Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life

G. Sateesh Babu, Peilin Zhao, and Xiao-Li Li

Institute for Infocomm Research,
A*STAR, Singapore
{giduthurisb,zhaop,xlli}@i2r.a-star.edu.sg
http://www.i2r.a-star.edu.sg

Abstract. Prognostics technique aims to accurately estimate the Remaining Useful Life (RUL) of a subsystem or a component using sensor data, which has many real world applications. However, many of the existing algorithms are based on linear models, which cannot capture the complex relationship between the sensor data and RUL. Although Multilayer Perceptron (MLP) has been applied to predict RUL, it cannot learn salient features automatically, because of its network structure. A novel deep Convolutional Neural Network (CNN) based regression approach for estimating the RUL is proposed in this paper. Although CNN has been applied on tasks such as computer vision, natural language processing, speech recognition etc, *this is the first attempt* to adopt CNN for RUL estimation in prognostics. Different from the existing CNN structure for computer vision, the convolution and pooling filters in our approach are applied along the temporal dimension over the multi-channel sensor data to incorporate automated feature learning from raw sensor signals in a systematic way. Through the deep architecture, the learned features are the higher-level abstract representation of low-level raw sensor signals. Furthermore, feature learning and RUL estimation are mutually enhanced by the supervised feedback. We compared with several state-of-the-art algorithms on two publicly available data sets to evaluate the effectiveness of this proposed approach. The encouraging results demonstrate that our proposed deep convolutional neural network based regression approach for RUL estimation is not only more efficient but also more accurate.

Keywords: Multivariate time series analysis, Deep Learning, Convolutional Neural Networks, Supervised Learning, Regression Methods, Prognostics, Remaining Useful Life

1 Introduction

Prognostic technologies are very crucial in condition based maintenance for diverse application areas, such as manufacturing, aerospace, automotive, heavy industry, power generation, and transportation. While accessing the degradation from expected operating conditions, prognostic technologies estimate the future performance of a subsystem or a component to make RUL estimation. If we can accurately predict when an engine will fail, then we can make informed maintenance decision in advance to avoid disasters, reduce the maintenance cost, as well as streamline operational activities. This paper proposes a data driven approach to predict RUL of a complex system when the run-to-failure data is available. Existing algorithms in the literature for RUL estimation are either based on multivariate analysis or damage progression analysis [9, 16, 23, 15, 17]. However, it is extremely challenging, if not impossible, to accurately predict RUL without a good feature representation method. It is thus highly desirable to develop a systematical feature representation approach to effectively characterize the nature of signals related to the prognostic tasks.

Recently, a family of learning models has emerged called as deep learning that aim to learn higher level abstractions from the raw data [2, 7], deep learning models doesn't require any hand crafted features by people, instead they will automatically learn a hierarchical feature representation from raw data. In deep learning, a deep architecture with multiple layers is built up for automating feature design. Specifically, each layer in deep architecture performs a non-linear transformation on the outputs of the previous layer, so that through deep learning models the data are represented by a different levels of hierarchy of features. Convolutional neural network, auto-encoders and deep belief network are the mostly known models in deep learning. Depending on the usage of label information, the deep learning models can be learned in either supervised or unsupervised manner. While deep learning models achieve remarkable results in computer vision [11], speech recognition [10], and natural language processing [5]. *To our best knowledge, it has not been exploited in the field of prognostics for RUL estimation.*

Recurrent neural network, a class of deep learning architectures is more intuitive model for time series data [6], however it is suitable for time series future value prediction. In this paper we treat RUL estimation problem as multivariate time series regression and solve it by adapting one particular deep learning model, namely Convolutional Neural Network (CNN) adapted from deep learning model for image classification [1, 13, 12], which is the first attempt to leverage deep learning to estimate RUL in prognostics. The key attribute of CNN is to conduct different processing units (e.g. convolution, pooling, sigmoid/hyperbolic tangent squashing, rectifier and normalization) *alternatively*. Such a variety of processing units can yield an effective representation of local salience of the signals. Additionally, the deep architecture allows multiple layers of these processing units to be stacked, so that this deep learning model can characterize the salience of signals *in different scales*. Therefore, the features extracted by CNN are task dependent and non-handcrafted. Moreover, these features also own more predictive power, since CNN can be learned under the supervision of target values.

Recently, different CNN architectures are applied on multi-channel time series data for activity recognition problem which is a classification task [25, 26, 24]. In [25], a shallow CNN architecture is used consists of only one convolution and one pooling layer, and is restricted to the accelerometer data. In [26, 24], deep CNN architectures are used and in these architectures all convolutional and pooling filters are one-dimensional which applied along the temporally over individual sensor time series separately. Different from classification tasks, in the application on RUL estimation which is a regression task, the convolution and pooling filters in CNN are applied along the temporal dimension over all sensors, and all these feature maps for all sensors need to be unified as a common input for the neural network regressor. Therefore, a novel architecture of CNN is developed in this paper. In the proposed architecture for RUL estimation, convolutional filters in the initial layer are two-dimensional which applied along the temporally over all sensors time series and final neural network regression layer employs squared error loss function which makes the proposed architecture is different from the existing CNN architectures for multi-channel time series data [25, 26, 24]. In the experiments, the proposed CNN based approach for RUL estimation is compared with existing regression based approaches, across two public data sets. Results clearly demonstrates that the proposed approach is accurately predicts RUL than existing approaches significantly.

This paper is structured as follows: First, section 2 briefly describes the problem settings, including data sets, evaluation metrics and data preprocessing steps that are used to evaluate the effectiveness of different algorithms. Then, section 3 describes our proposed novel deep architecture CNN based regression approach for RUL estimation. Next, section 4 presents the performance comparison of the proposed approach with the standard regression algorithms for RUL estimation. Section 5 summarizes the conclusions from this work.

2 Problem Settings

In prognostics, it is an important problem to estimate the RUL of a component or a subsystem, such as the engine of an airplane. Usually, some sensors, e.g. vibration sensors, are used to collect its information that serve as features to estimate RUL. Formally, assume that d sensors with component index i are employed, so a multivariate time series data $X^i \in \mathbb{R}^{d \times n_i}$ can be obtained, where the j-th column of X^i, denoted as $X^i_j \in \mathbb{R}^d$, is a vector consisting of the signals from the d sensors at the j-th time cycle, and $X^i_{n_i}$ denotes the vector of signals when the component fails and n_i is the useful life time of a component i from the starting. Suppose we have N same category components, e.g., N engines, then we can collect a training set of examples $\{X^i_j | i = 1, \ldots, N; j = n_1, \ldots, n_N\}$. Then the task is to construct a model based on the given training set and to perform RUL estimation on a test set $\{Z^i \in \mathbb{R}^{d \times m_i} | i = 1, \ldots, M\}$, where Z^i_j, $j = 1, \ldots, m_i$ are signals when the component works well. Here RUL for a component i in test set is the number of remaining time cycles it works well from m_i-th time cycle before failure. Now let's introduce two benchmark data sets.

Data sets: Two data sets chosen in this work, namely the NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) data set and the PHM 2008 Data Challenge data set [19]. The C-MAPSS data set is further divided into 4 sub-data sets as given in Table 1. Both datasets contain simulated data produced using a model based simulation program C-MAPSS developed by NASA [20].

Both data sets are arranged in an n-by-26 matrix where n corresponds to the number of data points in each component. Each row is a snapshot of data taken during a single operating time cycle and in 26 columns, where 1^{st} column represents the engine number, 2^{nd} column represents the operational cycle number, 3 - 5 columns represent

Table 1. Data sets details (Simulated from C-MAPSS)

Data set	**C-MAPSS**				**PHM**
	FD001	**FD002**	**FD003**	**FD004**	**2008**
Train Trajectories	100	260	100	249	218
Test Trajectories	100	259	100	248	218
Operating Conditions	1	6	1	6	6
Fault Conditions	1	1	2	2	2

the three operating settings, and 6 - 26 columns represent the 21 sensor values. More information about the 21 sensors can be found in [22]. Engine performance can be effected by three operating settings in the data significantly. Each trajectory within the train and test trajectories is assumed to be life-cycle of an engine. While each engine is simulated with different initial conditions, these conditions are considered to be of normal conditions (no faults). For each engine trajectory within the training sets, the last data entry corresponds to the moment the engine is declared unhealthy or failure status. On the other hand, test sets contains data some time before the failure and aim here is to predict RUL in the test set for each engine. For each of the C-MAPSS data set, the actual RUL value of the test trajectories were made available to the public, while the actual RUL value of the test trajectories in PHM 2008 Data Challenge data set is not available.

To fairly compare the estimation model performance on the test data, we need some objective performance measures. In this work, we mainly employ 2 measures: *scoring* function, and Root Mean Square Error (RMSE), which are introduced in details as follows:

Scoring Function: The *scoring* function used in this paper is identical to that used in PHM 2008 Data Challenge. This scoring function is illustrated in Eq. (1), where N is the number of engines in test set, S is the computed *score*, and $h = (Estimated\, RUL - True\, RUL)$.

$$S = \begin{cases} \sum_{i=1}^{N} \left(e^{-\frac{h_i}{13}} - 1\right) \; for \;\; h_i < 0 \\ \sum_{i=1}^{N} \left(e^{\frac{h_i}{10}} - 1\right) \;\; for \;\; h_i \geq 0 \end{cases} \tag{1}$$

This scoring function penalizes late predictions (too late to perform maintenance) more than early predictions (no big harms although it could waste maintenance resources). This is in line with the risk adverse attitude in aerospace industries. However, there are several drawbacks with this function. The most significant drawback being a single outlier (with a much late prediction) would dominate the overall performance *score* (pls. refer to the exponential increase in the right hand side of Figure 1), thus masking the true overall accuracy of the algorithm. Another drawback is the lack of consideration of the prognostic horizon of the algorithm. The prognostic horizon assesses the time before failure which the algorithm is able to accurately estimate the RUL value within a certain confidence level. Finally, this scoring function favors algorithms which artificially lowers the *score* by underestimating RUL. Despite all these shortcomings, the scoring function is still used in this paper to provide comparison results with other methods in literature.

RMSE: In addition to the scoring function, the Root Mean Square Error (RMSE) of estimated RUL's is also employed as a performance measure. RMSE is chosen as it gives equal weight to both early and late predictions. Using RMSE in conjunction with the scoring function would avoid to favor an algorithm which artificially lowers the *score* by underestimating it but resulting in higher RMSE. The RMSE is defined as given below:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} h_i^2} \tag{2}$$

A comparative plot between the two evaluation metrics is shown in Figure 1. It can be observed that at lower absolute error values the scoring function results in lower values than the RMSE. The relative characteristics of the two evaluation metrics will be useful during the discussion of experimental results in the later part of this paper.

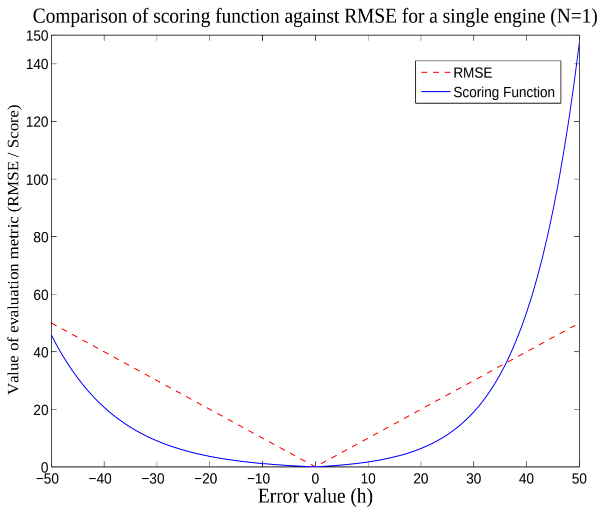

Fig. 1. Comparison of evaluation metric values for different error values

In addition, to learn a model, we need to perform some data pre-processing for which the details are given as follows.

Operating Conditions: Several literature [23, 16, 9], have shown that by plotting the 3 operating setting values, the data points are clustered into six different distinct clusters. This observation is only applicable for data sets with different operating conditions, but data points from FD001 and FD003 in C-MAPSS data set are all clustered at a single point instead – they are single operating condition sub-data sets. These clusters are assumed to correspond to the six different operating conditions. It is therefore possible to include the operating condition history as a feature. This is done for FD002, FD004 and PHM 2008 Data Challenge data sets by adding 6 columns of data (multiple operating condition data sets), representing the number of cycles spent in their respective operating condition since the beginning of the series [16].

Data Normalization: Due to the 6 operating conditions, each of these operating conditions results in disparate sensor values. Therefore prior to any training and testing, it is imperative to do data normalization so that the data points to be within uniform scale range using Eq. (3). As normalization was carried out within the uniform scale range for each sensor and each operating condition, this will ensure equal contribution from all features across all operating conditions [16]. Alternatively, it is also possible to incorporate operating condition information within the data to take into consideration various operating conditions.

$$Norm(x^{c,f}) = \frac{x^{(c,f)} - \mu^{(c,f)}}{\sigma^{(c,f)}}, \ \forall c, f \tag{3}$$

where c represents operating conditions; f represents each of the original 21 sensors. $\mu^{(c,f)}$ is the mean and$\sigma^{(c,f)}$ is the standard deviation in c operating condition.

RUL Target Function: In its simplest form prognostic algorithms are similar to regression problems. However, unlike typical regression problems, an inherent challenge for data driven prognostic problems is to determine the desired output values for each input data point. This is because in real world applications, it is impossible to accurately determine the system health status at each time step without an accurate physics based model. A sensible solution would be to simply assign the desired output as the actual time left before functional failure [16]. This approach however inadvertently implies that the health of the system degrades linearly with usage. An alternative approach is to derive the desired output values based on a suitable degradation model. For this data-set a piece-wise linear degradation model has proposed in [9], which limits the maximum value of the RUL function as illustrated in Figure 2. The maximum value was chosen based on the observations and its numerical value is different for each data-set.

Both these approaches have their own advantages. The piece-wise linear RUL target function is more likely to prevent the algorithm from overestimating the RUL. In addition, it is also a more logical model as the degradation of the system typically only starts after a certain degree of usage. On the other hand, the linear RUL function follows the definition of RUL in the strictest sense which defined as the time to failure. Therefore, the plot of time left of a system against the time passed naturally results in a linear function. However, it should be noted that in cases where knowledge of a suitable degradation model is unavailable, the linear model is the most natural choice to use.

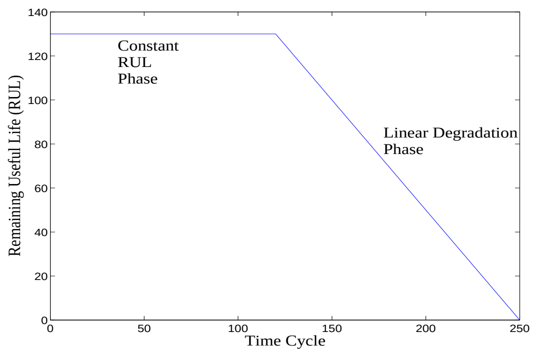

Fig. 2. Piece-wise Linear RUL Target Function

3 Deep Convolution Neural Network for RUL Estimation

This section presents the architecture of deep learning CNN for RUL estimation from multi-variate time series sensor signals. The inputs are normalized sensor signals in addition to the extracted features corresponding to the operating condition history. The target values are the RUL of system at corresponding time cycle. The considered target RUL function is a piece-wise linear function as described in the previous sections.

Convolutional neural networks have great potential to identify the various salient patterns of sensor signals. Specifically, lower layers processing units obtain the local salience of the signals. The higher layers processing units obtain the salient patterns of signals at high-level representation. Note that each layer may have a number of convolution or pooling operators (specified by different parameters) as described below, so multiple salient patterns learned from different aspects are jointly considered in the CNN. When these operators with the same parameters are applied on local signals (or their mapping) at different time segments, a form of translation invariance is obtained [8, 2, 7]. Consequently, what matters is only the salient patterns of signals instead of their positions or scales. However, in RUL estimation we confront with multiple channels of time series signals, in which the traditional CNN cannot be used directly. The challenges in our problem include (i) Processing units in CNN need to be applied along temporal dimension and (ii) Sharing or unifying the units in CNN among multiple sensors. In what follows, we will define the convolution and pooling filters along the temporal dimension, and then present the entire architecture of the CNN used in RUL estimation.

3.1 Architecture

We start with the notations used in the CNN. A sliding window strategy is adopted to segment the time series signal into a collection of short pieces of signals. Specifically, an instance used by the CNN is a two-dimensional matrix containing r data samples each sample with D attributes (In case of single operating condition sub-data sets D

attributes are taken as d raw sensor signals and in case of multiple operating condition sub-data sets D attributes includes d raw sensor signals along with extracted features corresponding to the operating condition history as explained in operating condition subsection in problem settings section). Here, r is chosen to be as the sampling rate (15 used in the experiments because one of the test engine trajectories has only 15 time cycle data samples), and the step size of sliding a window is chosen to be 1. One may choose larger step size to decrease the amount of the instances for lesser computational cost. For training data, the true RUL of the matrix instance is determined by the true RUL of the last record.

In this proposed architecture as shown in Figure 3, conventional CNN is modified and applied to multi-variate time series regression as follows: On each segmented multi-variate time series we perform feature learning jointly. At the end of feature learning, we concatenate a normal multi-layer perceptron (MLP) for RUL estimation. Specifically in this work, we use 2-pairs of convolution layers and pooling layers, and one normal fully connected multi layered perceptron. It includes D-channel inputs and length of each input is 15. This segmented multi-variate time series ($D \times 15$) is fed into a 2-stages of convolution and pooling layers. Then, we concatenate all end layer feature maps into a vector as the MLP input for RUL estimation. Training stage involves the CNN parameters estimation by standard back propagation algorithm using stochastic gradient descent method to optimize objective function, which is cumulative square error of the CNN model.

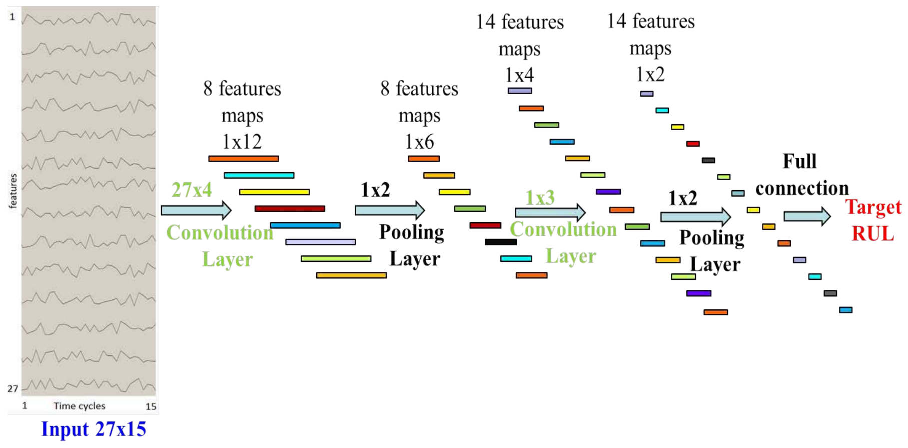

Fig. 3. Proposed CNN Architecture for RUL estimation on PHM 2008 Data Challenge Data set. This architecture consists of segmented multi-variate time series input, 2 convolutional filtering layers, 2 pooling filtering layers, and one fully connected layer.

Convolution Layer: In the convolution layers, the previous layer's feature maps are convolved with several convolutional kernels (to be learned in the training process). The output of the convolution operators added by a bias (to be learned) and the feature map for next layer is computed through the activation function. The output feature map of convolution layer computed as given below:

$$\mathbf{x}_j^l = sigm\left(\mathbf{z}_j^l\right), \quad \mathbf{z}_j^l = \sum_i \mathbf{x}_i^{l-1} * \mathbf{k}_{ij}^l + b_j^l \tag{4}$$

Where $*$ denotes the convolution operator, $\mathbf{x}_i^{l-1}$ and $\mathbf{x}_j^l$ are the convolution filter input and output, $sigm()$ denotes the sigmoid function, and $\mathbf{z}_j^l$ is the input of non-linear sigmoid function. Sigmoid function is used due to its simplicity. We apply convolution filters of size $D \times 4$ in the first convolution layer. In the second convolution layer we apply convolution filters of size 1×3.

Pooling Layer: In the pooling layers, the input features are sub-sampled by suitable factor such that the feature maps resolution is reduced to increase the invariance of features to distortions on the inputs. We utilize $average$ pooling without overlapping for all stage in our work. The input feature-maps are partitioned by the average pooling and results into a set of non-overlapping regions. For each sub-region output is the average value. Pooling layer output feature map is computed as given below:

$$\mathbf{x}_j^{l+1} = down\left(\mathbf{x}_j^l\right) \tag{5}$$

Where $\mathbf{x}_j^l$ is the input and $\mathbf{x}_j^{l+1}$ is the output of pooling layer, and $down(.)$ represents the sub-sampling function for $average$ pooling. We apply pooling filters of size 1×2 in the first and second pooling layers.

3.2 Training Process

As in traditional MLP training for regression task, we used the squared error loss function in our CNN based architecture defined as: $E = \frac{1}{2}(y(t) - y^*(t))^2$, where $y^*(t)$ is the predicted RUL value and $y(t)$ is the target RUL of the t-th training sample. In the training of our CNN model, we utilize stochastic gradient descent based optimization method for optimal parameters estimation of the network and back propagation algorithm to minimize the loss function [14]. Training procedure includes three cascaded phases of forward propagation, backward propagation and the application of gradients.

Forward Propagation: The objective of the forward propagation is to determine the predicted output of CNN model on segmented multi-variate time series input. Specifically, each layer output feature maps are computed. As mentioned in the before sections, each stage contains convolution layer followed by pooling layer. We compute the output of convolution and pooling layers using Eqs. (4) and (5) respectively. Eventually, a single fully connected layer is connected with feature extractor.

Backward Propagation: Once one iteration of forward propagation is done, we will have the error value, with the squared error loss function. The predicted error propagates back on each layer parameters from last layer to first layer, derivatives chain commonly applied for this procedure.

For the backward propagation of errors in the second stage pooling layer, the $\mathbf{x}_j^{l-1}$'s derivative is calculated by the up-sampling function $up(.)$, it is an inverse operation of the sub-sampling function $down(.)$

$$\frac{\partial E}{\partial \mathbf{x}_j^{l-1}} = up(\frac{\partial E}{\partial \mathbf{x}_j^{l}}) \tag{6}$$

In the second stage feature extraction layer, $\mathbf{z}_j^l$'s derivative is calculated as same in hidden layer of MLP.

$$\delta_j^l = \frac{\partial E}{\partial \mathbf{z}_j^l} = \frac{\partial E}{\partial \mathbf{x}_j^l}\frac{\partial \mathbf{x}_j^l}{\partial \mathbf{z}_j^l} = sigm'(\mathbf{z}_j^l) \odot up(\frac{\partial E}{\partial \mathbf{x}_j^{l+1}}) \tag{7}$$

In the above equation element wise product is denoted by $\odot$ symbol and bias derivative is calculated by summating all values in δ_j^l as given below:

$$\frac{\partial E}{\partial b_j^l} = \sum_u (\delta_j^l)_u \tag{8}$$

The kernel weight $\mathbf{k}_{ij}^l$'s derivative is calculated by summating all values related the kernel and it is calculated with convolution operation as given below:

$$\frac{\partial E}{\partial \mathbf{k}_{ij}^l} = \frac{\partial E}{\partial \mathbf{z}_j^l}\frac{\partial \mathbf{z}_j^l}{\partial \mathbf{k}_{ij}^l} = \delta_j^l * reverse(\mathbf{x}_i^{l-1}) \tag{9}$$

Where $reverse(.)$ is the function of reversing corresponding feature extractor. At the end, we calculate $\mathbf{x}_i^{l-1}$'s derivative as given below:

$$\frac{\partial E}{\partial \mathbf{x}_i^{l-1}} = \sum_j \frac{\partial E}{\partial \mathbf{z}_j^l}\frac{\partial \mathbf{z}_j^l}{\partial \mathbf{x}_i^{l-1}} = \sum_j pad(\delta_j^l) * reverse(\mathbf{k}_{ij}^l) \tag{10}$$

In the above equation $pad(.)$ denotes the padding function, it pads zeros to δ_j^l at both ends. Specifically, $pad(.)$ function will pad at each end of δ_j^l with $n_2^l - 1$ zeros, where n_2^l is the size of $\mathbf{k}_{ij}^l$.

Apply Gradients: After the calculation of values of parameters derivatives, we can apply them to update parameters. Assume that the cost function that we want to minimize is $E(\mathbf{w})$. Gradient descent tells us to modify weights $\mathbf{w}$ in the direction of steepest descent in E:

$$w_{ij}^l = w_{ij}^l - \eta \frac{\partial E}{\partial w_{ij}^l} \tag{11}$$

Where η is the learning rate, the learning rate is a parameter that determines how much an updating step influences the current value of weights, and if it's too large it will have a correspondingly large modification of the weights w_{ij}. More details about forward propagation, backward propagation and application of gradients can be found in [14, 3].

4 Experimental Results

In this section, we have performed extensive experiments for comparison of our proposed CNN based regression model (CNN in short) with three regression algorithms in the state-of-the-art, including Multi-layer Perceptron (MLP) [18], Support Vector Regression (SVR) [4] and Relevance Vector Regression (RVR) [21], on two publicly available data sets. The tunable parameters of all the four techniques, namely CNN, MLP, SVR and RVR, are chosen using standard 5-fold cross-validation procedure based on the *training set* only, where we tune their parameter values for training these models on the randomly selected four folds and choose their final values that give the best results in the last fold.

4.1 Results on C-MAPSS Data set

The four algorithms were tested on four C-MAPSS sub-data sets (see Table 1). Table 2 illustrates their comparison results across four sub-data sets in terms of RMSE values. It is observed that CNN achieved the lower RMSE values consistently on all the sub-data sets than MLP, SVR and RVR, regardless of the operating conditions, indicating the proposed deep learning method can find more informative features than shallow features and features from naive MLP network. Among the four methods, MLP achieved higher RMSE values on all the four sub-data sets than the remaining methods, signifying that naive deep model can even harm the performance and further verified the necessity to explore modern deep learning techniques. SVR achieved the lower RMSE values than MLP and RVR on single operating condition data sets, i.e. the first and third sub-data sets. Furthermore, RVR achieved the lower RMSE values than MLP and SVR on multiple operating condition data sets, i.e. the second and fourth sub-data sets. This demonstrates that none of the existing traditional methods can beat the others consistently, while our proposed CNN method consistently achieves significantly better results across multiple data sets.

Similarly, in the same C-MAPSS data sets, Table 3 describes the comparison results for all the four methods in terms of the evaluation *scores*, illustrated as *scoring function* in Figure 1. It is observed that CNN achieved lower (better) score values than the MLP, SVR and RVR on multi operating condition data sets, i.e. second and fourth sub-data sets, as well as on 1 single operating condition data set, i.e. first sub-data set. Among the four methods MLP achieved higher score values (worst results) on all the four sub-data sets than remaining methods regardless of the operating conditions. CNN achieved slightly higher (worse) scores than the RVR on one single operating condition data set, i.e. third sub-data set, even though the RMSE values are lower. Coupled with the characteristics of each evaluation metric (Figure 1), it implies that the slightly high score

Table 2. *RMSE* for various algorithms on C-MAPSS data set

Algorithms	C-MAPSS Data sets			
	FD001	FD002	FD003	FD004
MLP	37.5629	80.0301	37.3853	77.3688
SVR	20.9640	41.9963	21.0480	45.3475
RVR	23.7985	31.2956	22.3678	34.3403
CNN	**18.4480**	**30.2944**	**19.8174**	**29.1568**

could be caused by certain outliers in predicting the RUL. Based on these observations, we find that performance of the methods for RUL estimation also depends on their operating conditions.

Table 3. *Scores* for various algorithms on C-MAPSS data set

Algor ithms	C-MAPSS Data sets			
	FD001	FD002	FD003	FD004
MLP	$1.7972*10^4$	$7.8028*10^6$	$1.7409*10^4$	$5.6166*10^6$
SVR	$1.3815*10^3$	$5.8990*10^5$	$1.5983*10^3$	$3.7114*10^5$
RVR	$1.5029*10^3$	$1.7423*10^4$	$\mathbf{1.4316*10^3}$	$2.6509*10^4$
CNN	$\mathbf{1.2867*10^3}$	$\mathbf{1.3570*10^4}$	$1.5962*10^3$	$\mathbf{7.8864*10^3}$

4.2 Results on PHM 2008 Data Challenge Data set

Finally, we also evaluate the performance of the four algorithms on the PHM 2008 Data Challenge test data set. After we execute the 4 algorithms to compute the estimated RULs of 218 engines in the test data set, they were then uploaded to the NASA Data Repository website and a single score was then calculated by the website as the final output.

Table 4. Scores for various algorithms on PHM 2008 Data Challenge test data set

Algorithms	*Score*
MLP	3212
SVR	15886
RVR	8242
CNN	**2056**

We can observe from the results in Table 4, our proposed CNN based approach outperforms the existing regression methods based approaches significantly by producing much lower *score* (see Figure 1), indicating that the predicted failure time from our proposed CNN model is very near to the actual failure time or their ground truth values. Hence, we can conclude that CNN based regression approach is better than the standard shallow architecture based regression methods for RUL estimation.

5 Conclusion

Clearly, accurate estimation of RUL has great benefits and advantages in many real-world applications across different industrial verticals. As the first attempt to adapt deep learning to estimate RUL for prognostic problem, this paper investigated a novel deep architecture CNN based regressor to estimate the RUL of complex system from multi-variate time series data. This proposed deep architecture mainly employs the convolution and pooling layers to capture the salient patterns of the sensor signals at different time scales. All identified salient patterns are systematically unified and finally mapped into the RUL in the estimation model. To evaluate the proposed algorithm, we examined its empirical performance on two public data sets and our experimental results shows that it significantly outperforms the existing state-of-the-art shallow regression models that have been utilized extensively for RUL estimation in literature. As in our future study, we would like to further explore novel deep learning techniques to tackle a variety of emerging real-world problems in prognostics field.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8), 1798–1828 (Aug 2013)
2. Bengio, Y.: Learning deep architectures for AI. Foundations and trends® in Machine Learning 2(1), 1–127 (2009)
3. Bouvrie, J.: Notes on convolutional neural networks (November 2006), `http://cogprints.org/5869/1/cnn_tutorial.pdf`
4. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
5. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning. pp. 160–167. ACM (2008)
6. Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent neural networks and robust time series prediction. IEEE Transactions on Neural Networks 5(2), 240–254 (1994)
7. Deng, L.: A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing (2014)
8. Fukushima, K.: Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36(4), 193–202 (1980)
9. Heimes, F.O.: Recurrent neural networks for remaining useful life estimation. In: International Conference on Prognostics and Health Management, 2008. PHM 2008. pp. 1–6 (Oct 2008)

10. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. Signal Processing Magazine, IEEE 29(6), 82–97 (2012)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
12. LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. pp. 253–256 (May 2010)
13. LeCun, Y., Bengio, Y.: The handbook of brain theory and neural networks. chap. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. MIT Press, Cambridge, MA, USA (1998)
14. LeCun, Y., Bottou, L., Orr, Genevieve, B., Müller, K.R.: Efficient backprop. In: Orr, G., Müller, K.R. (eds.) Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 1524, pp. 9–50. Springer Berlin Heidelberg (1998)
15. Lim, P., Goh, C.K., Tan, K.C., Dutta, P.: Estimation of remaining useful life based on switching kalman filter neural network ensemble. In: Annual Conference of the prognostics and Health Management Society 2014. pp. 1–8 (2014)
16. Peel, L.: Data driven prognostics using a kalman filter ensemble of neural network models. In: International Conference on Prognostics and Health Management, 2008. PHM 2008. pp. 1–6 (Oct 2008)
17. Ramasso, E., Saxena, A.: Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset. In: Annual Conference of the Prognostics and Health Management Society 2014. pp. 1–11 (2014)
18. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Neurocomputing: Foundations of research. chap. Learning Representations by Back-propagating Errors, pp. 696–699. MIT Press, Cambridge, MA, USA (1988), http://dl.acm.org/citation.cfm?id=65669.104451
19. Saxena, A., Goebel, K.: PHM08 challenge data set, NASA AMES prognostics data repository. Tech. rep., Moffett Field, CA (2008)
20. Saxena, A., Goebel, K., Simon, D., Eklund, N.: Damage propagation modeling for aircraft engine run-to-failure simulation. In: International Conference on Prognostics and Health Management, 2008. PHM 2008. pp. 1–9 (Oct 2008)
21. Tipping, M.E.: The relevance vector machine. In: Solla, S.A., Leen, T.K., Müller, K.R. (eds.) Advances in Neural Information Processing Systems. vol. 12, pp. 652–658. MIT Press (2000)
22. Wang, P., Youn, B.D., Hu, C.: A generic probabilistic framework for structural health prognostics and uncertainty management. Mechanical Systems and Signal Processing 28, 622–637 (2012)
23. Wang, T., Yu, J., Siegel, D., Lee, J.: A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In: International Conference on Prognostics and Health Management, 2008. PHM 2008. pp. 1–6 (Oct 2008)
24. Yang, J.B., Nguyen, M.N., San, P.P., Li, X.L., Krishnaswamy, S.: Deep convolutional neural networks on multichannel time series for human activity recognition. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 3995–4001. AAAI Press (2015)
25. Zeng, M., Nguyen, L.T., Yu, B., Mengshoel, O.J., Zhu, J., Wu, P., Zhang, J.: Convolutional neural networks for human activity recognition using mobile sensors. In: Mobile Computing, Applications and Services (MobiCASE), 6th International Conference on. pp. 197–205. IEEE (2014)

26. Zheng, Y., Liu, Q., Chen, E., Ge, Y., Zhao, J.L.: Time series classification using multi-channels deep convolutional neural networks. In: Web-Age Information Management, pp. 298–310. Springer (2014)