DOWELL: Diversity-induced Optimally Weighted EnsembLe Learner for Predictive Maintenance of Industrial Internet of Things Devices

Onat Gungor, Tajana S. Rosing, *Fellow, IEEE*, and Baris Aksanli, *Member, IEEE*

Abstract—**The Industrial Internet of Things (I-IoT) enables a smarter maintenance approach for various industrial applications, such as manufacturing, logistics, etc. This approach is based on continuously observing system data to predict device failures and increase device efficiency. This smart maintenance, also known as predictive maintenance (PDM), finds an optimal maintenance schedule to reduce operational and capital costs. Accurate remaining useful life (RUL) prediction is critical for an effective PDM system. Data-driven RUL estimation methods are quite popular owing to their easier 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. We propose an ensemble learning framework where accurate and diverse base learners are selected out of 20 different state-of-the-art deep learning models. For accuracy, we discover the optimal weights of base learners by constructing an optimization problem. For diversity, we measure the similarity among base learner predictions and iteratively select the most diversified set of models while keeping the accuracy at a certain level. We show that our approach can have 39.2% faster retraining compared to an accuracy-based ensemble with only 3.4% loss in accuracy.**

Index Terms—**Industrial internet of things (I-IoT), industry 4.0, predictive maintenance, remaining useful life prediction, deep learning, ensemble learning, mathematical optimization**

I. Introduction

INDUSTRY 4.0 or fourth industrial revolution is an important milestone in factories and production systems, where smart manufacturing has become an essential component. This leverages the adaptation of traditional Internet of Things (IoT) for industrial manufacturing, creating the Industrial IoT (I-IoT) notion. The I-IoT enables the interconnection of anything (e.g. sensors, actuators), anywhere, and at any time in the context of industrial applications [1]. This allows better automation, higher reliability, and more fine-grained control, utilizing computer networks to collect enormous amount of data from the connected machines and convert this data into actionable information [2]. To optimize the performance of these smart and automated systems, timely and correct maintenance decisions are necessary, eliminating unplanned downtime, and better inventory management. The focus of I-IoT is the appropriate management of industrial assets along with predictive maintenance [3]. For example, BOSCH maintains quality management through predictive maintenance based on its use of big data analysis, smart sensors, and AI [4]. Predictive maintenance (PDM), or prognostics, predicts time-to-failure (i.e. lifetime) of a machine by employing copious mathematical approaches. PDM finds an optimum time to schedule a maintenance before any failure occurs. It has a burgeoning 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].

Remaining useful life (RUL) is defined as the remaining time of a machine to perform its functions until it fails. RUL prediction is an important PDM application [6], which can significantly improve the overall system performance and efficiency. Even the slightest improve in the RUL prediction performance can lead to enormous cost gains via increased production efficiency, reduced parts replacement costs, etc. Recently, data-driven RUL prediction methods have become popular with the abundance of available data. These machine learning (ML) oriented methods find a mapping between historical data and RUL. One robotic manufacturing company reduced downtime by 50% and increased performance by 25% by utilizing an ML algorithm for failure prediction [7]. Among these, deep learning (DL) methods are more preferable due to their superior performance. However, industrial systems consist of large number of individual systems and components, and finding a single method that works best in these various settings is a difficult task. Also, since PDM data might show extreme variations in early vs. late phases (e.g. machines do not fail early), these ML models may need retraining with the new incoming data. The retraining phase is the performance bottleneck of these models, with significant computation time overhead. To achieve the vision of I-IoT and utilizing from the huge data generated by I-IoT devices, there is a need for lightweight prediction approaches [3].

Approaches for big data analytics play a primary role in next-generation I-IoT systems [3]. Instead of a single method, ensemble learning combines multiple algorithms (i.e. base learners) and improves base learner performance. This results in more successful data analytics methodology, increasing system robustness and decreasing maintenance costs of I-IoT systems. Ensemble learning eliminates single method selection but may create additional overhead as multiple

Onat Gungor is with the Electrical and Computer Engineering Department, University of California San Diego, La Jolla, CA 92093 USA and with the Electrical and Computer Engineering Department, San Diego State University, San Diego, CA 92182 USA (e-mail: ogungor@ucsd.edu).

Tajana Rosing is with the Computer Science and Engineering Department, University of California San Diego, La Jolla, CA 92093 USA (e-mail: tajana@ucsd.edu).

Baris Aksanli is with the Electrical and Computer Engineering Department, San Diego State University, San Diego, CA 92182 USA (e-mail: baksanli@sdsu.edu).

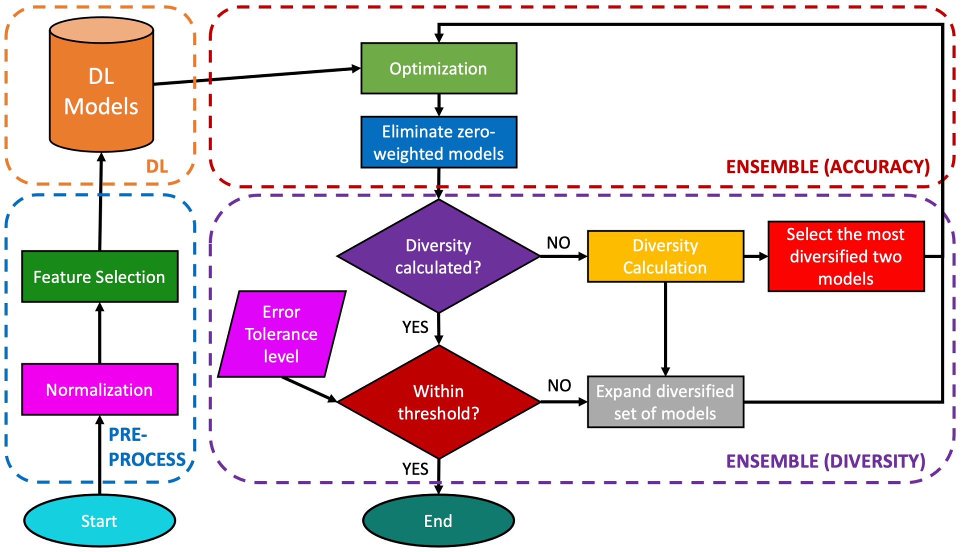


Fig. 1: Proposed Framework for RUL Prediction

methods need to be trained (and retrained with new data). Previous works generally use a small number of base learners, around 5-7, to avoid this issue [8]. However, there may be other methods performing better. Thus, the main issues with ensemble learning-based RUL prediction become 1) how many learners to consider for best performance and 2) how to select from them to minimize the retraining overhead.

To address these issues, we propose a diversity-induced optimally-weighted ensemble learner for RUL prediction (*OPTDIV*). To address the first issue, we dramatically increase the set of base learners to 20 methods, more than 2x compared to the state-of-the-art. We show single method performance on NASA C-MAPSS data set [9]. Some of these methods (e.g. layer normalized RNN, Wavenet) are used for RUL prediction for the first time. We observe that the best performing algorithm changes with data set. Next, our ensemble learner uses a quadratic programming model to find the base learner optimal weights. This optimizes the accuracy of our ensemble learner. Finally, we consider model diversity to reduce base learner set, by calculating the *Euclidean* distance among base learner predictions. This step selectively reduces the base learner set, reducing the computation overhead of the retraining phase. We compare our proposed model *OPTDIV* against optimally weighted ensemble learner *OPT* (without diversity) and see that *OPTDIV* has 39.2% faster retraining than *OPT* with only 3.4% loss in accuracy. This enables *OPTDIV* to adapt to changing conditions faster with high accuracy.

II. Related Work

Predictive Maintenance (PDM) uses predictive tools to determine when maintenance actions are necessary based on continuous monitoring [21]. Accurate RUL estimation is extremely beneficial to PDM. Accordingly, the research on improving RUL prediction performance gained popularity owing to advances in condition and health monitoring techniques. We can categorize the RUL prediction approaches under three main categories: experience based, model-driven (physical) and data-driven. Experience based models rely solely on expert knowledge and they are specific to a machine, that is why they are really hard to generalize. Physical models incorporate the physics of failure into RUL estimation [22]. The failure mechanism, e.g. fatigue, wear, is included in a mathematical model, establishing a relationship between the usage of a system to a lifetime prediction. It is arduous to create these models and they are machine specific.

Huge amounts of data are collected in I-IoT environments on a regular basis by connecting every machine and activity through network a sensors to the Internet [23]. Processing and analyzing this data plays a significant role in obtaining more effective I-IoT systems. Accordingly, data-driven models (the ML approach) use historical data to learn a model of system behavior [24]. ML methods play a vital role in I-IoT with use cases in quality control, maintenance cost optimization, and manufacturing process improvement. [25]. Since traditional ML models require extensive feature extraction, deep learning (DL) recently became more popular in order to provide end-to-end RUL prediction [26]. For instance, Hyundai Motors Co., a car manufacturer, recently developed a DL-based car fault diagnosis system that is superior to expert analysis [4]. It is also shown DL provides better prediction performance than traditional ML methods. There are numerous DL methods adopted for RUL prediction: recurrent neural network [27], convolutional neural network [13], long short-term memory [28], [29], auto-encoders [30], [31], and etc. There are also hybrid models which combine two or more models for better prediction such as CNN and LSTM [18].

Best performing algorithm may change across different systems (and thus data sets) [32]. An ensemble learner combines predictions from multiple models for more generalizable and robust models. In general, ensemble models perform better than the single methods. Thus, this brings more sophisticated predictive data analytics towards advanced and state-of-the-art I-IoT systems. To fulfill the theoretical requirements of good ensembles, base learners should be both accurate and diverse

TABLE I: Selected Deep Learning Models

Architecture	Category	Sub-category	Abbreviation	Explanation
Recurrent	Recurrent Neural Network(RNN) [10]	Vanilla	RNN	Simple RNN
		Layer Normalized	LNRNN	Normalization across the features dimension
	Long Short-Term Memory (LSTM) [11] Gated Recurrent Unit (GRU) [12]	Vanilla	LSTM GRU	Special memory cells
		Bi-directional	BLSTM BGRU	Backward direction (future and past data)
		Parallel	PLSTM PGRU	Two parallel paths
Convolutional	1-D Convolutional Neural Network (CNN) [13]–[16]	Vanilla	CNN	Simple 1D convolution
		Depthwise separable	DSEPCNN	Single input channel convolution
		Wavenet	WAVE	Dilated and causal convolutions, residual and parameterized skip connections
		Temporal Convolutional Networks	TCN	Dilated and causal convolutions
Hybrid	CNN-RNN [17]	Vanilla Parallel	CRNN PCRNN	Serially and in parallel connected CNN and RNN layers
	CNN-LSTM [18]	Vanilla Parallel	CLSTM PCLSTM	Serially and in parallel connected CNN and LSTM layers
	CNN-GRU [19]	Vanilla Parallel	CGRU PCGRU	Serially and in parallel connected CNN and GRU layers
	CNN-LSTM-GRU [20]	Parallel	CLG	In parallel connected CNN, LSTM, and GRU layers
	CNN-RNN-LSTM-GRU	Parallel	CRLG	In parallel connected CNN, RNN, LSTM, and GRU layers

[33]. In our work, we incorporate these two components into our ensemble learner by proposing an iterative framework. For RUL prediction, there are different ensemble learning approaches. Li et al. [34] combine multiple traditional ML-based learners by using particle swarm optimization and sequential quadratic programming to determine their weights. Shi et al. [35] utilize ensemble learning to predict RUL of bearings. Li et al. [36] assign an optimized, degradation-dependent weight to each learner to obtain better prediction accuracy. All these works do not reach great prediction performance since they did not use DL methods (i.e. they solely consider traditional ML methods). They also did not consider the diversity aspect of the base learners. In our work, we utilize from a variety of DL-based methods and improve their single prediction performance by proposing a diversity-induced ensemble learner framework.

III. Proposed Framework

Deep learning (DL) uses nonlinear processing layers to discover data representations. DL is particularly useful for I-IoT applications since it can model a complex production environment with high prediction accuracy. However, a single DL model may not provide accurate RUL predictions. Alternatively, distinctive models can be combined (ensemble learner) for better prediction. Similarly, it is onerous to choose the base learners. Mistakenly selected base learners may lead to deficient performance. To solve this problem, we create a diversity-induced optimally weighted ensemble learner. We present our framework in Fig. 1. First, we start with data pre-processing module where we normalize the data and select the most useful features. In DL module, we create a DL library where we consider 20 state-of-the-art models ranging from vanilla recurrent neural networks to temporal convolutional networks. In our last module, we create our ensemble model by following two steps: accuracy and diversity. In accuracy, we formulate an optimization problem to find the optimal weights of the base learners, then we eliminate the models which have zero weight. In diversity, we first create a diversity matrix to measure similarity among selected base learner predictions. Afterwards, we select the most diverse models and discover their optimal weights while keeping ensemble accuracy at a certain level by introducing error tolerance level.

A. Data pre-processing Module

We normalize the input sensor data using min-max normalization. For the feature selection, we utilize the Random Forest (RF) algorithm to discover variable importance which is calculated based on the reduction in residual sum of squares. Based on the calculated feature importance values, we keep the features that take positive importance values.

B. Deep Learning Module

Deep learning module outputs single prediction model RUL values using the input sensor data. We implement 20 DL prediction methods. We employ sliding time window approach to convert time series sensor data into a regression problem. The time window represents the number of past observations and we slide this window from the first observation to the last. Accordingly, 2-D input (selected features and time sequence of each feature) is provided to DL models. Table I presents the selected deep learning models. We categorize those models under three different architectures.

1) Recurrent Architectures: For recurrent models, hidden state can represent everything that has been seen so far. Recently, they become popular in RUL prediction and demonstrated good performance [28]. All models contain 3 recurrent layers (e.g. RNN, LSTM) having 64, 32, and 16 units. After recurrent layers, all models are connected to 2 fully connected feed forward neural networks (each with 8 units) and final 1-dimensional output layer.

2) Convolutional Architectures: Convolutional neural networks (CNN) use multiple feature extraction stages that can automatically learn hidden representations. Particularly, for RUL prediction, many CNN models has shown a great success [13]. Especially, 1-D CNN is common for time series applications, making it a suitable model for RUL prediction. Our CNN network structures contains five consecutive CNN layers, Flatten (Dropout) layer, one fully-connected layer (with 100 nodes) and an output layer with 1 node.

3) Hybrid Architectures: Hybrid models blend two or more methods to integrate strong aspects of different methods and to create more powerful estimator. For instance, combining CNN and LSTM can be useful since CNN can handle feature extraction, and LSTM can build long term time dependencies. Based on the previously selected vanilla recurrent and convolutional models, we construct hybrid models. We consider both in series and in parallel connection.

C. Ensemble Module

A theoretically good ensemble method should include both accurate and diverse base learners [33]. In our ensemble module, we satisfy this condition by considering accuracy and diversity steps consecutively. In accuracy, we find the optimal weights for our base learners by solving an optimization problem. Then we eliminate the zero-weighted models to keep the most important ones. In diversity, we measure similarity among remaining models based on *Euclidean* distance and select the smallest model subset that is able to meet the desired threshold criteria. The threshold level measures how much worse performance our diversified subset can tolerate than the optimally weighted ensemble learner.

1) Accuracy: The goal of the accuracy step is to discover the most important methods in RUL prediction. The importance is measured by the optimal weights assigned to the base learners. To find those weights, we formulate a mathematical optimization problem where we minimize the mean squared error (MSE). Mathematically, MSE is formulated using the variance and bias of an estimator $\hat{\theta}$:

$$MSE(\hat{\theta}) = Variance(\hat{\theta}) + Bias^2(\hat{\theta}) \tag{1}$$

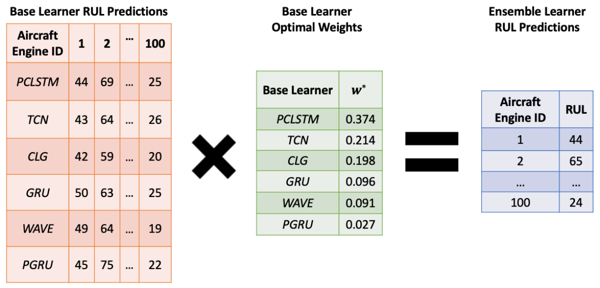
We specifically consider MSE since minimization of MSE is equivalent to minimizing bias and variance simultaneously which is shown in Equation 1. Accordingly, we formulate the following mathematical optimization model:

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

$$\text{subject to} \quad \sum_{j=1}^{\mathcal{M}} w_j = 1 \tag{3}$$

$$w_j \geq 0 \quad \forall j = 1, \ldots, \mathcal{M} \tag{4}$$

where $\mathcal{N}$ 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 (2) minimizes the MSE, constraint (3) ensures that weights sum up to 1, and constraints (4) restrict all weights to be non-negative. The constructed model has a convex quadratic objective function. We prove this by showing that Hessian (i.e. matrix that organizes all the second partial derivatives of a function) of our objective function is positive semidefinite


Base Learner RUL Predictions

Aircraft Engine ID	1	2	...	100
PCLSTM	44	69	...	25
TCN	43	64	...	26
CLG	42	59	...	20
GRU	50	63	...	25
WAVE	49	64	...	19
PGRU	45	75	...	22

Base Learner Optimal Weights

Base Learner	w^*
PCLSTM	0.374
TCN	0.214
CLG	0.198
GRU	0.096
WAVE	0.091
PGRU	0.027

Ensemble Learner RUL Predictions

Aircraft Engine ID	RUL
1	44
2	65
...	...
100	24

Fig. 2: Combination of base learner predictions with their optimal weights using a dot product operation

(PSD). Without loss of generality, objective function (2) can be reformulated using L2 norm and matrix notation:

$$\|y - \hat{Y}w\|_2^2 \tag{5}$$

where y is an $\mathcal{N}$ dimensional vector storing real values, $\hat{Y}$ is an $\mathcal{N} \times \mathcal{M}$ prediction matrix and w is an $\mathcal{M}$ dimensional weight vector. For simpler notation, let ψ be a function that maps w to $\|y - \hat{Y}w\|_2^2$:

$$\psi : w \mapsto \|y - \hat{Y}w\|_2^2 = \|y\|_2^2 - 2y^T\hat{Y}w + \|\hat{Y}w\|_2^2 \tag{6}$$

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

$$\frac{\partial \psi}{\partial w} = -2y^T\hat{Y} + 2w^T\hat{Y}^T\hat{Y} \tag{7}$$

$$\frac{\partial^2 \psi}{\partial w \partial w^T} = 2\hat{Y}^T\hat{Y} \tag{8}$$

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

$$\xi^T(\hat{Y}^T\hat{Y})\xi = (\hat{Y}\xi)^T(\hat{Y}\xi) = \|\hat{Y}\xi\|_2^2 \geq 0 \tag{9}$$

We proved that our objective function is convex and it is also quadratic. We also have affine constraints. Thus, above model is a quadratic program (QP) for which there is a guarantee that a local minimum is also the global minimum [37]. Solving this QP yields the optimal weights w^* for each DL model. Based on discovered w^*, we eliminate the models which have 0 weight. This means that unimportant models are eliminated from the larger set. Most importantly, we can achieve the optimal ensemble prediction performance using the base learner predictions and their corresponding weights. Fig. 2 demonstrates our approach to combine the base learner predictions with their optimal weights. Here, we perform a dot product operation which takes single algorithm RUL predictions and optimal weights, and outputs RUL predictions. To exemplify, consider the first row of the ensemble learner RUL predictions (blue table) in Fig. 2. In order to obtain the RUL prediction value as 44 (corresponding to the first aircraft engine), we multiply the first column of base learner RUL predictions (orange table) with the base learner optimal weights (green table) and round to the nearest integer. That is, $(44 \times 0.374) + (43 \times 0.214) + (42 \times 0.198) + (50 \times 0.096) + (49 \times 0.091) + (45 \times 0.027) \approx 44$. This optimization procedure considers solely accuracy of the base learners. We call this ensemble learner *OPT* for the rest of this paper and we denote the number of models in *OPT* as τ. We shrink the number of models by including diversity in our ensemble learner.

2) Diversity: There is a trade-off between accuracy and diversity in ensemble learners. Adding diversity leads to worse prediction performance and vice versa. Our main goal of adding diversity is to obtain the smallest subset of DL models while preserving the accuracy at a certain level. We call this diversity-aware optimally weighted ensemble learner *OPTDIV*. The main motivation behind adding diversity is the reduced retraining overhead. This overhead becomes crucial when new data arrives. In that case, there is a need to retrain the selected models. Since *OPTDIV* has smaller number of models, it would be faster to retrain than *OPT* approach.

To measure diversity, there is no generally accepted metric to be used [33]. In a regression problem, covariance between individual estimators' outputs can be used for instance. We choose to utilize *Minkowski* distance to measure the similarity among methods' predictions of *OPT*. *Minkowski* distance measures the similarity between two points in the normed vector space. Consider an $\mathcal{N}$ dimensional vector space and two points: $X = (x_1, x_2, ..., x_{\mathcal{N}})$ and $Y = (y_1, y_2, ..., y_{\mathcal{N}})$. *Minkowski* distance $D(X, Y)$ of order p is formulated as:

$$D(X,Y) = \left(\sum_{i=1}^{\mathcal{N}} |x_i - y_i|^p \right)^{1/p} \tag{10}$$

When $p = 2$, *Euclidean* distance is obtained:

$$\sqrt{\sum_{i=1}^{\mathcal{N}} (x_i - y_i)^2} \tag{11}$$

Here, any p value can be used without loss of generality. We calculate Euclidean distance among all possible pair of models and create a distance matrix $\Xi_{\tau \times \tau}$. This matrix demonstrates diversity among selected model predictions. For instance, $\Xi_{(TCN,GRU)}$ denotes the distance between prediction vectors of temporal convolutional network and vanilla GRU. If the distance is bigger, we expect those two models to be more diverse (since their predictions are less similar).

First, we create the smallest subset of models by selecting the two most diverse models (*OPTDIV*). This is achieved by finding the maximum value in Ξ and its corresponding models. Afterwards, we solve previously formulated QP to calculate the optimal weights for those base learners and measure prediction performance. At this point, we check whether *OPTDIV* prediction performance is within desired threshold. We determine the threshold based on error tolerance level (δ) which is a parameter in [0,1]. Based on selected δ, we determine allowable performance difference between *OPT* and *OPTDIV*. To put more concretely, assume that in *OPT*, we found optimal ensemble RMSE to be 20. If we select error tolerance level δ to be 1%, then our proposed approach iterates until we reach an RMSE of 20.2 or less. At each iteration, we expand *OPTDIV* by selecting the next most diverse model (i.e. select the model that has the second highest element in Ξ and so on). Overall, our approach starts by selecting the two most diverse models, and increases the number of models until it meets the desired threshold level.

Let υ denote the number of models in *OPTDIV*. Observe that for any δ, we have $\upsilon \leq \tau$. Without loss of generality, we can assume that this inequality is strict. This assumption reveals the trade-off between computational overhead and accuracy. Since *OPTDIV* has smaller number of models, when the new data comes, retraining will be less costly than *OPT*. However, its accuracy would be worse (but not worse than δ). If we need much faster training and sub-optimal accuracy, *OPTDIV* would be a better choice.

IV. Experimental Analysis

A. *Experimental Setup*

Dataset Description: NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) is a commonly used data set for RUL prediction. The simplified engine diagram in Fig. 3 demonstrates the major components (e.g. fan, combustor) of an aircraft engine [9]. Input data is created by different sensors (e.g. temperature, pressure) placed on these components. NASA C-MAPSS has four sets of data: FD001∼FD004. Each data has different complexity levels (e.g. operating conditions, fault conditions) as shown in Table II. While FD001 is the simplest data set, FD004 is the most complicated (i.e. the highest number of operating and fault conditions) one. Each data set has separate training and testing sets. The training data contains the entire lifetime of an engine, yet test data is terminated at some point before engine failure. Each row corresponds to a single operating time cycle with 26 columns: the engine ID, cycle index, three operational settings, and 21 sensor measurements. At the beginning, the engine is operating normally and develops a fault at some point in the future. The real RUL values are provided for the test data, and the goal is to estimate the RUL before a failure occurs. RUL prediction of aircraft engines is crucial in aviation industry. For instance, Rolls Royce, an aircraft engine manufacturer, recently began using big data (collected from an I-IoT environment) to maintain aircraft engines. This company prevents flight delays through predictive maintenance of their aircraft engines [4].

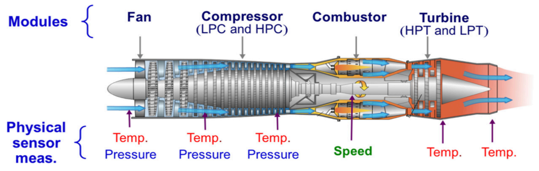


Fig. 3: Engine Diagram Simulated in C-MAPSS [9]

TABLE II: 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: The easiest and the most common approach to model RUL is linear where its value decreases

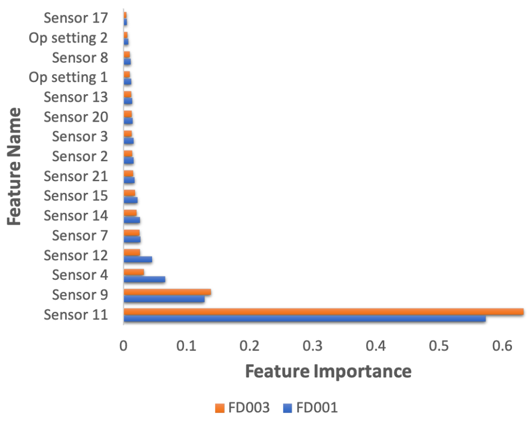


Fig. 4: Random Forest Feature Importance

with time. Nonetheless, it is hard to observe an apparent degradation behavior of an engine in the beginning of its lifetime. In general, an engine degrades more as it approaches its end of life. For the C-MAPSS data set, it is shown that piece-wise linear degradation model is more appropriate than linear model [38]. Thus, we adopt a piece-wise linear RUL target function where we set the maximum RUL limit constant (the break point) to 125-time cycles as in [39].

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:

$$RMSE = \sqrt{\frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \epsilon_i^2} \tag{12}$$

B. Data pre-processing

We select the most important features in predicting RUL for (1) better prediction performance, (2) decreased computational complexity, and (3) preventing overfitting. We perform feature selection for FD001 and FD003 because specifically these two demonstrate explicit health degradation behaviors. We use Random Forest (RF) algorithm to find the feature importance. Fig. 4 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. It is important to note that this process is fully automatized meaning that we do not need to visually inspect coming data. When the new data comes, RF will update itself to calculate new feature importance values. Based on that, the most useful features will be updated.

C. Deep Learning Models Performance

All experiments are run on a PC with 16 GB RAM and an 8-core 2.3 GHz Intel Core i9 processor. 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).

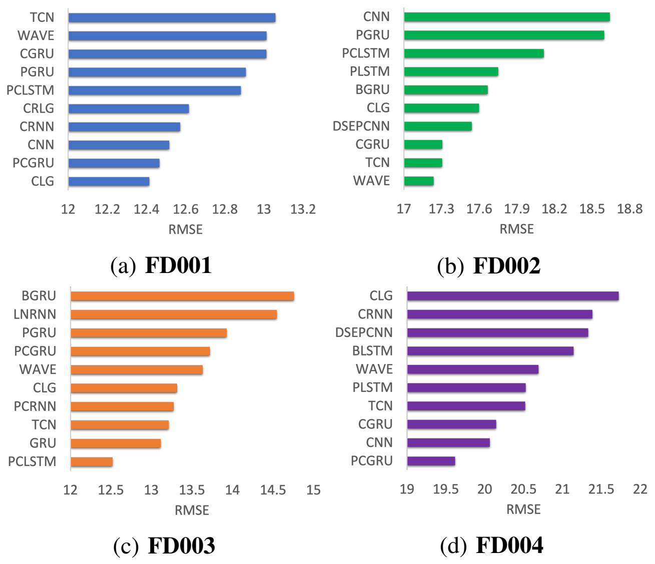


Fig. 5: Deep Learning Models Prediction Performace

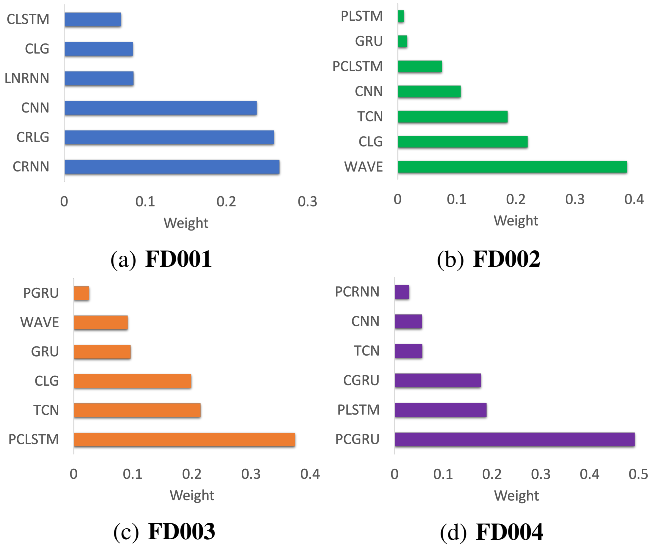


Fig. 6: Base Learner Optimal Weights

We replicate each experiment 10 times and report average values. We adopt a sliding time window approach for better prediction performance. Here, we select the time window based on the minimum number of cycles for the test data since only sequences that meet the window-length is possible to consider. To use the entire test data, the window size is selected as 30, 20, 35, and 15 respectively from FD001 to FD004. Fig. 5 depicts the best performing 10 models across the data sets. At each sub-figure, horizontal axis represents RMSE value, and vertical axis provides specific DL model. The smaller the RMSE value, the better the model is. One important observation is that best algorithm varies depending on data set. Specifically, CLG, WAVE, PCLSTM, and PCGRU are the best algorithms with RMSE values 12.42, 17.24, 12.52, and 19.62 respectively. We can also observe that hybrid and convolutional models perform better than recurrent ones.

D. Ensemble Learner Performance

Accuracy: To discover the most important models, we formulate and solve the previously formulated quadratic optimization problem in YALMIP [40] using the solver MOSEK [41]. As an output, we obtain the optimal weights w^* corresponding to each base learner. In Fig. 6, we share the models which take non-zero weights. In each sub-figure, on the x-axis, we have optimal weights w^*, and on the y-axis we have corresponding DL models. CRNN, WAVE, PCLSTM, and PCGRU takes the greatest weights with 0.27, 0.39, 0.37, and 0.49 respectively. The number of models that have positive weights is different at FD002 where we have 6 models ($\tau = 6$) as opposed to 5 ($\tau = 5$) as in the remaining data sets. We also examine the relationship between base learner RMSEs and their optimal weights. Intuitively, we expect the base learner with the smallest RMSE to take the largest weight. However, this cannot be observed at FD001. Besides, even though some methods do not perform really well by themselves, they can still take a positive weight, e.g. PCRNN at FD004. In terms of optimal weights, we again cannot see one algorithm to be the best across all data sets (i.e. no dominant algorithm). When we multiply w^* with the corresponding learner predictions, we obtain *OPT* predictions. RMSE values of *OPT* are 11.95, 16.08, 12.04, and 19.17 respectively. Compared to the best single estimators, *OPT* brings 3.7%, 6.7%, 3.9%, and 2.3% accuracy improvement. Note that *OPT* has the most accurate predictions since we solve an optimization problem. Adding diversity degrades the estimation accuracy (trade-off between accuracy and diversity), yet it decreases the number of models which leads to lower computational overhead.

Comparison with State-of-the-Art Ensemble Methods: To verify the performance of our ensemble approach (*OPT*), we compare it with the state-of-the-art ensemble methods where we select Bagging, Gradient Boosting, AdaBoost, and Stacking [42]. We select these methods since these are the most well-known and successful ensemble approaches. As an input, 20 single method predictions are provided to these ensemble methods. We implement these methods using scikit-learn library in Python [43]. For the optimal hyper-parameter selection, we utilize GridSearch which builds model on each parameter combination possible and finds the best hyper-parameter configuration [44]. Accordingly, we use the following hyper-parameters: AdaBoost $\mapsto$ number of estimators: 30, learning rate: 1, loss: linear; Gradient Boosting $\mapsto$ number of estimators: 90, learning rate: 0.1, loss: least squares; Bagging $\mapsto$ number of estimators: 20, max samples: 2, max features: 1; Stacking $\mapsto$ averaging the predictions of all base learners. We make a prediction performance comparison based on test data error. Fig. 7 shows this comparative analysis where our method *OPT* (represented with green color) outperforms selected ensemble methods in all data sets. Specifically, *OPT* improves the prediction performance of the best ensemble method by up to 7.3% and 4.6% on average.

Diversity: We create the diversity matrix $\Xi_{\tau\times\tau}$ based on Euclidean distance between *OPT* model predictions. For demonstration purposes, we share a sub-matrix of $\Xi_{3\times 3}$:

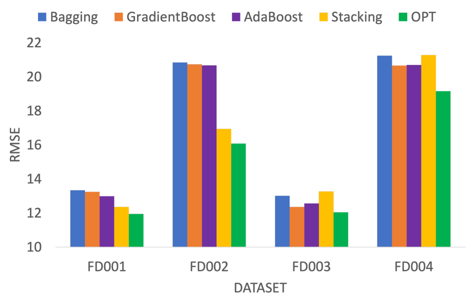
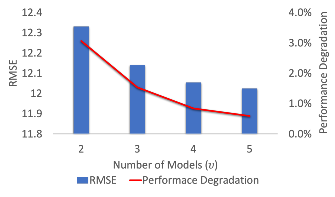
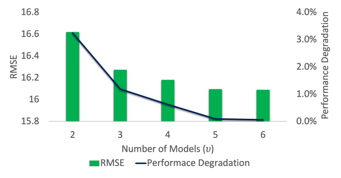


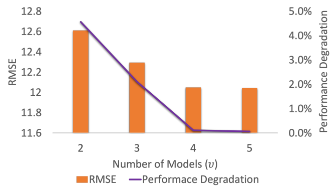
Fig. 7: State-of-the-art Ensemble Comparison



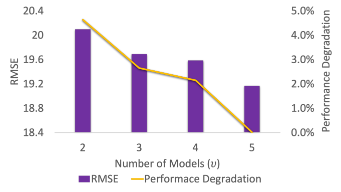
(a) **FD001**



(b) **FD002**



(c) **FD003**



(d) **FD004**

Fig. 8: *OPTDIV* Performance for Different υ

$$\Xi_{3\times 3} = \begin{array}{c} \\ \\ \\ \end{array}\begin{array}{ccc} CRNN & CRLG & CNN \\ \left[\begin{array}{c} 0 \\ 7 \\ 3 \end{array}\right. & \begin{array}{c} 7 \\ 0 \\ 1 \end{array} & \left.\begin{array}{c} 3 \\ 1 \\ 0 \end{array}\right] \end{array}\begin{array}{c} \\ CRNN \\ CRLG \\ CNN \end{array}$$

where we have a symmetric matrix with zero diagonal. Here, each cell represents a normalized diversity score. For instance, the most two diverse models are CRNN, and CRLG since the distance between those two points is the maximum. We start with the 2 most diverse models and expand this set until we reach desired error tolerance level δ. To provide better understanding, we consider all the models in *OPT* and create *OPTDIV* until $\upsilon = \tau - 1$. To exemplify, for FD001, we have $\tau = 6$, so we consider the most diverse models until $\upsilon = 5$. For each υ value, we calculate its RMSE and its performance degradation compared to *OPT*. Fig. 8 demonstrates *OPTDIV* performance under varying υ values.

At each sub-figure, on the x-axis, we have υ values, on the y-axis we depict two different measurements: on the left we have RMSE values and on the right we have performance degradation measurements. Performance degradation is calculated by measuring how much accuracy difference there are between *OPTDIV* and *OPT*. For example, when we only use the two most diverse models in FD001, *OPTDIV* is 3.1% less accurate than *OPT*. Naturally, the more models we add, the more accurate *OPTDIV* becomes. However, at some instances increasing υ does not bring significant advantage, e.g. in

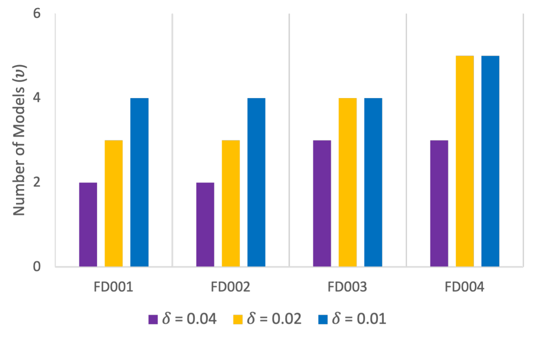


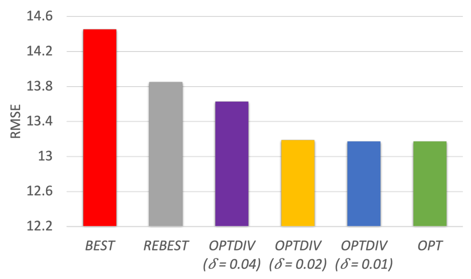
Fig. 9: Effect of δ on υ

FD003, going from $\upsilon = 4$ to $\upsilon = 5$ increases accuracy only by 0.05%. Hence, we bring the idea of error tolerance level δ to select the smallest model subset for *OPTDIV* while not losing a significant accuracy. To exemplify, given $\delta = 0.01$, *OPTDIV* has 4 models (υ = 4) at all data sets except FD004. If we select a bigger δ, then we end up with smaller υ values and vice versa. Fig. 9 demonstrates this relationship. This figure shows selected υ values based on varying $\delta \in$ [0.04, 0.02, 0.01]. In general, we can observe that as δ value gets larger, υ values become smaller. It means that *OPTDIV* has to select more models to obtain desired accuracy level. As we observed previously, adding more models may not bring a significant advantage at some cases.

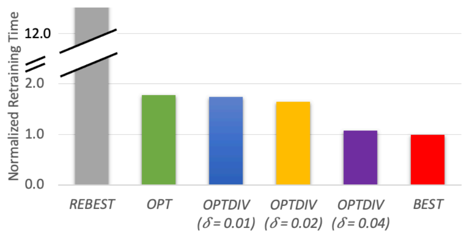
E. Analysis on Retraining

This section analyzes the retraining performance and overhead of *OPT*, *OPTDIV*, and single ML methods when there is new data and retraining is required. For *OPT* and *OPTDIV*, we retrain only τ and υ number of models respectively. For single ML method, we have two approaches: *BEST* and *REBEST*. Both methods use a single ML method but the former uses the previously-identified best method, whereas *REBEST* retrains all methods to find whether the best method has changed with new incoming data. *BEST* eliminates the high retraining overhead, but has the risk to perform worse with new data. *REBEST* can update the best single model but with a much greater computation cost. We compare *OPT*, *OPTDIV*, *BEST*, and *REBEST* in terms of retraining time and prediction accuracy. To make that comparison, we use trained models of FD001, and treat FD003 as newly arrived data. We retrain the selected methods (pre-trained on FD001) based on FD003 train data, and test performance using FD003 test data. This represents that newly coming data can be different from the old data, i.e. FD003 has 2× fault conditions than FD001. Fig. 10 shows the results for this analysis: the top figure has RMSE values and the bottom figure compares retraining time, normalized against *BEST*. We consider three *OPTDIV* configurations with varying δ values.

Accuracy comparison: When we compare *OPT* and *OPTDIV* with $\delta = 0.01$, we observe that they perform equally. However, this was not the case on FD001 training where we observed 0.84% accuracy difference (see Fig. 8). This means that performance difference has disappeared when we retrain



(a) **RMSE**



(b) **Normalized Retraining Time**

Fig. 10: Retraining Approach Comparison

these two approaches. Similarly, there is a 0.1% difference between *OPTDIV* ($\delta = 0.02$) and *OPT* which was previously 1.54%. This proves that adding diversity leads to better generalization, which is helpful when the data change. We should also note that *OPTDIV* with any δ value outperforms *BEST* and *REBEST* approaches significantly.

Overhead comparison: Between *OPT* and *OPTDIV*, retraining overhead difference becomes apparent when we compare *OPTDIV* ($\delta = 0.04$), and *OPT* approaches. As *OPTDIV*'s error tolerance increases, it becomes much faster than *OPT*. *OPTDIV* ($\delta = 0.04$) is 39.2% faster with only a 3.4% loss in accuracy. We also discover that *OPTDIV* is 92% faster than *REBEST*. Compared to *BEST*, *OPTDIV* brings 5.7% performance improvement while only needing 8.5% slower retraining. In summary, *OPTDIV* can adapt to changing data much faster, with small loss in accuracy.

V. Conclusions and Future Work

One of the major objectives of I-IoT is the better management of industrial assets through their predictive maintenance [3]. Accordingly, predictive maintenance (PDM) systems can utilize accurate remaining useful life (RUL) prediction to obtain the best performance from a production system. Deep learning based data-driven RUL prediction methods become prevalent due to their high accuracy and easy implementation. However, choosing one algorithm may not provide accurate predictions across different settings and data sets. Hence, we propose diversity included accurate ensemble learner. We discover the smallest subset of learners out of 20 different DL methods while keeping accuracy at a certain level. We include accuracy by finding optimal weights of the base learners. We then measure the similarity among base learner predictions and select the most diversified set of models. We show that our approach can have 39.2% faster retraining compared to an accuracy-based ensemble learner with only 3.4% loss in accuracy. In the future, we are planning to improve our method considering the real-time aspects of data collection, where the base learners, their optimal weights, and diversity calculations are evaluated and updated as new data arrive.

Acknowledgement

This work has been funded in part by NSF, with award numbers #1527034, #1730158, #1826967, #1830331, #1911095, and #1952225.

References

[1] H. Xu *et al.*, "A survey on industrial internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[2] R. Zhao *et al.*, "Deep learning and its applications to machine health monitoring: A survey," *arXiv preprint arXiv:1612.07640*, 2016.

[3] W. Z. Khan *et al.*, "Industrial internet of things: Recent advances, enabling technologies and open challenges," *Computers & Electrical Engineering*, vol. 81, p. 106522, 2020.

[4] S. M. Lee, D. Lee, and Y. S. Kim, "The quality management ecosystem for predictive maintenance in the industry 4.0 era," *International Journal of Quality Innovation*, vol. 5, no. 1, pp. 1–11, 2019.

[5] "Pdm 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 *ICAI*, pp. 358–365, 2019.

[7] H. J. Wilson *et al.*, "How companies are using machine learning to get faster and more efficient," *Harvard Business Review*, 2016.

[8] O. Gungor, J. Garnier, T. S. Rosing, and B. Aksanli, "Lenard: Lightweight ensemble learner for medium-term electricity consumption prediction," in *IEEE SmartGridComm*, pp. 1–6, IEEE, 2020.

[9] A. Saxena *et al.*, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *IEEE ICPHM*, pp. 1–9, IEEE, 2008.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[11] K. Greff *et al.*, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232.

[12] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint*, 2014.

[13] 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.

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE CVPR*, pp. 1251–1258, 2017.

[15] A. v. d. Oord *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[16] 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.

[17] X. Zhang *et al.*, "Remaining useful life estimation based on a new convolutional and recurrent neural network," in *IEEE CASE*, pp. 317–322, IEEE, 2019.

[18] A. Al-Dulaimi *et al.*, "Hybrid deep neural network model for remaining useful life estimation," in *IEEE ICASSP*, pp. 3872–3876, IEEE, 2019.

[19] K. Lee *et al.*, "Cnn and gru combination scheme for bearing anomaly detection in rotating machinery health monitoring," in *IEEE ICKII*, pp. 102–105, IEEE, 2018.

[20] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Multipath parallel hybrid deep neural networks framework for remaining useful life estimation," in *IEEE ICPHM*, pp. 1–7, IEEE, 2020.

[21] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019.

[22] 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.

[23] T. Kotsiopoulos, P. Sarigiannidis, D. Ioannidis, and D. Tzovaras, "Machine learning and deep learning in smart manufacturing: The smart grid paradigm," *Computer Science Review*, vol. 40, p. 100341, 2021.

[24] M. Paolanti *et al.*, "Machine learning approach for predictive maintenance in industry 4.0," in *2018 14th IEEE/ASME MESA*, pp. 1–6.

[25] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using iot sensor data," in *2017 ICSESS*, pp. 87–90, IEEE, 2017.

[26] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: a survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[27] L. Guo *et al.*, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98–109, 2017.

[28] S. Zheng *et al.*, "Long short-term memory network for remaining useful life estimation," in *IEEE ICPHM*, pp. 88–95, IEEE, 2017.

[29] D. Bruneo and F. De Vita, "On the use of lstm networks for predictive maintenance in smart industries," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 241–248, IEEE, 2019.

[30] X. Bampoula, G. Siaterlis, N. Nikolakis, and K. Alexopoulos, "A deep learning model for predictive maintenance in cyber-physical production systems using lstm autoencoders," *Sensors*, vol. 21, no. 3, p. 972, 2021.

[31] A. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional lstm neural network autoencoders," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6069–6078.

[32] 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.

[33] S. Gu, R. Cheng, and Y. Jin, "Multi-objective ensemble generation," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 234–245, 2015.

[34] 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.

[35] J. Shi *et al.*, "Remaining useful life prediction of bearings using ensemble learning," *Journal of Computing and Information Science in Engineering*, pp. 1–35, 2020.

[36] Z. Li *et al.*, "An ensemble learning-based prognostic approach with degradation-dependent weights for remaining useful life prediction," *Reliability Engineering & System Safety*, vol. 184, pp. 110–122, 2019.

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

[38] E. Ramasso, "Investigating computational geometry for failure prognostics.," 2014.

[39] J. Wang *et al.*, "Remaining useful life estimation in prognostics using deep bidirectional lstm neural network," in *IEEE PHM*, pp. 1037–1042, IEEE, 2018.

[40] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 ICRA*, pp. 284–289, IEEE, 2004.

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

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

[43] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] S. M. LaValle *et al.*, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.