OPELRUL: <u>OP</u>timally Weighted <u>E</u>nsemble <u>L</u>earner for <u>R</u>emaining <u>U</u>seful <u>L</u>ife Prediction

Onat Gungor[1,3], Tajana S.Rosing[2], and Baris Aksanli[3]

[1]Department of Electrical and Computer Engineering, University of California San Diego
[2]Department of Computer Science and Engineering, University of California San Diego
[3]Department of Electrical and Computer Engineering, San Diego State University
ogungor@ucsd.edu, tajana@ucsd.edu, baksanli@sdsu.edu

***Abstract*—Smart manufacturing utilizes a smart maintenance approach, constantly observing system data to estimate machine failure. This smart maintenance, also known as predictive maintenance (PDM), estimates time-to-failure of a machine in order to enable advanced maintenance decisions which bring many advantages such as better inventory management and maximization of equipment lifetime. Predicting remaining useful life (RUL) is crucial to achieving an effective PDM system. Among various RUL prediction approaches, data-driven methods are more appealing due to their easier and quicker implementation. We observe that the performance of data-driven methods varies drastically based on the data set and underlying system parameters, thus making it difficult to have a single algorithm and a parameter set that work best for all settings. Our work proposes an optimally weighted ensemble learner for RUL prediction. We first pre-process input data using Kalman filter, then implement five state-of-the-art deep learning models. We formulate a mathematical optimization problem which determines optimal weights for the selected deep learning models. Our ensemble learner increases the prediction performance by up to 23.4% compared to best single prediction method, and up to 12.1% against the best ensemble method.**

I. Introduction

Industry 4.0, the fourth industrial revolution, is the integration of massively deployed smart computing and network technologies in manufacturing settings [1]. Recent developments in the Internet of Things (IoT) and Cloud Computing help revolutionizing these manufacturing systems, creating the Industrial Internet of Things (I-IoT) notion. I-IoT focuses on adopting the IoT to enable the interconnection of anything, anywhere, and at any time in a manufacturing system [1]. This enables higher automation, reliability, and fine-grained control using computer networks to gather huge amounts of data from the connected machines and turn this data into actionable information [2]. This leads to a closed-loop Cyber Physical System (CPS) where computers constantly monitor physical devices and provide feedback on their operating conditions. In I-IoT systems, machines play an essential role in automating and optimizing the production process. Maintenance of physical equipment, machinery, and other infrastructure is a major process for ensuring successful operation by eliminating unplanned downtime, better inventory management, and maximization of device lifetime [3]. The U.S. industry spends \$200 billion per year on equipment maintenance and ineffective maintenance leads to more than \$60 billion loss [4].

There are three maintenance strategies: reactive, preventive, and predictive. Reactive maintenance occurs after a machine failure. Although it is a preferable method for inexpensive systems (e.g. light bulb), it may lead to catastrophic outcomes for critical systems such as an aircraft engine. Preventive maintenance performs pre-determined regular checks. These specific control times may not be cost-efficient when it is extremely early for maintenance. Predictive maintenance (PDM), or prognostics, estimates time-to-failure of a machine using a variety of mathematical approaches. PDM aims to calculate an optimum schedule for maintenance before any failure occurs. PDM has a growing interest in the industry. The global predictive maintenance market size is expected to grow from \$3.0 billion in 2019 to \$10.7 billion by 2024 [5].

PDM has multiple application domains, including anomaly detection, real-time prognostics, and remaining useful life (RUL) prediction [6]. Anomaly detection focuses on discovering sudden changes that can be used for fault source identification and fault isolation. Real-time prognostics create and update models on-the-fly. RUL is described as the remaining time of a machine to perform its functions until it fails. RUL prediction is achieved via similarity-based, health index (HI)-based, and direct data mapping approaches. Recent advances in machine learning (ML) made the last approach very appealing where related studies use various ML-based methods to find these mappings. However, it is difficult to find a single method that works best in different settings to estimate RUL [7]. To address this issue, we formulate an optimization problem to identify the optimal combination of different methods.

Implementing a single prediction method for RUL estimation may not be optimal. By combining different methods, we may obtain better performance. To reach that goal, we propose an optimally weighted ensemble learner by creating a quadratic programming optimization model. Our approach finds the optimal weights for the selected deep learning models so that enhanced RUL predictions can be obtained. We use NASA C-MAPSS data set [8] which is a widely used benchmark data set for PDM [9]. Different than existing works, we pre-process the input data with Kalman filter which helps increasing prediction performance by up to 23.4%. Then, we implement the base deep learning models to obtain single model RUL predictions. Our ensemble learner increases prediction performance by up to 23.4% compared to the best single prediction method, and up to 12.1% against the best ensemble method. Besides, we show that our proposed method outperforms all state-of-the-art RUL prediction approaches. Our proposed solution OPELRUL eliminates the need for choosing a single method, thus enables combining the strengths of multiple methods in an optimal way.

II. Related Work

The research on how to estimate RUL has gained popularity recently due to rapid advances in condition and health monitoring techniques. Fundamentally, RUL prediction methods fall under three main categories: experience based, model-driven (physical) and data-driven. Experience based models utilize expert knowledge and engineering experience. Statically-constructed if-then statements determine the action to be taken when failure occurs. They are generally created specific to a machine, that is why they are really hard to generalize. Model-driven (physical) models incorporate the physics of failure into RUL assessment [10]. The failure mechanism, e.g. fatigue, wear, is captured in a mathematical model, relating the usage of a system or a component to a degradation rate or lifetime prediction. Building these models is extremely difficult and time-consuming due to the complexity and noisy working condition of a machine [11].

Data-driven models use historical sensor data to build ML models. According to a recent report [12], deploying ML-based PDM can reduce system downtime by 20-50% and costs by 5-10%. Traditional ML models require feature extraction and selection steps. Hence, they rely on domain knowledge. State-of-the-art includes many examples that implement different ML methods to predict RUL, such as Multi-Layer Perceptron (MLP), Relevance Vector Regression (RVR), Support Vector Machine (SVM) [6] etc. Recently, deep learning (DL) became more popular than traditional ML due to its success in RUL prediction and breaking dependence on domain knowledge [13]. There are studies which employ sophisticated DL methods such as convolutional neural networks (CNN) [14], long short-term memory (LSTM) networks [13]. There are also hybrid models which combine multiple models to combine the strengths of those, e.g. combination of CNN and LSTM [15].

A single prediction method may not perform the best across different systems (and thus data sets) [16]. Ensemble learning solves this problem where predictions from multiple models are strategically combined. Ensemble methods construct a set of hypotheses and combine those to improve the prediction performance. The advantage of these methods is that they break the assumptions inherent in single prediction algorithms and provide more generalizable and robust models. There are a variety of state-of-the-art ensemble models such as AdaBoost, gradient tree boosting, and random forest regression [17]. We compare our proposed ensemble learner with these current ensemble approaches. For RUL prediction, there are different ensemble model approaches. Li et al. [7] combine multiple traditional ML-based learners (e.g. random forest, classification and regression tree) by using particle swarm optimization and sequential quadratic programming to determine their weights. Similarly, Shi et al. [18] utilize ensemble learning to predict RUL of bearings. These ensemble methods lack of two main elements which lead to limited prediction performance: not using deep learning based methods and not pre-processing the input data. Different than these works, after data pre-processing with discrete Kalman filter, we combine DL based learners optimally by constructing a quadratic programming optimization model to find the optimal weights.

III. Proposed Framework

Traditional ML models require extensive feature extraction and selection steps and they lack of high prediction accuracy. As an alternative, deep learning (DL) allows end-to-end predictive maintenance without extensive feature extraction or domain knowledge. It employs consecutive layers of nonlinear processing to learn the representations of data where the hidden patterns are identified and predicted [19]. DL is promising due to its high prediction accuracy. However, implementing a single DL model may not provide accurate RUL predictions, there is still a huge potential of DL we can utilize. This potential can be reached by an appropriate data pre-processing step and using an ensemble learner where different DL models are combined. To reach that goal, after a rigorous data pre-processing step, we utilize from different DL models and combine them optimally via our ensemble learning framework, shown in Figure 1. It consists of three main modules: data pre-processing, deep learning, and ensemble. Given raw sensor data, our framework outputs RUL prediction values.

A. Data pre-processing Module

Data pre-processing is an indispensable step in any machine learning task. It helps us obtaining more accurate RUL predictions. Provided time series sensor data, this module outputs de-noised selected sensor data. It consists of discrete Kalman filter, data normalization and feature selection steps.

1) Discrete Kalman Filter: In PDM applications, the system depends highly on a variety of sensor data, collected continuously. These data can contain significant noise due to environmental conditions. Thus, it is important to denoise the data before using it. There are a variety of smoothing algorithms for data denoising [20], such as mean filter, Savitzky-Golay filter, etc. We assume that our sensor measurements have additive white Gaussian noise which is a simple and often sufficiently accurate noise model. Thus, we select discrete Kalman filter as our data smoothing methodology. Kalman filter can estimate the state of a dynamic system from a series of measurements that contain noise. There is a two-step iterative calculation procedure: time update (i.e. prediction) and measurement update (i.e. correction). Mathematically, Kalman filter estimates the state $x \in \mathcal{R}^n$ based on an affine transformation of x_{t-1} with an additive Gaussian noise in the time update step [21]:

$$x_t = \Xi \cdot x_{t-1} + \epsilon_t$$

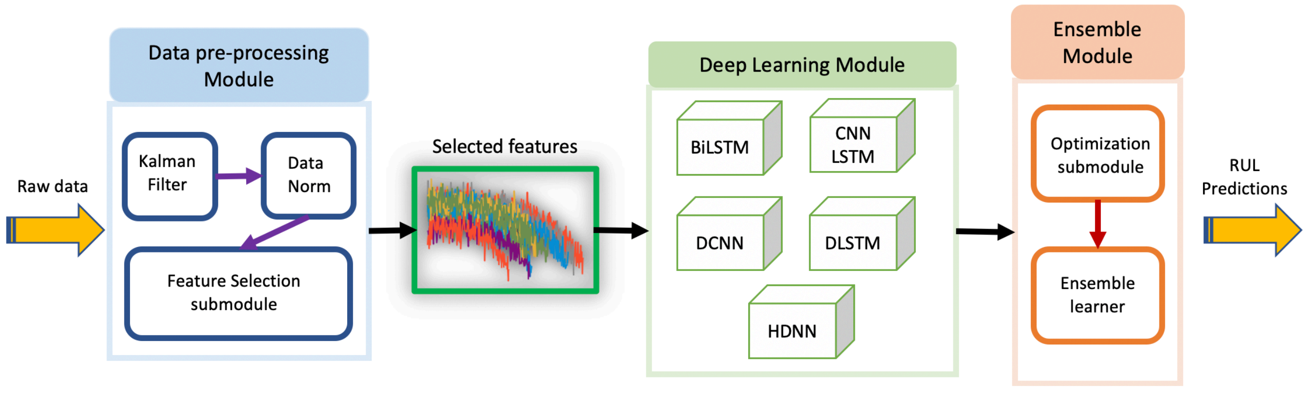


Fig. 1: Proposed Framework for RUL Prediction

In the measurement update, we correct the measurement $y \in \mathcal{R}^m$ (m: number of observations) by an affine transformation of x_t with an additive Gaussian noise:

$$y_t = \Psi \cdot x_t + \phi_t$$

Initial state x_0 is distributed according to a Gaussian distribution with the mean μ_0 and the covariance Σ_0:

$$x_0 \sim \mathcal{G}(\mu_0, \Sigma_0)$$

The random variables ϵ_t and ϕ_t represent the process and measurement noise, respectively. They are assumed to be independent of each other, white, and with Gaussian distributions:

$$p(\epsilon) \sim \mathcal{G}(0, Q)$$

$$p(\phi) \sim \mathcal{G}(0, R)$$

where Q (transition covariance) and R (observation covariance) are process noise and measurement noise covariance matrices, respectively. The $n \times n$ matrix Ξ (transition matrices) relates the state at the previous time step to the state at the current step. The $m \times n$ matrix Ψ (observation matrices) in the measurement equation relates the state to the measurement y_t. We utilize pykalman [21] library for the implementation. To run the filter, we provide sensor measurements y. We set the transition matrices Ξ, observation matrices Ψ, and initial state covariance Σ_0 to 1×1 identity matrices as well as initial state mean μ_0 to the first element of each sensor measurement. For the rest of the parameters (transition covariance Q, and observation covariance R), we use the Expectation-Maximization algorithm which is also available in pykalman library. As an output, we obtain the state estimates x which corresponds to denoised sensor measurements.

2) Data Normalization and Feature Selection: After Kalman filter is applied, we normalize the data using min-max normalization according to Equation 1 where x_i is an input data to be normalized and $\ddot{x}_i$ indicates the normalized data in [0,1]. Data normalization is a fundamental step since sensor measurements can have different ranges. By changing the measurement values to a common scale, we eliminate possible data related problems that can significantly affect the prediction performance.

$$\ddot{x}_i = \frac{x_i - \min x_i}{\max x_i - \min x_i} \tag{1}$$

DL methods do not need extensive domain knowledge, i.e. the end-to-end structure is able to map raw machinery data to targets. However, the data might include numerous features, sometimes making it prohibitively costly to execute DL methods. To reduce this overhead, we select the most relevant features in predicting RUL. First, we use Random Forest (RF) to discover variable importance which is calculated based on the reduction in residual sum of squares. We then perform a trend analysis (i.e. apparent upward/downward change in measurements) to observe whether degradation can be discovered specific to each feature, and select the features that demonstrate trendy behavior across time. Lastly, we perform correlation analysis to detect possible high correlations among sensor measurements. We eliminate the least important features common in selected methodologies (i.e. RF, trend analysis and correlation analysis).

B. Deep Learning Module

Given selected sensor data, deep learning module outputs single prediction model RUL values. As our base learners, we adapt five state-of-the-art deep learning models: BiLSTM [13], CNNLSTM [22], DCNN [14], DLSTM [11] and HDNN [15] due to their success in RUL prediction. We use sliding time window approach to convert time series sensor data into a regression problem. This approach also provides a better feature extraction. The time window represents the number of past observations to be considered and we slide this window from the first observation to the last. To illustrate, provided that the window size is 30, our first window includes the observations from 1 to 30, second window includes observations from 2 to 31 and so on. Accordingly, 2-D input is provided to DL models where the first dimension represents the selected features and the other one has the time sequence of each feature. Selected DL models are as follows:

1) Deep LSTM (DLSTM): Long Short-Term Memory (LSTM) networks are proposed to prevent vanishing and exploding gradient problem. They are specific types of recurrent neural networks (RNN) with special memory cells to store information over longer periods of time. Updates in the memory cell can occur by the activation of three distinctive gates: 1) forget gate (the memory cell is cleared completely), 2) input gate (memory cell stores the received input), and 3) output gate (next neurons obtain the stored knowledge from the memory cell) [23]. Since LSTM can recall information for long periods of time, it is a good fit for RUL prediction tasks. Zheng et al. [11] propose a DLSTM network for enhanced RUL prediction, with 2 consecutive LSTM layers (each with 32 nodes) followed by 2 fully connected feed forward neural networks (each with 8 nodes). Final 1-dimensional output layer provides RUL prediction.

2) Deep Convolutional Neural Network (DCNN): DCNNs use multiple feature extraction stages that can automatically learn hidden representations. Especially, 1-D CNN is common for time series applications, making it a suitable model for RUL prediction. Li et al. [14] propose a DCNN model where input is represented in 2-D where one dimension is the feature number and the other is the time sequence of each feature. They show that the model practically is a 1-D CNN owing to relationship between spatially neighboring features in the data sample. Their network structure consists of five consecutive CNN layers (plus a flattened layer), one fully-connected layer (with 100 nodes) and an output layer with 1 node.

3) Bidirectional LSTM (BiLSTM): LSTM networks only consider past data where hidden states are learned in forward direction. However, in RUL prediction, future sensor data may be as important as the past, i.e. there is a need to take full advantage of the collected sensor data. BiLSTM solves this problem by adding a backward direction to LSTM networks. Wang et al. [13] propose a deep BiLSTM network for RUL prediction. There are two consecutive BiLSTM networks which have 64 and 32 nodes respectively. These two layers are connected to 2 fully connected feed forward neural network layers with 16 and 8 nodes. The final output layer is only one node calculating the RUL prediction.

4) Convolutional Neural Network LSTM (CNNLSTM): While CNNs are good at feature extraction, LSTM networks are capable of building long term time dependencies. By combining these two, one can utilize the benefits of two approaches. Jayasinghe et al. [22] connect convolutional layers with LSTM networks sequentially.

5) Hybrid Deep Neural Network (HDNN): Instead of connecting different network architectures sequentially, Al et al. [15] propose a hybrid deep neural network framework that integrates CNN and LSTM architectures simultaneously and in a parallel fashion. The model structure consists of three paths. The LSTM path (3 LSTM layers) and CNN path (3 CNN layers followed by max pooling layers) are in parallel. These two paths are connected to the fusion path where 3 fully connected neural networks provide the final RUL prediction.

C. Ensemble Module

Here, we present our ensemble module which consists of optimization submodule and ensemble learner. The goal of this module is to optimally combine predictions from the deep learning module in an algorithmic manner such that single prediction performance is improved. Provided deep learning model RUL predictions, this module learns the optimal weights corresponding to the base learners and combine them to obtain OPELRUL predictions.

1) Optimization submodule: To find the optimal weights for the base learners from the deep learning module, we construct a mathematical model to minimize the mean squared error (MSE). MSE can be formulated only using the variance and bias of an estimator $\hat{Y}$:

$$MSE(\hat{Y}) = Variance(\hat{Y}) + Bias^2(\hat{Y}) \tag{2}$$

We consider MSE because we can minimize bias and variance simultaneously by minimizing MSE as shown in Equation 2. Accordingly, we formulate a mathematical optimization model:

$$\text{minimize} \quad \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \left(y_i - \sum_{j=1}^{\mathcal{M}} w_j \hat{y_{ij}}\right)^2 \tag{3}$$

$$\text{subject to} \quad \sum_{j=1}^{\mathcal{M}} w_j = 1 \tag{4}$$

$$w_j \geq 0 \quad \forall j = 1, \ldots, \mathcal{M} \tag{5}$$

In this model, $\mathcal{N}$ is the number of observations, $\mathcal{M}$ is the number of base learners, y_i is the true values for an observation i ($i = 1, \ldots, \mathcal{N}$), $\hat{y_{ij}}$ is the predicted values for an observation i by the base learner j ($j = 1, \ldots, \mathcal{M}$). w_j is the weight corresponding to the base learner j. The objective function (3) minimizes the MSE, constraint (4) ensures that weights sum up to 1, and constraints (5) restrict all weights to be non-negative. This model contains a convex quadratic objective function which can be proven by showing that Hessian (i.e. matrix that organizes all the second partial derivatives of a function) is positive semidefinite (PSD). Without loss of generality, we can rewrite our objective function (3) using L2 norm and matrix notation:

$$\|y - \hat{Y}w\|_2^2$$

where y is the $\mathcal{N}$ dimensional vector holding true values, $\hat{Y}$ is the $\mathcal{N} \times \mathcal{M}$ matrix holding predictions and w is the $\mathcal{M}$ dimensional weight vector. For simpler notation, let ξ be a function that maps w to $\|y - \hat{Y}w\|_2^2$:

$$\xi : w \mapsto \|y - \hat{Y}w\|_2^2 = \|y\|_2^2 - 2y^T\hat{Y}w + \|\hat{Y}w\|_2^2.$$

Note that ξ is twice differentiable. The first and second partial derivatives of ξ with respect to w and w^T are as follows:

$$\frac{\partial \xi}{\partial w} = -2y^T\hat{Y} + 2w^T\hat{Y}^T\hat{Y}$$

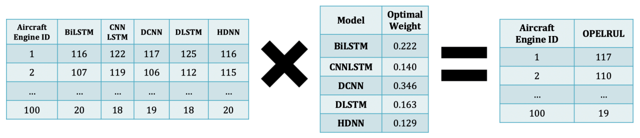
$$\frac{\partial^2 \xi}{\partial w \partial w^T} = 2\hat{Y}^T\hat{Y}$$

We also need to show $\hat{Y}^T\hat{Y}$ is a PSD matrix. Let ζ be an M dimensional vector. We prove that $\hat{Y}^T\hat{Y}$ is PSD:

$$\zeta^T(\hat{Y}^T\hat{Y})\zeta = (\hat{Y}\zeta)^T(\hat{Y}\zeta) = \|\hat{Y}\zeta\|_2^2 \geq 0.$$

We have shown that our objective function is convex which is also quadratic. Our constraint functions are all affine. Hence, we have a quadratic program (QP) for which there is a guarantee that a local minimum is also the global minimum [24]. As an output of the optimization submodule, we retrieve the optimal weights (w^*) for each DL model.

2) Ensemble learner: The role of the ensemble learner is to use the optimal weights and calculate the final RUL prediction values. This is simply performed by taking the weighted average of base learner predictions. To put differently, ensemble learner performs a dot product operation which takes single algorithm RUL predictions and optimal weights. As an output, we obtain OPELRUL remaining useful life predictions. Ensemble learner dot product calculation is illustrated in Figure 2. In this figure, we have single prediction RUL values (coming from the deep learning module) which are multiplied by the optimal weights (coming from the optimization submodule).



Aircraft Engine ID	BiLSTM	CNN LSTM	DCNN	DLSTM	HDNN
1	116	122	117	125	116
2	107	119	106	112	115
...	...	...	...	...	...
100	20	18	19	18	20

Model	Optimal Weight
BiLSTM	0.222
CNNLSTM	0.140
DCNN	0.346
DLSTM	0.163
HDNN	0.129

Aircraft Engine ID	OPELRUL
1	117
2	110
...	...
100	19

Fig. 2: Ensemble Learner Dot Product Calculation

IV. Experimental Analysis

A. Dataset Description

NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) data set is a widely used benchmark for RUL prediction [9]. The engine diagram in Figure 3 depicts the main elements of the engine model [8]. The main components include: fan, compressor, combustor, and turbine. Variety of sensors (e.g. temperature, pressure) placed on these components construct the input data for RUL prediction. More information about this data set (e.g. aircraft engine physical details, etc.) can be found at [8]. There exist four sets of data: FD001$\sim$FD004. Each data set has its own distinctive features (e.g. operating conditions, fault conditions) as indicated in Table I. FD004 is the most complicated (i.e. the highest number of operating and fault conditions) whereas FD001 is the simplest data set. Each data set has separate training and testing sets. While the training data contains the entire lifetime of an engine, test data is terminated at some point before engine failure. Each row represents data during a single operating time cycle with 26 columns: the engine ID, cycle index, three operational settings, and 21 sensor measurements. At the beginning of each time series, the engine is operating normally and develops a fault at some point in the future. The ground-truth RUL values are provided for the test data, and the goal is to predict the RUL before failure on the test data.

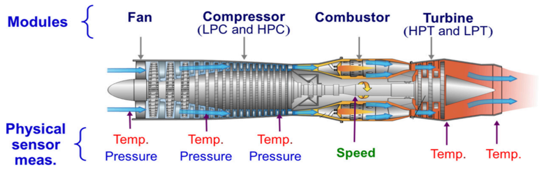


Fig. 3: Engine Diagram Simulated in C-MAPSS [8]

TABLE I: C-MAPSS Data Set

Data Set	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	249
Test trajectories	100	259	100	248
Max/Min cycles for train	362/128	378/128	525/145	543/128
Max/Min cycles for test	303/31	367/21	475/38	486/19
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

RUL Target Function: RUL is generally modeled linearly where its value decreases with time. However, the degradation of the machine performance is not apparent in the beginning of its lifetime and increases when a machine approaches its end of life. For the selected data set, it is proven that piece-wise linear degradation model is more suitable and effective than linear model [25]. Hence, we model RUL using a piece-wise linear function. The maximum RUL limit constant (the break point) is set to 125-time cycles as in [13]. The dotted line in Figure 4 shows the selected RUL target function.

Performance Evaluation Metric: Prediction error (ϵ) is the difference between the estimated RUL (RUL_{est}) and the true RUL (RUL_{true}) (i.e. $\epsilon = RUL_{est} - RUL_{true}$). We use Root Mean Square Error (RMSE) for evaluation, formulated as:

$$RMSE = \sqrt{\frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}} \epsilon_i^2}$$

B. Data pre-processing

Discrete Kalman filter: There are large random fluctuations and noise jamming in CMAPSS dataset, which might impact the RUL prediction performance [26]. We apply Kalman filter on the sensor and operational setting data to reduce the noise. This step is mostly overlooked by the state-of-the-art, but it improves the overall prediction performance. To illustrate, Figure 5 depicts raw sensor data and its Kalman filtered version for the first aircraft engine and sensor 2 from the first dataset. Kalman filter helps smoothing raw sensor data while keeping the trend.

Feature Selection: We select the most relevant features in predicting RUL to (1) increase prediction accuracy, (2) decrease computational complexity, and (3) avoid overfitting. Feature selection is performed for FD001 and FD003 since these two data sets exhibit clear health degradation processes. We first use random forest (RF) to find the feature importance. Figure 6 illustrates nonzero feature importance values for the FD001 and FD003 training data sets. X-axis has feature importance values in [0,1] and y-axis shows different features. We observe that the most informative feature is sensor 11 for both data sets.

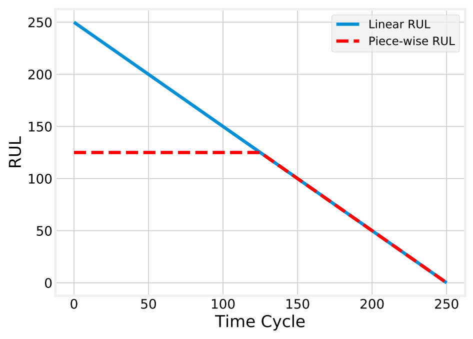


Fig. 4: RUL Target Function

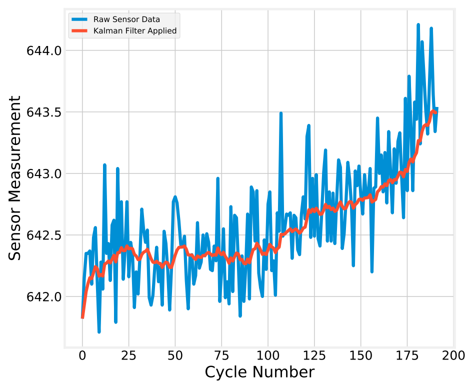


Fig. 5: Kalman Filtered Sensor Data

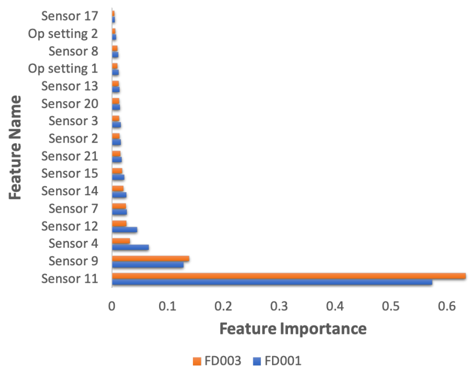


Fig. 6: Random Forest Feature Importance

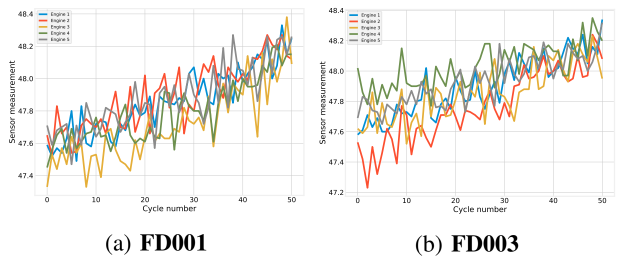


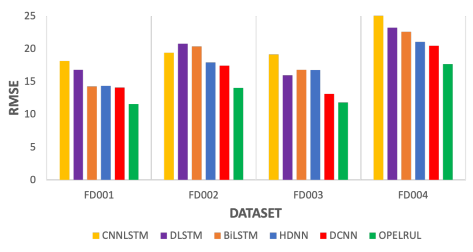
Fig. 7: Trend Analysis for the Most Informative Sensor

We also perform a trend analysis to determine which features exhibit behavior correlated with time. Figure 7 provides an example trend analysis where we provide the most informative sensor (sensor 11) measurements over the last 50 cycles of the first 5 aircraft engines. X-axis shows the cycle numbers and the y-axis provides the sensor measurement values. Each color represents a different aircraft engine. We see trendy behavior across time where sensor measurement values increase with time. In this analysis, we detected that some sensor values stay constant for all engines, providing no useful information in RUL prediction. Based on RF and trend analysis, we eliminate the least important 8 features (operational setting 3 and sensor measurements 1, 5, 6, 10, 16, 18, 19). We also perform a variety of correlation analysis methods (Pearson, Kendall, and Spearman) to detect high correlations among sensor values to remove potential duplicate information. We discovered that sensors 9 and 14 are highly correlated with each other. Thus, we remove sensor 14 as it is less correlated with RUL. Overall, we eliminate 9 features (37% of all features).

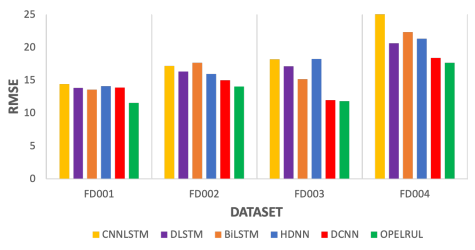
C. Deep Learning Models

Model configurations: We select five DL models: BiLSTM, DCNN, HDNN, DLSTM, and CNNLSTM. For all methods, we adopt a sliding time window approach where we select the window size based on the minimum number of cycles for the test data (in the sliding window approach, only sequences that meet the window-length are considered). As shown in Table I, this value differs across the data sets. To use the entire test data, the window size is selected as 30, 20, 30, and 15 respectively from FD001 to FD004. Previously, it was shown that no considerable further improvement in the prediction performance is attained when this parameter's value is increased [14]. We run all models with the same parameter configuration: *Adam* optimizer with learning rate 0.001, *elu* activation function, *He* initialization, batch size of 512, and a max number of epochs of 250 where callback is activated (patience is set to 10 for validation data). We replicate each experiment 10 times and report average values. All experiments are run on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor. As an output of this module we obtain single model RUL predictions. Overall, we evaluate the DL models without (i.e. baseline) and with Kalman filter.

Baseline models and impact of Kalman filter: Baseline models are the original five DL models with no Kalman filter applied. For Kalman filter added DL models, we reran the baseline experiments and added Kalman filter. Figure 8 demonstrates baseline and Kalman filter added DL model RMSE values with OPELRUL comparison. While x-axis represents data set, y-axis corresponds to RMSE values. Different colors represent the selected DL models plus our proposed approach OPELRUL. For baseline models, we observe that DCNN is the best across all data sets. Different than the baseline models, BiLSTM performs best at the first data set among Kalman filter added DL models. Note that our proposed approach is the best at all instances. This figure clearly shows us the benefit of adding Kalman filter where prediction performance is improved in general. We show the



(a) **Baseline Models+OPELRUL**



(b) **Kalman Filter Added Models+OPELRUL**

Fig. 8: RMSE Comparison

improvement gain of adding Kalman filter in Table II. This table shows that Kalman filter plays an important role in improving predictor performance, by up to 23.4%.

TABLE II: Improvement (%) After Kalman Filter

	Dataset			
	FD001	**FD002**	**FD003**	**FD004**
Average	5.1	**13.1**	8.3	8.5
Maximum	11.3	18.9	**23.4**	15.9

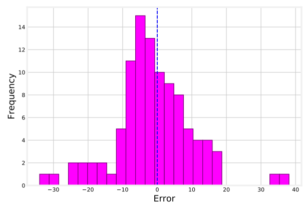
D. Optimally Weighted Ensemble Learner

Our ensemble learner finds the optimal weights corresponding to the base learners and uses those weights to calculate OPELRUL predictions. We formulate and solve the quadratic optimization problem in YALMIP [27] using MOSEK [28] as the solver. Table III presents the optimal weights of the base learners with their average values. As expected, the model assigns more weights to the best model (DCNN) for all data sets, with an average weight of 0.569. Ensemble learner then performs a dot product operation between single DL models RUL predictions and optimal weights.

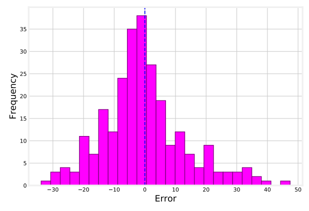
TABLE III: Optimal Weights for Ensemble Learner

	Dataset				
Model	**FD001**	**FD002**	**FD003**	**FD004**	**Average**
BiLSTM	0.222	0.087	0.1	0.005	0.104
CNNLSTM	0.140	0.092	0.002	0.065	0.075
DCNN	**0.346**	**0.471**	**0.843**	**0.618**	**0.569**
DLSTM	0.163	0.151	0.01	0.201	0.131
HDNN	0.129	0.199	0.045	0.111	0.121

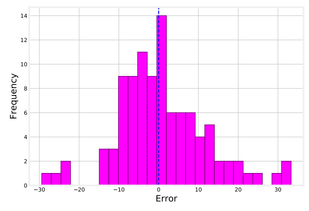
We visualize the prediction error distribution for all data sets in Figure 9. On the horizontal axis, we have the error values between the predicted and the true RUL and on the vertical axis, we have the number of engines corresponding to the error region. We can observe that error varies less for FD001 and FD003, and the corresponding RMSE values are also lower. This is due to low complexity of these data sets



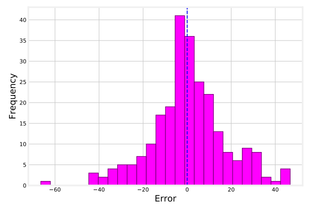
(a) **FD001**



(b) **FD002**



(c) **FD003**



(d) **FD004**

Fig. 9: Distribution Histogram of Prediction Error

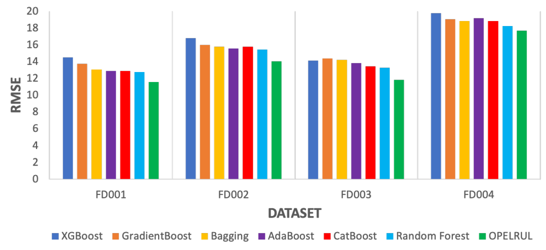


Fig. 10: Ensemble Model Comparison

(i.e. fewer number of operating and fault conditions). There are also outlier (i.e. extreme) points in each sub-figure. This is due to extremely small true RUL values (e.g. 6, 7 cycles) of some aircraft engines.

We compare the performance of our method against the selected deep learning models, other well-known ensemble models, and the state-of-the-art RUL prediction methods.

Comparison with single predictors: We calculate the improvement over the best single prediction algorithm against the best algorithm from both baseline and Kalman filter-added DL models. Table IV shows these results where OPELRUL outperforms single prediction methods by up to 23.4%.

TABLE IV: Improvement (%) Over Best Baseline and Kalman Filter Added DL Method

		Dataset			
		FD001	**FD002**	**FD003**	**FD004**
Best Baseline DL	Average	12.9	**18.2**	9.6	11.2
	Maximum	17.4	21.8	**23.4**	16.5
Best Kalman DL	Average	**8.3**	5.9	1.4	2.9
	Maximum	10.8	**12.4**	4.1	6.1

Comparison with other ensembles: Figure 10 compares OPELRUL with the state-of-the-art ensemble models. OPELRUL is better than other ensemble methods at all data sets, by up to 12.1%.

Comparison with the state-of-the-art: Table V compares OPELRUL with various other state-of-the-art models in the literature [15], [22], [29]. This table directly uses the reported

RMSE values. OPELRUL is the most accurate prediction model among all RUL prediction methods at all data sets.

TABLE V: RMSE State-of-the-art Comparison

Method	Year	Dataset			
		FD001	**FD002**	**FD003**	**FD004**
MLP [29]	2016	37.56	80.03	37.39	77.37
SVR [29]	2016	20.96	42	21.05	45.35
RVR [29]	2016	23.8	31.3	22.37	34.34
CNN [29]	2016	18.45	30.29	19.82	29.16
DLSTM [29]	2017	16.14	24.49	16.18	28.17
ELM [29]	2017	17.27	37.28	18.47	30.96
DBN [29]	2017	15.21	27.12	14.71	29.88
MODBNE [29]	2017	15.04	25.05	12.51	28.66
BLSTM [29]	2018	14.26	21.7	16.33	25.9
RNN [29]	2018	13.44	24.03	13.36	24.02
DCNN [29]	2018	12.61	22.36	12.64	23.31
BiLSTM [29]	2018	13.65	23.18	13.74	24.86
CNNLSTM [22]	2018	23.57	20.45	21.17	21.03
DAG [29]	2019	11.96	20.34	12.46	22.43
HDNN [15]	2019	13.02	15.24	12.22	18.16
OPELRUL	2021	**11.57**	**14.03**	**11.83**	**17.67**

V. Conclusion

Timely maintenance of a machine has become key to equipment health management. Predictive maintenance (PDM) and remaining useful life (RUL) prediction play a crucial role in obtaining the best performance from a production system. By predicting the RUL of a machine, one can adjust the mechanical operation and propose a targeted and optimized maintenance strategy [29]. As a data-driven method, DL performs better than traditional ML approaches in RUL prediction. This work proposes an optimally weighted ensemble learner for RUL prediction. Our framework has a rigorous data pre-processing step and then formulates a mathematical optimization problem to determine optimal weights of the selected DL models. Our ensemble learner increases the prediction performance by up to 23.4% compared to best single prediction method, and up to 12.1% against the best ensemble method. We also show that OPELRUL outperforms state-of-the-art RUL prediction methods at all data sets.

References

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, “A survey on industrial internet of things: A cyber-physical systems perspective,” *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring: A survey,” *arXiv preprint arXiv:1612.07640*, 2016.

[3] M. Kraus and S. Feuerriegel, “Forecasting remaining useful life: Interpretable deep learning approach via variational bayesian inferences,” *Decision Support Systems*, vol. 125, p. 113100, 2019.

[4] R. K. Mobley, *An introduction to predictive maintenance*. Elsevier, 2002.

[5] “Predictive maintenance market.” https://www.marketsandmarkets.com/Market-Reports/operational-predictive-maintenance-market-8656856.html.

[6] M. S. K. Kopuru, S. Rahimi, and K. Baghaei, “Recent approaches in prognostics: State of the art,” in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pp. 358–365, The Steering Committee of The World Congress in Computer Science …, 2019.

[7] Z. Li, K. Goebel, and D. Wu, “Degradation modeling and remaining useful life prediction of aircraft engines using ensemble learning,” *Journal of Eng. for Gas Turbines and Power*, vol. 141, no. 4, 2019.

[8] A. Saxena, K. Goebel, D. Simon, and N. Eklund, “Damage propagation modeling for aircraft engine run-to-failure simulation,” in *IEEE ICPHM*, pp. 1–9, IEEE, 2008.

[9] E. Ramasso and A. Saxena, “Performance benchmarking and analysis of prognostic methods for cmapss datasets.,” 2014.

[10] T. Tinga and R. Loendersloot, “Physical model-based prognostics and health monitoring to enable predictive maintenance,” in *Predictive Maintenance in Dynamic Systems*, pp. 313–353, Springer, 2019.

[11] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, “Long short-term memory network for remaining useful life estimation,” in *IEEE ICPHM*, pp. 88–95, IEEE, 2017.

[12] “Using predictive technologies for asset maintenance.” https://www2.deloitte.com/us/en/insights/focus/industry-4-0/using-predictive-technologies-for-asset-maintenance.html.

[13] J. Wang, G. Wen, S. Yang, and Y. Liu, “Remaining useful life estimation in prognostics using deep bidirectional lstm neural network,” in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pp. 1037–1042, IEEE, 2018.

[14] X. Li, Q. Ding, and J.-Q. Sun, “Remaining useful life estimation in prognostics using deep convolution neural networks,” *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[15] A. Al-Dulaimi, S. Zabihi, A. Asif, and A. Mohammadi, “Hybrid deep neural network model for remaining useful life estimation,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3872–3876, IEEE, 2019.

[16] O. Güngör, B. Akşanlı, and R. Aydoğan, “Algorithm selection and combining multiple learners for residential energy prediction,” *Future Generation Computer Systems*, vol. 99, pp. 391–400, 2019.

[17] O. Sagi and L. Rokach, “Ensemble learning: A survey,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[18] J. Shi, T. Yu, K. Goebel, and D. Wu, “Remaining useful life prediction of bearings using ensemble learning: The impact of diversity in base learners and features,” *Journal of Computing and Information Science in Engineering*, pp. 1–35, 2020.

[19] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, “Deep learning for smart manufacturing: Methods and applications,” *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018.

[20] P. Kowalski and R. Smyk, “Review and comparison of smoothing algorithms for one-dimensional data noise reduction,” in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pp. 277–281, IEEE, 2018.

[21] “pykalman.” https://pykalman.github.io.

[22] L. Jayasinghe, T. Samarasinghe, C. Yuen, J. C. N. Low, and S. S. Ge, “Temporal convolutional memory networks for remaining useful life estimation of industrial machinery,” *arXiv preprint:1810.05644*, 2018.

[23] A. Gensler, J. Henze, B. Sick, and N. Raabe, “Deep learning for solar power forecasting—an approach using autoencoder and lstm neural networks,” in *2016 IEEE international conference on systems, man, and cybernetics (SMC)*, pp. 002858–002865, IEEE, 2016.

[24] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[25] E. Ramasso, “Investigating computational geometry for failure prognostics.,” 2014.

[26] J. Wu, K. Hu, Y. Cheng, H. Zhu, X. Shao, and Y. Wang, “Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network,” *ISA transactions*, vol. 97, pp. 241–250, 2020.

[27] J. Lofberg, “Yalmip: A toolbox for modeling and optimization in matlab,” in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pp. 284–289, IEEE, 2004.

[28] M. ApS, *The MOSEK Optimization Toolbox for MATLAB manual. Version 9.2*, 2020.

[29] J. Li, X. Li, and D. He, “A directed acyclic graph network combined with cnn and lstm for remaining useful life prediction,” *IEEE Access*, vol. 7, pp. 75464–75475, 2019.