Novel RUL Prediction of Assets based on the Integration of Auto-Regressive Models and an RUSBoost Classifier

Georgios Fagogenis[1], David Flynn[2], David Lane[1]

{gf63, D.Flynn, D.M.Lane}@hw.ac.uk

***Abstract*— This paper presents a novel, data-driven algorithm for the computation of the Remaining Useful Life (RUL) of an asset. The algorithm utilizes the asset's state history to learn a prognostic model from data. The prognostic model comprises an ensemble of Auto-Regressive (AR) models, together with a state-of-the-art classifier. The AR part of the algorithm is used to predict the system's state evolution. The classifier discriminates between healthy and faulty operation, given the asset's current state. The predicted state, as computed by the AR model, is fed to the classifier. The first time when the predicted state is classified as faulty is returned as the RUL of the system. The resulting prognostic algorithm was tested on the CMAPSS dataset as provided from NASA Ames Research Center. Cases of unknown future input trajectory as well as cases with multiple faults have been investigated.**

I. Introduction

Prognostics pertain to the computation of the Remaining Useful Life (RUL) of an asset. The RUL is defined as the time interval during which the asset's performance satisfies certain criteria. By understanding the factors that are relevant to the RUL, the operation and maintenance of the asset can be optimitised. Furthermore, the insight provided by the prognostic models informs Design for Reliability (DFR) for next generation assets.

Data-driven methods have recently become popular for prognostic applications. This is a result of the abundance of historical data from modern engineering systems. Moreover, recent advances in machine learning provide the means of interpreting these data effectively. There are two important issues, however, that need to be addressed to enable successful prognoses. Firstly, the data that are available to the engineers are not always suitable for training machine learning algorithms. Assets are often stopped before total failure. Consequently, the available datasets are censored; the full trajectory of the state of the asset is not recorded until failure. Even in the case of uncensored data, however, the majority of the data samples belong to the healthy region of operation. This skewness in the data hinders the training of the algorithms. Another common problem is related to the asset's variance in dynamics. Different operating conditions, as well as manufacturing uncertainty are two possible explanations for that. The need for a "smart" model, which is robust to the asset's dynamic versatility, is prominent.

[1] Ocean Systems Laboratory (OSL), School of Engineering and Physical Sciences, Earl Mountbatten Building, Heriot-Watt University

[2] MISEC,School of Engineering and Physical Sciences, Earl Mountbatten Building, Heriot-Watt University

In this paper, a novel algorithm is presented, which addresses both the problems of data skewness and dynamic variance. It employs a state-of-the-art classifier, together with an adaptive autoregressive predictive model. Random Undersampling Boosting (RUSBoost) [1], is used to classify the state of the system as healthy or faulty; hence acting as a fault detector. RUSBoost addresses the skewness of the dataset using random undersampling of the over-represented class. Moreover, RUSBoost can be applied without loss of performance on multi-class problems. This is particularly useful in the presence of multiple types of faults. The dynamic model is based on the Locally Weighted Projection Regression (LWPR) [2], a regression algorithm for non-linear functions with high dimensional input. LWPR is accurate and stable, when trained on sufficiently large datasets (more than 2000 samples). The parameters of the LWPR are updated in real time, as more data become available. The incremental update of the parameters ascribes adaptivity to LWPR. Hence, the dynamic model compensates for changes in the asset's dynamics. The asset's dynamic model is utilised to forecast the evolution of the asset's state. The predicted state trajectory, as computed by the dynamical model, is fed to the classifier. The classifier, in turn, infers the health state of the asset. The first time instant at which the state is classified as faulty is returned by the algorithm as the remaining useful life of the asset. The algorithm was tested on two datasets, as provided from NASA's Ames Research Center [3], [4].

The structure of this brief is as follows: Section II presents the current state of the art in data-driven prognostics. Next, the LWPR method, as used for the adaptive autoregression, and the RUSBoost classifier are described. The combination of the latter, to predict the remaining useful life, is analyzed in section V. The results of the algorithm, applied to a prognostic competition dataset, are given in section VI. Finally, the paper concludes with a brief discussion on the results and on possible extensions of the algorithm.

II. Relevant Work

Various data-driven methods have been reported in the prognostic literature for the estimation of the remaining useful life. Auto-Regressive Moving Average (ARMA) and Exponential Projection methods [5] are amongst the most common techniques. [6] employed an ARMA model to compute the RUL of bearings. Random processes have been used to tackle the problem of varying operating conditions and manufacturing variance. Random Coefficient Regression,

originally proposed by [7], utilizes a stochastic model, that consists of the actual degradation level combined with a corrective term. The corrective term is, in fact, a normal distribution with zero mean and constant standard deviation. Similarly, other techniques are based on stochastic processes (Wiener Process, Gamma Process) for forecasting. Gebraeel et al [8] developed a random coefficient regression method based on the Wiener Process (Standard Brownian Motion). The parameters of the method are updated online in a Bayesian fashion.

Another tool, often found in data-driven prognostic applications, is the Kalman Filter. The Kalman Filter is used to forecast the mean value of a set of fault precursor variables. To this end, a dynamic model is employed to predict the evolution of the precursor variables. The parameters of the model are identified using historical data. The choice of the dynamical model influences the accuracy of the results significantly. The state vector of the Kalman Filter is often augmented to include the parameters of the model. In this way the filter can adapt to changes in the environment [9]. In [10], the Kalman Filter is utilized to predict the evolution of a crack in a tensioned steel band. A generalization of the Kalman Filter, namely the Particle Filter, is used in cases where the noise statistics are not Gaussian. Particle Filters parametrize the relevant distributions with a population of samples (particles) [11]. Consequently, they can approximate arbitrarily complex distributions.

Clustering pertains to the study of grouping objects based on feature similarity. Clustering algorithms maximize feature similarity within the groups. Additionally, inter-group diversity is also maximized. In the case of fault detection, the objects are the vectors that hold the sensor data and the clusters are associated with potential faults. One of the most common clustering algorithms is K-Means. The feature similarity amongst objects is quantified using some sort of 'distance' metric. In particular, distance metrics for fault detection can be found in [12]. Moreover, Support-Vector-Machines [13] and their kernel-based extensions [14] are used to optimize the boundaries between the clusters.

Several prognostic algorithms leverage Hidden Markov Models (HMMs) to predict the remaining useful life of an asset. Approaches based on HMMs are documented in [15], [16]. One drawback of HMMs arises from the fact that the duration of each state is not modelled explicitly. Therefore, staying at a certain state follows a geometrical distribution. This assumption can often present practical applications with shortcomings, as it is too rudimentary. As a remedy, Hidden Semi-Markov Models (HSMM) [17] have been developed. In a Hidden Semi-Markov Model, the state duration is also treated as a random variable; it is drawn from a probability distribution. The parameters of such a distribution are learned during the training phase. HSMM have been shown to deliver better prognostic accuracy [18].

Finally, another extension of Hidden Markov Models uses *Mixtures of Gaussians* for representing the emission probabilities of the observation sequence, in the case of continuous variables. A prognostic algorithm subject to this concept has been documented by [19].

Similar to our work, [20] uses an ensemble of regression models together with a classifier to compute the Remaining Useful Life. The two methods belong to the same prognostic regime, albeit different algorithms have been employed for each of the modules of the prognostic system.

III. Adaptive Autoregression

The prognostic accuracy depends on the reliable prediction of the system's state. Moreover, the system's model needs to be adaptive, to account for variance in dynamic parameters, as well as for diverse operating conditions. To accomplish that, a modern incremental learning method, namely the Locally Weighted Projection Regression (LWPR) [2], has been used. In the rest of this section, the adaptive autoregression part of the prognostic approach is described.

A. Linear Weighted Projection Regression

Locally Weighted Projection Regression (LWPR) [2] is a state-of-the-art tool for non-linear function approximation. It combines dimensionality reduction together with locally weighted regression. The domain of the target function (i.e. the space of explanatory variables) is partitioned using a set of *Receptive Fields*. A receptive field is represented by a kernel. Each receptive field defines a neighborhood around a point. Furthermore, the kernel assigns weights to the points in that region. Following that, Partial Least Squares (PLS) [21] is applied on every "neighborhood", as defined by the kernels. In this way, the dimensions of the input space that best explain the variance of the dependent variables are identified. The local dimensionality reduction renders LWPR suitable for problems with high input dimension. After the dimensionality reduction, the locally applied PLS computes the parameters of the hyperplane, which best fit the data. Each hyperplane, as such, is a local approximation of the non-linear target function. Given a new query point, all local models compute a prediction based on the local hyperplane, that has been fitted during training. All these predictions are summed up, using the weights from the receptive fields, to compute the final prediction of the target function. In this manner every local model contributes only a fraction to the final estimation. The contribution of each model is regulated by the respective receptive field. Incoming data samples trigger the updating rules for both the local dimensionality reduction computation, as well the local regression. The incremental updating of the parameters of the model ascribe to LWPR adaptivity.

The training of the algorithm requires a certain set of parameters to be tuned by the user. The most important parameters are listed below:

- $init\_D$: initial size of the Gaussian kernel
- $init\_alpha$: learning rate for the optimization of the receptive field parameters
- $update\_D$: switches the online adaptation of the receptice fields on/off
- $penalty$: this parameter is used to prevent the algorithm from converging to indefinetely small receptive fields (this is actually the term γ in Equation 1).

The parameter $init\_D$ is proportional to the inverse of the Gaussian kernel's variance. Small values of $init\_D$ result to wider receptive fields. Depending on the target function, this may lead to poor performance of the model; approximating a non-linear function with scant local linear models. On the contrary, an excessive value for this parameter favors small receptive fields. This may cause overfitting problems. The size of the receptive fields is adapted on-line, if the parameter $update\_D$ is set to 'true'. In that case, the parameters of the receptive fields are computed by optimizing the following objective function:

$$J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i (y_i - \hat{y}_{i,-i})^2 + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \quad (1)$$

where M is the number of training samples, w_i are the weights, as computed by the existing kernels, γ is a penalty term to prevent the receptive fields from shrinking indefinitely, D is the covariance matrix of the Gaussian kernel and N is the dimension of the input space.

B. Autoregressive LWPR

The prognostic algorithm uses LWPR to predict the asset's future state, given part of the state history (depending on the order of the model) and the current input (see Equation 2).

$$y_n = f(y_{n-k:n-1}, u_{n-m:n-1}) \quad (2)$$

Equation 2 describes the general case of an Auto-Regressive model with Exogenous inputs (ARX). In typical ARX models $f(.,.)$ is a polynomial of the inputs and the previous states of the system: $A(B)y(t) = C(B)u(t-1)$, where A and C are polynomials of the *back shift operator* B. LWPR extends the representational power of the ARX, to include arbitrarily complex functions of the inputs. However, using LWPR doesn't yield theoretical guarantees about the ARX model's stability. The authors intend to investigate further on this matter.

C. Training

The LWPR model was trained as follows: Firstly, $init\_D$ was tuned without activating the online adaptation feature. Tuning this parameter was formulated as an optimization problem. The objective function was the Mean Square Error (MSE) of the predictions of the model, applied to a cross-validation test. The optimization problem was solved using $fmincon$ from MATLAB's optimization toolbox. This particular solver ($fmincon$), however, is mostly suitable for convex optimization problems. For this reason, the optimization problem was solved several times from random starting points, to avoid local minima. Finally, the parameter $init\_alpha$ was optimized in a similar fashion, having first enabled the online adaptation of the receptive field parameters ($update\_D = 1$). The resulting parameters are summarized in Table I.

Figures 1, 2 show the generalization capacity of the resulting model. The performance of the model is tested on an engine, which was not used for training. In Figure 1, the model was utilized to perform one-step predictions on random state samples. In Figure 2, the estimated trajectory of the state is compared with the ground truth.

TABLE I
LWPR PARAMETERS

$init\_D$	$init\_alpha$	$penalty$
40	20	0.001

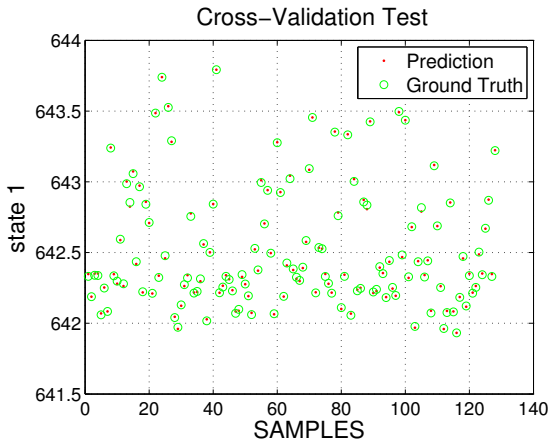

Fig. 1. This graph shows the one step prediction from the model on a cross-validation set compared with the original value. The size of the cross-validation set consists of approximately 20% of the full data. The mean square error for the one-step prediction was $1.1 \cdot 10^{-4}$.

IV. Robust Classification

State-of-the-art classification algorithms combine under-sampling or oversampling with boosting. In boosting, multiple weak classifiers are trained sequentially, to compensate for the misclassified samples. The most popular algorithm of this class is AdaBoost [22]. AdaBoost begins with the training of a single weak classifier (e.g. SVM). The missclassified samples of the training dataset are given higher weights.

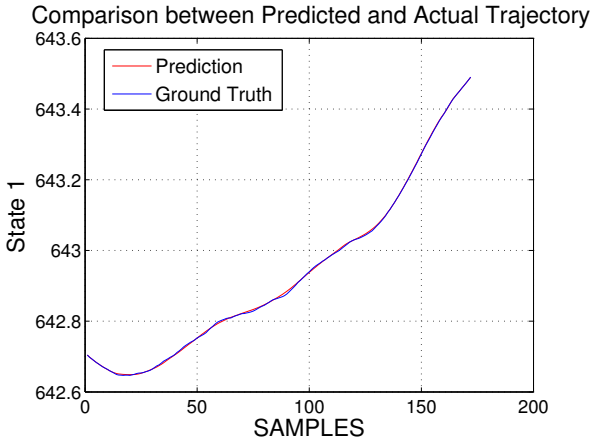

Fig. 2. This graph shows the reconstructed trajectory of the first state of the engine as computed by our model. The auto-regressive model uses 8 lags for both the input and the output. To create this graph the actual input of the engine was used. We can see that the estimated trajectory accurately follows the original state trajectory of the engine.

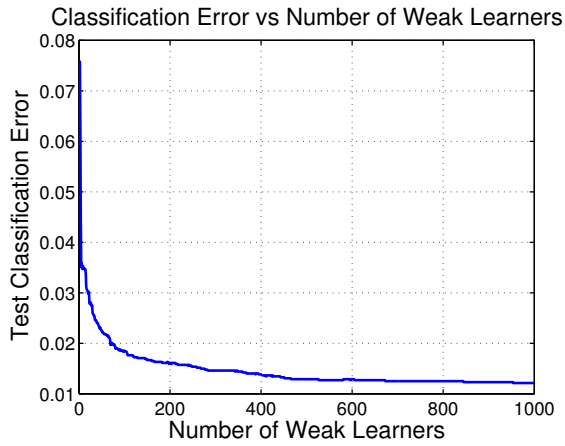

Fig. 3. This figure shows the miss-classification rate for the RUSBoost classifier versus the number of weak learners used. The performance of the algorithm was evaluated on a cross-validation set. The classification error converges asymptotically to its minimum after 700 weak learners.

Then another weak classifier is trained on the dataset, with focus on the data with higher weights. This process is repeated until the desired performance has been achieved. Below, two algorithms, which utilize sampling together with boosting, are described.

Synthetic Minority Oversampling TEechnique Boosting (SMOTEBoost) [23] iteratively trains classifiers on datasets with oversampled instances of the under-represented class [24]. In this way, the importance of the misclassification for the minority class is increased. The overfitting problem persists, albeit the missclassification error is decreased. Random Under-Sampling Boosting (RUSBoost) [1] balances the training set by removing instances of the over-represented class. One weak classifier is trained on the resulting dataset. Then the classifiers performance is checked against the original dataset. The weights are adjusted as in the AdaBoost algorithm to give higher importance to the misclassified instances. A new dataset is created and the process is repeated. In this way, RUSBoost avoids the drawbacks of discarding useful information, while it preserves the class balance within each dataset.

RUSBoost was applied on the training dataset, provided by NASA Ames center. The training set comprises historical data from a fleet of one hundred engines. Decision Trees were the weak classifier of choice. Figure 3 shows the miss-classification rate as a function of the number of weak classifiers used. The confusion matrix of the RUSBoost classifier and of an AdaBoost classifier are shown in Table II for comparison. It is obvious how RUSBoost outperforms AdaBoost, especially in the number of false positives.

V. Remaining Useful Life Computation

The remaining useful life is computed by combining the autoregressive model with the classifier. Firstly, the LWPR is trained on a subset of the full dataset. After the training is over, the algorithm is equipped with an ensemble of $n \times m$ models, where n is the number of engines in the training dataset and m is the dimension of the engine's state space.

TABLE II
Confusion matrices for AdaBoost (above) and RUSBoost (below)

		Predicted	
Actual	Class	Healthy	Faulty
	Healthy	100	0
	Faulty	100	0

		Predicted	
Actual	Class	Healthy	Faulty
	Healthy	98.8073	1.1927
	Faulty	0	100

The RUSBoost classifier is trained on the same dataset. Following that, for each engine in the test dataset a set of $1 \times m$ models is selected from the ensemble, which most accurately capture its dynamics. To this end, the response of all the models in the ensemble is simulated, using the input of the engine of interest. Then for each estimated trajectory the *coefficient of determination* (see Equation 3) is computed. The model with a coefficient nearest to unity is selected.

$$\begin{aligned} R^2 &= 1 - \frac{SS_{res}}{SS_{tot}} \\ SS_{res} &= \sum_i (y_i - f_i)^2 \\ SS_{tot} &= \sum_i (y_i - \bar{y})^2 \end{aligned} \tag{3}$$

After each engine has been associated with a model, the respective states are simulated and fed into the classifier. The time instant at which the state is classified as faulty is the remaining useful life of the engine. A major issue is that the future input of the engine is not known a priori. A simple solution would be to use a constant input, equal to the mean of the input hitherto. Alternatively, the input could be kept constant to the value of the *Root Mean Square* (RMS) or to its last known value. A more sophisticated way to deal with this problem would be to use a seperate AR model to forecast the input. This model is trained using the input history. During the experiments, all the aforementioned input forecasting techniques were tested. The results are discussed in detail in section VI.

VI. Experiments

The dataset that was used to test the prognostic algorithm comprises information from one hundred turbofan engines. For each engine, data samples from twenty-one sensors, together with three input variables (operating conditions) have been recorded. Different initial wear and variation in the engine's dynamics have been assumed. The engine develops a fault at a random time instant. The state trajectories, however, are recorded from the beginning of the engine's operation, to the point of failure. Only one fault is present in this dataset. We used 80% of the engines for training and 20% for validation. The classifier and the adaptive autoregressive models were trained as described above.

For each engine in the validation set, a model was selected, as discussed in Section V. Next, the model was simulated

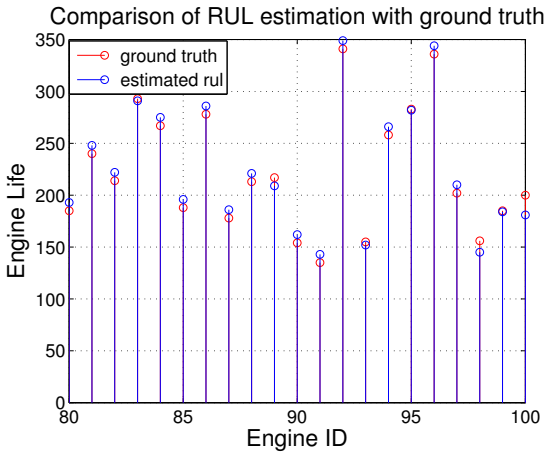

Fig. 4. This graph shows the total engine life, as given by the dataset, compared to the total life, computed by our prognostic algorithm. The total life is computed by adding the time when the forcast started with the remaining useful life prediction of the engine. The input used in this experiment was assumed to be known (the input history from the dataset has been used)

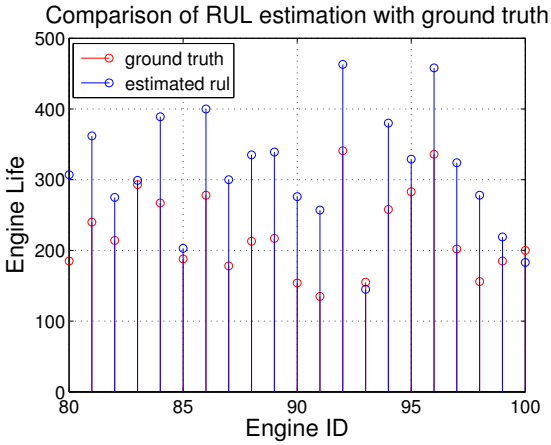

Fig. 5. In this experiment, the computation was repeated as described in Figure 4 with different assumptions over the input. Here we didn't use the true values of the future input. A constant value equal to the last known input was used for the future predictions of the model. We can see that the accuracy of the estimation dropped significantly.

starting from a user-defined time (e.g. 50 time samples before the end of the trajectory) to the future. The starting time of the simulation may influence the prognostic accuracy, depending on which input forcasting method is used. In Figure 4 the total life of the test engines is shown. In this experiment, the future input is assumed to be known a priori. The algorithm predicts the Remaining Useful Life with a mean square error of six cycles. Next, in an effort to relax the assumption with respect to the input, three more experiments have been conducted. In the first experiment, the input was kept constant, to its latest value (see Figure 5). In the next experiment, the input was kept constant to the mean of the input history (see Figure 6). Finally, the last graph shows the total life prediction, when the input was forecasted by a fourth-order auto-regressive model.

One last experiment was performed, to test the robustness of the RUSBoost algorithm, in the case of multiple faults. To

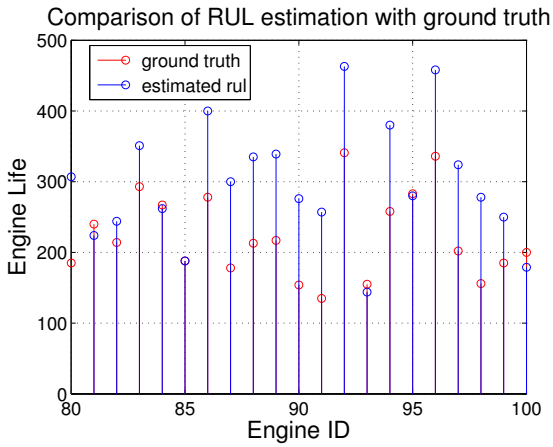

Fig. 6. In this experiment, the input was assumed to be equal to the mean value of the previous input trajectories. Again, the performance becomes worse for the majority of the test engines.

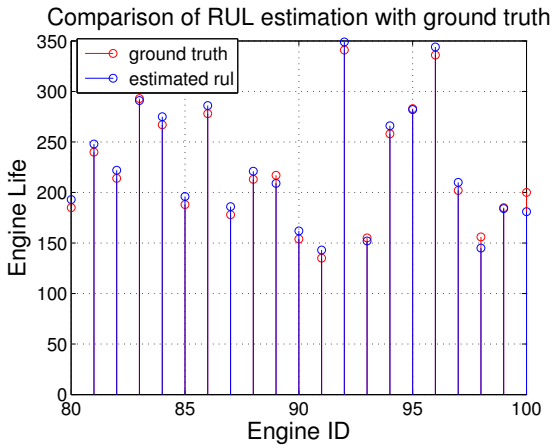

Fig. 7. In this experiment, the input was forecasted by an Auto-Regressive model (AR). The AR model was of order 4 and its parameters were identified using the past input history. In this figure, we can see that the algorithm still predicts the total life of the engine quite accurately.

this end, the prognostic algorithm was applied on a dataset with two types of faults. The results are illustrated in Figure 8. The performance of the classifier was not influenced by the presence of different fault types.

VII. Conclusions and Future Work

In this paper, a novel approach for the computation of the remaining useful life was presented. Combining a state of the art regression method for state estimation, together with a robust classifier yielded accurate prognostic predictions. In the case of unknown future inputs, an Auto-Regressive model has been used. Forecasting the input using AR doesn't seem to influence the accuracy of the predictions. Moreover, the variation of the engine's dynamics has been addressed. Given a representative training dataset, it was shown that the algorithm is robust to manufacturing uncertainty. Additionally, the algorithm performed equally well on the dataset where two distinctive faults occured.

The authors intend to explore further theoretical notions, such as stability criteria for the autoregressive model. More-

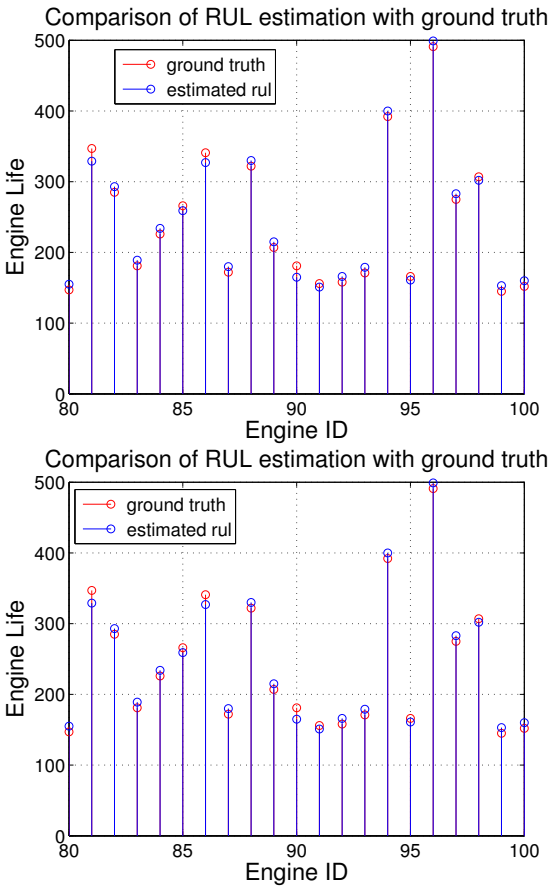

Fig. 8. We repeated the experiment described in Figure 4 on another dataset provided by Ames Center. The difference in that one is that it comprises engines with two types of fault. The purpose of this experiment was to check the robustness of the RUSBoost classifier in the case of multiple faults. The figure above shows the results when the original input is used. The figure below was created using an autoregressive model to forecast the input.

over, the model selection process can be improved. We are considering of introducing the initial wear and the manufacturing uncertainty as latent variables inside the model. In this manner, the model selection will be more rigorous and computationally efficient. Tests on real-world data are also planned to evaluate the algorithm's performance.

REFERENCES

[1] C. SEIFFERT, T. M. KHOSHGOFTAAR, J. VAN HULSE, AND A. NAPOLITANO, "RUSBOOST: IMPROVING CLASSIFICATION PERFORMANCE WHEN TRAINING DATA IS SKEWED," IN *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, PP. 1–4. 00014.

[2] S. VIJAYAKUMAR, A. D'SOUZA, AND S. SCHAAL, "LWPR: A SCALABLE METHOD FOR INCREMENTAL ONLINE LEARNING IN HIGH DIMENSIONS," 2005.

[3] A. SAXENA, K. GOEBEL, D. SIMON, AND N. EKLUND, "DAMAGE PROPAGATION MODELING FOR AIRCRAFT ENGINE RUN-TO-FAILURE SIMULATION," IN *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, PP. 1–9, IEEE, 2008.

[4] A. SAXENA AND K. G. (2008), "C-MAPSS DATA SET", NASA AMES PROGNOSTICS DATA REPOSITORY." HTTP://TI.ARC.NASA.GOV/PROJECT/PROGNOSTIC-DATA-REPOSITORY.

[5] N. GEBRAEEL, M. LAWLEY, R. LIU, AND V. PARMESHWARAN, "RESIDUAL LIFE PREDICTIONS FROM VIBRATION-BASED DEGRADATION SIGNALS: A NEURAL NETWORK APPROACH," *Industrial Electronics, IEEE Transactions on*, VOL. 51, NO. 3, PP. 694–700, 2004.

[6] K. KAZMIERCZAK, "APPLICATION OF AUTOREGRESSIVE PROGNOSTIC TECHNIQUES IN DIAGNOSTICS," IN *Proceedings of the Vehicle Diagnostics Conference, Tuczno, Poland*, 1983.

[7] C. J. LU AND W. O. MEEKER, "USING DEGRADATION MEASURES TO ESTIMATE A TIME-TO-FAILURE DISTRIBUTION," *Technometrics*, VOL. 35, PP. 161–174, MAY 1993.

[8] N. Z. GEBRAEEL, M. A. LAWLEY, R. LI, AND J. K. RYAN, "RESIDUAL-LIFE DISTRIBUTIONS FROM COMPONENT DEGRADATION SIGNALS: A BAYESIAN APPROACH," *IIE Transactions*, VOL. 37, NO. 6, PP. 543–557, 2005.

[9] D. Y. KIM, S.-G. LEE, AND M. JEON, "OUTLIER REJECTION METHODS FOR ROBUST KALMAN FILTERING," IN *Future Information Technology* (J. J. PARK, L. T. YANG, AND C. LEE, EDS.), NO. 184 IN COMMUNICATIONS IN COMPUTER AND INFORMATION SCIENCE, PP. 316–322, SPRINGER BERLIN HEIDELBERG, JAN. 2011.

[10] D. C. SWANSON, J. MICHAEL SPENCER, AND S. H. ARZOUMANIAN, "PROGNOSTIC MODELLING OF CRACK GROWTH IN A TENSIONED STEEL BAND," *Mechanical systems and signal processing*, VOL. 14, NO. 5, PP. 789–803, 2000.

[11] M. E. ORCHARD AND G. J. VACHTSEVANOS, "A PARTICLE-FILTERING APPROACH FOR ON-LINE FAULT DIAGNOSIS AND FAILURE PROGNOSIS," *Transactions of the Institute of Measurement and Control*, VOL. 31, NO. 3-4, PP. 221–246, 2009.

[12] V. SKORMIN, L. POPYACK, V. GORODETSKI, M. ARAIZA, AND J. MICHEL, "APPLICATIONS OF CLUSTER ANALYSIS IN DIAGNOSTICS-RELATED PROBLEMS," IN *Aerospace Conference, 1999. Proceedings. 1999 IEEE*, VOL. 3, PP. 161–168, IEEE, 1999.

[13] C. CORTES AND V. VAPNIK, "SUPPORT-VECTOR NETWORKS," *Machine learning*, VOL. 20, NO. 3, PP. 273–297, 1995.

[14] K. CRAMMER AND Y. SINGER, "ON THE ALGORITHMIC IMPLEMENTATION OF MULTICLASS KERNEL-BASED VECTOR MACHINES," *The Journal of Machine Learning Research*, VOL. 2, PP. 265–292, 2002.

[15] A. K. JARDINE, D. LIN, AND D. BANJEVIC, "A REVIEW ON MACHINERY DIAGNOSTICS AND PROGNOSTICS IMPLEMENTING CONDITION-BASED MAINTENANCE," *Mechanical Systems and Signal Processing*, VOL. 20, PP. 1483–1510, OCT. 2006.

[16] X.-S. SI, W. WANG, C.-H. HU, AND D.-H. ZHOU, "REMAINING USEFUL LIFE ESTIMATION - A REVIEW ON THE STATISTICAL DATA DRIVEN APPROACHES," *European Journal of Operational Research*, VOL. 213, PP. 1–14, AUG. 2011.

[17] O. GERAMIFARD, J. X. XU, J. H. ZHOU, AND X. LI, "CONTINUOUS HEALTH CONDITION MONITORING: A SINGLE HIDDEN SEMI-MARKOV MODEL APPROACH," IN *IEEE Conference on Prognostics and Health Management*, PP. 1–10, 2011.

[18] D. A. TOBON-MEJIA, K. MEDJAHER, N. ZERHOUNI, AND G. TRIPOT, "HIDDEN MARKOV MODELS FOR FAILURE DIAGNOSTIC AND PROGNOSTIC," IN *Prognostics and System Health Management Conference (PHM-Shenzhen)*, PP. 1–8, 2011.

[19] D. TOBON-MEJIA, K. MEDJAHER, N. ZERHOUNI, AND G. TRIPOT, "A MIXTURE OF GAUSSIANS HIDDEN MARKOV MODEL FOR FAILURE DIAGNOSTIC AND PROGNOSTIC.," IN *6th Annual IEEE Conference on Automation Science and Engineering, CASE'10.*, PP. 338–343, 2010.

[20] S. LÉTOURNEAU, C. YANG, AND Z. LIU, "ON-DEMAND REGRESSION TO IMPROVE PRECISENESS OF TIME TO FAILURE PREDICTION," IN *The Proceedings of the 2007 AAAI Fall Symposium on AI for Prognostics, Arlington, Virginia, USA*, 2007.

[21] S. WOLD, J. TRYGG, A. BERGLUND, AND H. ANTTI, "SOME RECENT DEVELOPMENTS IN PLS MODELING," *Chemometrics and Intelligent Laboratory Systems*, VOL. 58, NO. 2, PP. 131 – 150, 2001.

[22] Y. FREUND AND R. E. SCHAPIRE, "A DESICION-THEORETIC GENERALIZATION OF ON-LINE LEARNING AND AN APPLICATION TO BOOSTING," IN *Computational learning theory*, PP. 23–37, SPRINGER, 1995.

[23] N. V. CHAWLA, A. LAZAREVIC, L. O. HALL, AND K. W. BOWYER, "SMOTEBOOST: IMPROVING PREDICTION OF THE MINORITY CLASS IN BOOSTING," IN *Knowledge Discovery in Databases: PKDD 2003*, PP. 107–119, SPRINGER, 2003.

[24] N. V. CHAWLA, K. W. BOWYER, L. O. HALL, AND W. P. KEGELMEYER, "SMOTE: SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE," *arXiv preprint arXiv:1106.1813*, 2011. 01846.