG 5 days	95.30	94.90	**98.81**
Wafer	100	**100**	**100**
Chlorine	69.40	72.10	**74.19**
Adiac	75.40	78.80	**86.75**
Strawberry	95.30	95.10	**98.73**
Cricket_X	64.90	73.60	**73.95**
Cricket_Y	61.30	72.00	**76.77**
Cricket_Z	69.20	76.90	**76.98**
Uwave_X	74.40	**82.90**	72.69
Uwave_Y	67.90	**76.30**	70.35
Uwave_Z	71.40	**77.40**	67.27
Yoga	82.30	85.50	**86.94**
FordA	94.20	93.20	**96.81**
FordB	84.20	89.20	**92.93**
Best reported test accuracy	2	12	19

Result summary from Table IV: 1. Out of the total 44 UCR TSC cases, Sig-R^{2}ResNet has created new benchmark results in 25 cases: 59.1% new benchmark created 2. ResNet [5] and AutoML [19] have shown five best accuracy scores, followed by existing strong baseline COTE [11] that has shown three best results.

One of the primary observations from the comparative study (in Table III and Table IV) reveals that Sig-R^{2}ResNet demonstrates consistent results over the datasets from a domain. For example, Sig-R^{2}ResNet has shown poor performance over all types of Middle Phalanx (MiddleP), Uwave (Uwave_X,Y, Z) datasets, whereas it has steadily depicted better performance over Ford (FordA, B), ECG (ECG 5000, ECG 5days) datasets. We can intuitively explain this observation as following. When the domain dynamics is properly captured in Sig-R^{2}ResNet by feature space augmentation, then the residual refinement with regularization and learning rate adaptation would necessarily construct a better learning model.

We further investigate the test results using statistical hypothesis testing. We have considered Wilcoxon Rank-sum Test, which is a nonparametric test to compare non-independent samples. Assuming that the test accuracy results demonstrated by different algorithms are non-independent, Wilcoxon Rank-sum Test is suitable to find out the interdependence among the algorithms over 44 UCR datasets as shown in Table III, where p-values of the Wilcoxon Rank-sum Test (for different stat-of-the-art algorithms and Sig-R^{2}ResNet are shown). We observe that Sig-R^{2}ResNet has close dependencies with the benchmark result and ResNet, which is naturally correlating with the fact that Sig-R^{2}ResNet is an improvement over classical ResNet. In Figure 4, we further depict the Critical Difference (CD) value, which is calculated using the Bonferonni–Dunn post-hoc test [28]. The visual illustration shows the statistically equivalence consistent with the ranking analysis.

TABLE III. INVESTIGATION OF ALGORITHMS WITH P-VALUES OF WILCOXON RANK-SUM TEST. **BOLD** SIGNIFIES THE MOST DEPENDENT CLASSIFICATION RESULTS.

	Benchmark	1-NN DTW	BOSS	COTE	AutoML	ResNet	**Sig-R^{2}ResNet**
Benchmark	-------	0.00013	0.07	0.308	0.0005	0.321	**0.67**
1-NN DTW	0.00013	-------	0.035	0.0013	**0.94**	0.0014	0.0012
BOSS	0.07	0.035	-------	0.253	0.047	**0.35**	0.183
COTE	0.308	0.0013	0.253	-------	0.0037	**0.997**	0.723
AutoML	0.0005	**0.94**	0.047	0.0037	-------	0.0067	0.004
ResNet	0.321	0.0014	0.35	**0.997**	0.0067	-------	0.793
Sig-R^{2}ResNet	0.67	0.0012	0.183	0.723	0.004	**0.793**	-------

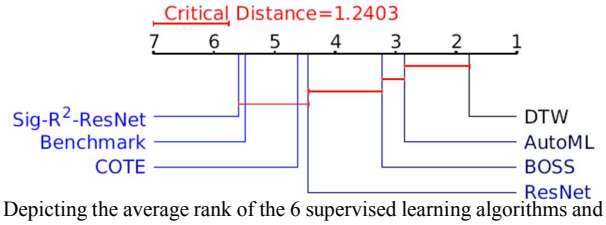

Fig. 4. Depicting the average rank of the 6 supervised learning algorithms and the benchmark over the 44 UCR datasets. The critical difference is 1.2403. The algorithms for which the average rank difference is lesser than the critical difference are connected by a red bar (———).

TABLE IV. TEST ACCURACY COMPARISON OF SIG-R^{2}RESNET WITH STATE-OF-THE-ART ALGORITHMS

Dataset	Length	No. of classes	No. of Training Examples	Benchmark	1-NN DTW [10]	BOSS [12]	COTE [11]	AutoML [19]	ResNet [5]	**Sig-R^{2}ResNet**
Adiac	176	37	390	80.98	60.33	74.94	80.98	86.70	82.60	**86.75**
Chlorine	166	3	467	**84.57**	64.55	65.95	73.55	64.64	82.80	74.19
Cricket_X	300	12	390	**81.40**	76.45	76.35	**81.40**	61.79	82.10	73.95
Mote	84	2	20	91.65	83.20	84.59	90.16	19.25	89.50	**91.87**
Wafer	152	2	1000	99.98	98.40	99.90	99.94	70.31	99.70	**100**
Car	577	4	60	90.18	66.56	85.50	89.90	80.00	93.30	**93.76**
FordA	500	2	3601	96.54	57.69	91.95	95.45	50.24	92.80	**96.81**
FordB	500	2	3636	92.86	64.72	91.10	92.86	88.97	90.00	**92.93**
Starlight	1024	3	1000	97.96	91.42	97.75	97.96	97.94	97.50	**98.02**
Earthquakes	512	2	322	75.92	70.30	74.58	74.68	**81.31**	78.60	78.82
Sony_1	70	2	20	91.84	80.08	89.74	89.90	**100**	98.50	86.34
Sony_2	65	2	27	95.98	84.60	88.77	95.98	90.56	96.20	**97.43**
Diatom	345	4	16	95.78	95.81	93.93	92.47	57.52	93.10	**97.52**
Italy_power	24	2	67	97.03	92.25	86.60	97.03	95.34	96.00	**97.11**
Trace	275	4	100	99.99	99.97	99.99	99.99	**100**	**100**	**100**
Lightning2	637	15	60	**83.70**	82.29	81.00	78.47	57.38	75.40	75.34
Lightning7	319	7	70	79.95	69.83	66.56	79.95	60.27	**83.60**	75.12
Cricket_Y	300	12	390	**81.49**	73.80	74.92	**81.49**	55.90	80.50	76.77
Cricket_Z	300	12	390	**82.69**	77.18	77.57	**82.69**	63.08	81.30	76.98
DistPhaloc	80	2	600	82.12	76.11	81.46	80.45	80.00	82.00	**82.91**
DistPhalTW	80	6	400	69.32	60.46	67.30	69.32	68.34	74.00	**75.12**
ECG 5000	140	5	500	94.61	92.56	94.04	94.61	93.18	93.10	**99.29**
ECG 5 days	136	2	23	98.62	76.03	98.33	98.62	83.16	95.50	**98.81**
Electric Devices	96	7	8926	**89.54**	77.14	79.95	88.28	70.19	72.80	72.80
Medical Images	99	10	381	78.50	74.11	71.45	78.50	76.71	77.20	**78.82**
Strawberry	235	2	613	97.43	95.53	97.02	96.31	95.95	95.80	**98.73**

Synthetic	60	6	300	99.92	99.11	96.78	99.92	90.72	**100**	**100**
Two patterns	128	4	1000	**100**	**100**	99.11	99.97	97.67	**100**	**100**
Uwave_X	315	8	896	76.56	72.57	75.32	76.56	78.43	**78.70**	72.69
Uwave_Y	315	8	896	**77.60**	62.32	66.12	76.56	75.82	66.80	70.35
Uwave_Z	315	8	896	**96.83**	65.65	69.51	75.95	68.12	75.50	67.27
Yoga	426	2	300	90.99	84.88	90.99	89.78	**99.77**	85.80	86.94
CBF	128	3	30	99.80	99.32	99.80	99.80	56.78	99.40	**100**
Arrowhead	251	3	36	87.68	71.98	87.52	87.68	64.57	81.70	**87.94**
Beef	470	5	30	**81.87**	49.70	61.50	76.40	53.33	76.70	74.61
Beetlefly	512	2	20	**94.85**	77.85	94.85	92.10	85.00	80.00	75.13
Haptics	1092	5	155	51.69	39.00	45.90	51.69	44.48	50.60	**51.85**
Large Kitchen	720	3	375	**93.25**	79.65	83.65	90.00	85.60	89.30	87.37
SmallKitchen	720	3	375	81.26	64.01	75.02	78.78	**82.13**	79.70	78.61
Bird Chicken	512	2	20	98.4	83.45	98.40	94.10	90.00	90.00	**100**
Proximal POC	80	2	600	84.84	81.19	86.73	87.08	87.63	91.80	**91.92**
Computers	720	2	250	80.23	70.07	80.23	76.96	72.80	82.40	**82.65**
Insect wingbeat sound	256	11	220	**63.89**	34.94	51.02	63.89	51.82	53.10	51.86
Fish	463	7	175	97.42	76.32	96.87	96.22	80.13	98.90	**99.03**
Best reported test accuracy				13	1	0	3	5	5	26

V. Conclusion

Our proposed Sig-R^{2}ResNet have demonstrated significant performance improvement for heterogeneous time series datasets over the baseline algorithm, as well as the current benchmark and state-of-the-art algorithms. The representation space enrichment with refinement of residual map of residual network along with signal dynamics controlled L_1 regularization and generalization loss dependent learning rate adaptation have made Sig-R^{2}ResNet a strong time series classification model. The interplay of model complexity and regularization have played important role to enable a significantly improved classification model. Our future direction is to introduce a concept of model equilibrium, where model complexity and regularization need to co-operate to form a better-learned model for favorable bias-variance trade-off.

References

[1] K. He, X. Zhang, S.Ren and J. Sun, "Deep residual learning for image recognition," CVPR, 2016.

[2] A. Veit, M. Wilber, S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," NIPS, 2016.

[3] Y. Shen, J. Gao. "Refine or Represent: Residual networks with explicit channel-wise configuration," IJCAI, 2018.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ICML, 2015.

[5] Z. Wang, W. Yan and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," IJCNN, 2017.

[6] C. Murdock, Z. Li, H. Zhou, T. Duerig, "Blockout: Dynamic model selection for hierarchical deep networks," CVPR, 2016.

[7] Y. Chen, E.Keogh, B. Hu, N. Begum, A. Bagnall, A.Mueen, and G. Batista, "The UCR time series classification Archive," 2015.

[8] A. Bagnall, J. Lines, A. Bostrom, J. Large and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," DMKD, 2017.

[9] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," Knowledge and Information Systems. 2005.

[10] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," DMKD, 2015.

[11] A. Bagnall, J. Lines, J. Hills and A. Bostrom, "Time-series classification with COTE: The collective of transformation-based ensembles," TKDE, 2015.

[12] P. Schäfer, "The BOSS is concerned with time series classification in the presence of noise," JDMKD, 2015.

[13] A. Ukil, P. Malhotra, S. Bandyopadhyay, T. Bose, I. Sahu, A. Mukherjee, L. Vig, A. Pal and G. Shroff, "Fusing Features based on Signal Properties and TimeNet for Time Series Classification," ESANN, 2019.

[14] S. Xie, R. Girshick, P. Dollar, Z. Tu and K. He, "Aggregated residual transformations for deep neural networks," CVPR, 2017.

[15] D. B. Percival and A. T. Walden, "Wavelet methods for time series analysis," Cambridge University Press, 2013.

[16] R. M. Gray and L. D. Davisson, "An introduction to statistical signal processing," Cambridge University Press. 2010.

[17] I. Goodfellow, Y. Bengio and A.Courville, "Deep Learning," The MIT Press, 2016.

[18] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pre-trained deep recurrent neural network for time series classiffcation," ESANN, 2017.

[19] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, "Efficient and robust automated machine learning," NIPS. 2015.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," Journal of Royal Statistical Society, Vol. 58, No. 1, pp. 267-288, 1996.

[21] A. Zaeemzadeh, N. Rahnavard, and M. Shah, "Norm-preservation: why residual networks can become extremely deep?", https://arxiv.org/pdf/1805.07477.pdf, 2018.

[22] M. Hardt and T. Ma, "Identity Matters in Deep Learning," ICLR, 2017.

[23] I. Daubechies, "Ten Lectures on Wavelets," CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 1992.

[24] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," ICLR, 2015.

[25] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," JMLR, 2011.

[26] T. Tieleman and G. Hinton, "Lecture 6.5 - RMSProp," COURSERA: Neural Networks for Machine Learning, Technical report, 2012.

[27] A. Ukil, A. J. Jara, L. Marin, "Data-driven automated cardiac health management with robust edge analytics and de-risking," Sensors, vol. 19, issue 12, 2019.

[28] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," J. Mach. Learn. 2006.

[29] C. Puri, A. Ukil, S. Bandyopadhyay, R. Singh, A. Pal, K. Mandana, "iCarMa: Inexpensive Cardiac Arrhythmia Management--An IoT Healthcare Analytics Solution," Proceedings of the first workshop on IoT-enabled healthcare and wellness technologies and systems, pp. 3-8, 2016.