Sparse Coding-Based Failure Prediction for Prudent Operation of LED Manufacturing Equipment

Jia-Min Ren[1], Chuang-Hua Chueh[1], and H. T. Kung[2]

[1]*Computational Intelligence Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan*
jmren@itri.org.tw
chchueh@itri.org.tw

[2]*Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge MA, USA*
kung@harvard.edu

ABSTRACT

A sudden failure of a critical component in light-emitting diode (LED) manufacturing equipment would result in unscheduled downtime, leading to a possibly significant loss in productivity for the manufacturer. It is therefore important to be able to predict upcoming failures. A major obstacle to failure prediction is the limited amount of equipment lifecycle data available for training, as equipment failure is not expected to be frequent. This calls for machine learning techniques capable of making accurate failure predictions with limited training data. This paper describes such a method based on sparse coding. We demonstrate the prediction performance of the method on a real-world dataset from LED manufacturing equipment. We show that sparse coding can draw out salient features associated with failure cases, and can thus produce accurate failure predictions. We also analyze how sparse coding-based failure prediction can lead to significant efficiency improvements in equipment operation.

1. INTRODUCTION

Reducing costs and increasing productivity are crucial concerns in the competitive manufacturing industry. Many manufacturers are seeking to implement intelligent manufacturing processes including the use of automated data analysis techniques (Scoville, 2011) which can allow for cost- and time-saving predictive maintenance (Rothe, 2008). This paper addresses approaches to optimizing predictive maintenance methods for light-emitting diode (LED) manufacturing equipment.

Modern LEDs are multilayered structures of chemical materials in which the thickness and composition of the various layers determine the color and brightness of the emitted light and device energy efficiency. The layers are deposited sequentially through the metal organic chemical vapor deposition (MOCVD) process, a critical determinant process in LED performance. The crystalline structure of each new layer is epitaxially aligned with that of the underlying layer. This complex process is affected by the conditions of a multitude of components, such as pumps, heaters, mass flow controllers, and particle filters [1]. Unexpected component failure can reduce LED production yields, and finding and repairing the source of the failure can take engineers up to 5 days. For example, the failure of a particle filter will cause the pump to shut down, resulting in all the raw materials consumed in that run to be wasted. Here, we focus on developing a failure prediction algorithm for the particle filter to ensure continuous high-performance operation in the MOCVD process.

Learning features associated with failure cases plays a critical role in a data-driven prediction approach. The goal is to come up a compact yet discriminative feature representation, in which samples related to failure cases can be accurately expressed and easily differentiated from others. Many feature learning methods have been discussed in the literature; see, e.g., Huang & Aviyente, 2006. These include principle component analysis (PCA), linear discriminant analysis (LDA) and sparse coding (SC). The SC approach used in this paper has emerged as one of the most popular feature learning methods in recent years, in areas such as computer vision (Wright et al., 2010). It computes a sparse representation of input data in terms of a linear combination of atoms in an overcomplete dictionary (more details are given in Section 4.1). Compared to methods based on

[1] Note that by a *particle filter* we mean in this paper a *physical* filter in MOCVD equipment. This is not to be confused with particle filtering methods in the diagnostics and prognostics prediction literature (see, e.g., Orchar & Vachtsevanos, 2007).

orthonormal transformations, SC has been shown to offer superior performance in a variety of applications including face recognition (Wright et al., 2009), emotion recognition (Chen et al., 2015), and wireless link prediction (Tarsa et al., 2015). Therefore, the proposed failure predictor is based on SC.

In evaluating our SC-based approach, we use real-world sensor data from LED MOCVD equipment. We found that the SC-based failure prediction method can improve F-measure about 39% over a conventional PCA-based approach. In terms of annual uptime of MOCVD equipment operation, the SC-based failure prediction method can increase it by about 600 hours over a traditional preventative maintenance policy. To the best of our knowledge, this work is the first application of SC to the prediction of component failure in MOCVD equipment.

The remainder of this paper is organized as follows: Section 2 provides an overview of failure prediction problem and prior work. Section 3 describes the dataset used in our experiments. Section 4 explains the basic concept of sparse coding and the proposed failure prediction pipeline for MOCVD equipment. In Section 5, we show the experimental results of PCA-based and SC-based failure prediction methods. Cost-benefit analysis for different maintenance and prediction policies is discussed in Section 6. Conclusions are given in Section 7.

2. FAILURE PREDICTION PROBLEM AND PRIOR WORK

The goal of failure prediction is to predict an upcoming component failure in MOCVD equipment, and raise an alarm or a maintenance advice to the manufacturer who can then intervene to prevent unscheduled downtime. In this paper, we focus on the next-run failure prediction. In other words, following each run, the system provides a prediction of whether the particle filter will fail in the next run. Here a run denotes an execution of a fabrication task on MOCVD equipment. The next-run failure prediction can be simply considered as a decision problem of predicting a yes or no outcome. We thus address it as a binary classification task.

Failure prediction methods can be roughly categorized into model-based and data-driven approaches (Lee et al., 2014). Model-based approaches have been traditionally used to understand failure mode progression associated with equipment components. In training physics-model parameters, such as those in Kalman or particle filter methods (Orchar and Vachtsevanos, 2007), model-based methods usually require relatively smaller amounts of data. However, to achieve acceptable performance in prediction, building an appropriate physical model would require detailed mechanistic knowledge and could be time-consuming. In contrast, data-driven approaches build a machine learned model based on observed sensor data from equipment without relying strongly on domain knowledge.

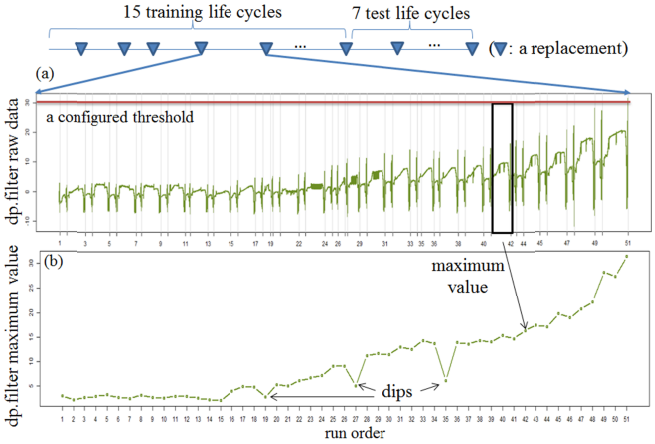

Figure 1. Data illustration of a particle filter replacement cycle. It is an example of the 22 cycles considered. (a) dp.filter raw data over runs in a replacement cycle and (b) dp.filter maximum values over runs in the same cycle. Here a run denotes an execution of a fabrication task on MOCVD equipment. (Runs may vary somewhat in their execution time.) The maximum value of dp.filter raw data is used to represent the feature of each run. Thus a replacement cycle can be represented as a sequence of these maximum values. More details about dp.filter are given in Section 3.

3. DATA DESCRIPTION

MOCVD processes produce powders and particulates which can cause unexpected and significant damage to costly pump equipment. A particle filter is therefore needed to avoid contamination of the pump. Among the sensors in MOCVD equipment, dp.filter is the most critical sensor in monitoring the particle filter status in the level of dust being accumulated there. This sensor measures the difference of pressure between the reactor and the pump. As fabrication tasks are carried out on MOCVD equipment, more powders and particulates are stacked on the filter, so dp.filter values usually increase gradually. Figure 1(a) shows an example of the degradation of the dp.filter signal in a replacement cycle. This cycle consists of multiple runs, with vertical lines denoting the boundaries between runs. Practically, the particle filter will be replaced when the maximum value of dp.filter over a run exceeds a configured threshold (e.g., 30). Accordingly, we extracted the maximum value for each run in our experiments. This means that each cycle was represented as a sequence of dp.filter maximum values as shown in Figure 1(b).

Note that the dips in Figure 1(b) were caused by executing "clean runs" on MOCVD equipment. Such clean runs are sometimes needed to clean up residual gases in the reactor mentioned earlier. The amount of gases used in a clean run is much less than those associated with a regular fabrication task execution, leading to dips in a sequence of the subsequent dp.filter maximum values.

In total, 22 replacement cycles were collected. In our experiments, the first 15 cycles and the remaining 7 cycles were respectively used to build the training and the test data. To predict whether the particle filter will fail in the next run, we labeled the previous run before the one whose dp.filter maximum value exceeded 30 as a *faulty run*. To incorporate historical information for failure prediction, we used a sliding window of size 10 runs (with adjacent windows overlapped by one run) to create the feature vector for each run. In other words, we used 10 dp.filter maximum values from the previous nine runs and the present run to represent the feature vector for the present run. Thus each created sample has a dimensionality of 10. Since each replacement cycle consists of a different number of runs (varying from 23 to 104 runs), different numbers of normal samples are thus collected for the training and the test data. Specifically, the training data consists of 387 normal and 15 faulty samples, and the test data is composed of 319 normal and 7 faulty samples.

4. SPARSE CODING BASED FAILURE PREDICTION

4.1 Basic Concept of Sparse Coding

Given N data samples $\boldsymbol{x}_i \in \mathbb{R}^{M\times 1}$, we first learn a dictionary D using the following mathematical optimization:

$$\min_{D\in C,\boldsymbol{\alpha}_i\in\mathbb{R}^{K\times 1}} \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\|\boldsymbol{x}_i - D\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1$$

$$C \triangleq \left\{D \in \mathbb{R}^{M\times K}\ s.t.\ \|\boldsymbol{d}_j\|_2^2 = 1, \forall j = 1,\dots,K\right\} \quad (1)$$

for certain $\lambda > 0$, where $\boldsymbol{\alpha}_i$ is the sparse code of $\boldsymbol{x}_i$, and D is the dictionary composed of K columns and $\boldsymbol{d}_j$ is the jth column (i.e., atom) in D. We split samples into smaller patches of length four; therefore, we have M = 4 in our experiments.

Given the learned dictionary D, we consider the following LASSO (Least Absolute Shrinkage and Selection Operator) formulated optimization problem:

$$\min_{\boldsymbol{\alpha}_i\in\mathbb{R}^{K\times 1}} \|\boldsymbol{x}_i - D\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1 \quad (2)$$

For a given data point $\boldsymbol{x}_i$, by solving the LASSO optimization using methods such as least angle regression (LARS) (Efron et al., 2004) and interior-point (Koh et al., 2007) we find the sparse code $\boldsymbol{\alpha}_i$ for $\boldsymbol{x}_i$.

4.2 LEARNING AND TRAINING PIPELINES

Figure 2 shows the learning and training pipelines for particle filter failure prediction via sparse coding. We use the following steps for dictionary learning and SVM classifier training.

1. *Patch generating*. To capture local variation within each sample, we split each sample into overlapping patches $\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_j,\dots,\boldsymbol{x}_{10\text{-}p+1}\}$, where the patch size is p and the overlapping is one. In our experiments, p is typically set to four.
2. *Patch selecting and dictionary learning*. Note that for next-run failure prediction, only the run before the last run in a replacement cycle is labeled as faulty. All other runs are labeled as normal. Thus there are far more normal runs than faulty runs. To deal with this imbalance, two dictionaries D_N and D_F are then respectively learned on normal and faulty samples using (1). (Note that if all samples were used to learn a single dictionary, the dictionary would be dominated by normal samples.)

   Specifically, to discriminatively learn these two dictionaries, only the patches whose values are all smaller than a threshold Th_N are used to learn D_N. On the other hand, if one value within the patches exceeds Th_F, these patches will be used to learn D_F. Other patches that do not satisfy either of these two conditions are discarded.
3. *Dictionary concatenating*. A simple concatenation forms the final dictionary $D = [D_N|D_F]$.

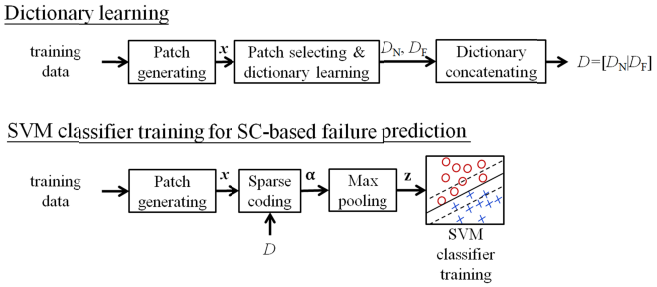

Figure 2. Dictionary learning and SVM classifier training pipelines.

The following steps are used in SVM classifier training for failure prediction based on sparse code.

1. *Patch generating*. We split each sample into overlapping patches $\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_j,\dots,\boldsymbol{x}_{10\text{-}p+1}\}$.
2. *Sparse coding*. Given the learned dictionary D, we use (2) to compute sparse code $\boldsymbol{\alpha}_j$ of each patch $\boldsymbol{x}_j$. In total, 10-p+1 sparse codes are thereby obtained for each sample.
3. *Max pooling*. To incorporate local variation of patches to reflect global features of each sample, we perform max pooling over these 10-p+1 sparse codes to obtain a pooled sparse code $\mathbf{z}$ such that the kth element in $\mathbf{z}$, $z_k = \max(\boldsymbol{\alpha}_{1,k},\dots,\boldsymbol{\alpha}_{j,k},\dots,\boldsymbol{\alpha}_{10-p+1,k})$ where $\boldsymbol{\alpha}_{j,k}$ is the kth element from $\boldsymbol{\alpha}_j$. Therefore, each sample is encoded as a pooled sparse code. The effectiveness of using pooled sparse code in classification as opposed to $\boldsymbol{\alpha}_j$ is well known (see, e.g., Chen et al., 2015, and Tarsa et al., 2015).

4. *Classifier training*. We use pooled sparse codes as features to train a linear SVM classifier for failure prediction.

4.3 FAILURE PREDICTION ON TEST SAMPLES

For a test sample, we first divide it into overlapping patches. Then, we compute sparse codes for each patch and perform max pooling on these sparse codes to obtain a pooled feature vector. Finally, the pooled feature vector is used as an input vector for the pre-trained SVM to obtain the prediction result.

5. EXPERIMENTAL SETUP AND RESULTS

5.1 Baseline: PCA+SVM

The proposed method was compared against a PCA-based baseline method. In this baseline method, PCA was used to project data onto a lower dimensional space spanned by a relatively small number of dominant eigenvectors of the covariance matrix of the training data. Then, a linear SVM was used as the predictor.

5.2 Experimental Setup and Evaluation Metrics

To provide a fair comparison, we used the same setting of the penalty parameter in a linear SVM (Chang & Lin, 2011) for both PCA- and SC-based approaches. The maximal number of principal components (PCs) used in the experiments was six, which can explain over 95% of the training data variance. Different numbers of atoms in dictionary D_N and dictionary D_F were set for comparison. Since a particle filter will be replaced when the maximum value of dp.filter over a run exceeds 30, we set Th_N and Th_F to 10 and 20 respectively. As mentioned earlier, the patch size p was empirically set to four. Sparse coding is set to use about three non-zero coefficients in our experiments.

Four standard metrics (Salfner et al., 2010)—true positive rate (*TPR*), false positive rate (*FPR*), *F-measure* and the area under the receiver operating characteristic (ROC) curve (*AUC*)—were used to compare the performance of different methods. Note that a positive sample means a faulty sample in our experiments.

5.3 Patch Selection for Dictionary Learning

To learn the two dictionaries D_N and D_F that can respectively represent normal and faulty regularities, we generated two patch sets. For this, two patch selection schemes were compared. As shown in Figure 3(a), two patch sets were separately created by considering whether the patch belongs to the last sliding window in each cycle. (Note that we can create seven patches from a window consisting of 10 runs.) Clearly, some patches overlapped, making it difficult to differentiate between atoms in the dictionaries D_N and D_F. In contrast, we defined two thresholds to select non-overlapped patches. As shown in Figure 3(b), only the patches satisfying the constraints described in Section 4.2 were used for dictionary learning. The other patches were discarded.

Figure 4 shows the sparse codes of a patch at a faulty run using dictionaries learned by different patch selection schemes. The dictionary D_N and the dictionary D_F are respectively indexed as 1~15 and 16~18. Obviously, when using the patch selection scheme 1, the overlapping of patches makes it difficult to train these two dictionaries discriminatively. Thus the patch at a faulty run can be incorrectly coded by atoms in D_N (e.g., indices of 6, 10 and 12, as shown in Figure 4(a)). In contrast, when we separated patches into two sets without overlap, different atoms can be learned in D_N and D_F. Accordingly, the same patch can be almost coded by only the atom in D_F (e.g., the index of 18, as shown in Figure 4(b)). In summary, the patch selection scheme 2, as used in our pipeline, creates patches that can be used to train D_N and D_F more discriminately.

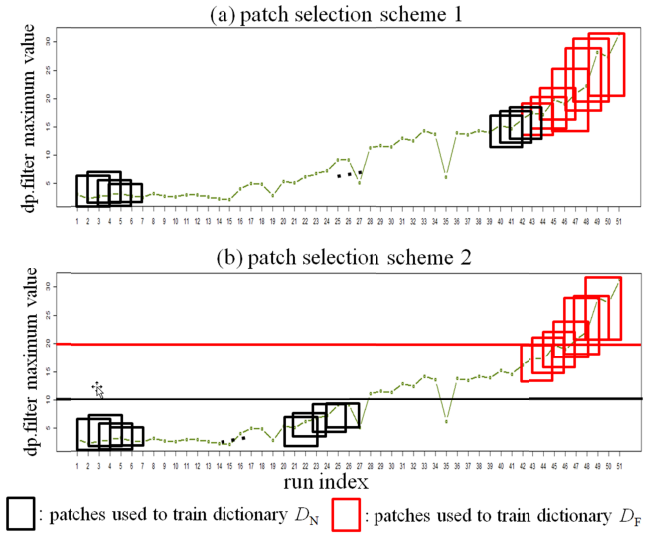

Figure 3. Two patch selection schemes. (For simplicity of plot, black points denote other overlapping patches used to train D_N)

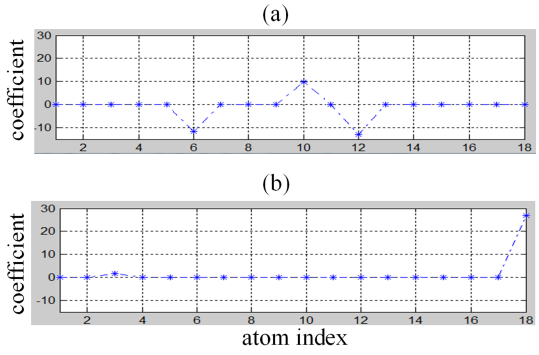

Figure 4. Sparse codes of a patch at a faulty run using dictionaries learned by patch selection (a) scheme 1 and (b) scheme 2.

Table 1. Performance comparisons. For PCA+SVM, PCs=2 (in parenthesis) denotes that data is projected onto *the first two PCs* (a similar definition applies to PCs=3 and PCs=6). For SC+SVM, 10/3 (in parenthesis) means we have *10 atoms in D_N and 3 atoms in D_F* (a similar definition applies to 15/3).

Column Index	1	2	3	4	5
Method / Metric	PCA +SVM (PCs=2)	PCA +SVM (PCs=3)	PCA +SVM (PCs=6)	SC +SVM (10/3)	SC +SVM (15/3)
TPR (%)	85.71	71.43	71.43	85.71	100
FPR (%)	16.93	12.23	5.64	9.4	13.17
F-measure	0.179	0.196	0.333	0.279	0.25
AUC	0.916	0.911	0.847	0.942	**0.983**

5.4 Prediction Results

Table 1 compares the performance of the baseline method (PCA+SVM) and the proposed method (SC+SVM). For the PCA+SVM method (columns 1, 2 and 3), when more PCs were used, we obtained higher F-measure values. This means that reserving more PCs is useful for failure prediction. Under TPR equals to 85.71% (columns 1 and 4), the proposed SC+SVM method achieves a lower FPR (9.4%) than that of the PCA+SVM method (16.93%). In other words, the proposed method raised fewer false alarms than the PCA+SVM method. The proposed method (column 5) also achieves the best prediction performance in terms of AUC.

Figure 5 shows the receiver operating characteristic (ROC) curves of these two methods under the best AUC values (columns 1 and 5 in Table 1). From this figure, we observed that the PCA+SVM method raises more false alarms.

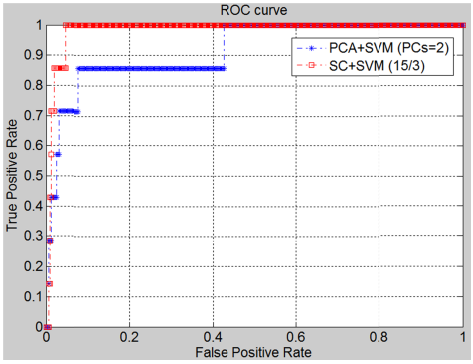

Figure 5. ROC curves of the baseline method (PCA+SVM) and the proposed method (SC+SVM).

In addition, when using a non-linear, radius basis function (RBF) SVM as a predictor rather than a linear SVM, the highest AUC value of PCA-based method achieves 0.925 (under PCs=2) and the highest AUC value of SC-based method achieves 0.965 (under 15 atoms in D_N and 3 atoms in D_F). This experiment shows again that the use of SC-based features outperforms that of traditional PCA features when a non-linear SVM is used as a predictor.

Furthermore, to assess robustness of performance against various partitions of the data set into training and test cycles, we also evaluated the performance of two other random partitions. Similar results as mentioned above were found.

6 COST-BENEFIT ANALYSIS OF DIFFERENT REPLACEMENT POLICIES FOR MOCVD EQUIPMENT

In this section, we provide cost-benefit analysis of the proposed SC-based prediction method when compared with some other methods for MOCVD equipment. Given the difficulty of quantizing detailed factors of costs such as hardware and software design, engineering qualification and certification (Saxena et al., 2010), we only consider the annual uptime of MOCVD equipment operation. We use the following notations to facilitate the discussion.

UT: average uptime per cycle,

DT: average downtime for a replacement, and

H: the probability that the maintenance time provided by a certain replacement policy is before an actual failure.

Practically, *UT* is calculated based on the maintenance time under a given replacement policy on the test cycles. *DT* is calculated as

$$H * 1.5 + (1 - H) * 108$$

where 1.5 and 108 are average downtimes (in hours) for a scheduled and an unscheduled replacement, respectively. Note that in contrast, the execution time of a run is about 8 hours. Here these average downtimes and the run's execution time were obtained from engineers who maintain MOCVD equipment.

The expected uptime in a year under a replacement policy is calculated by multiplying the number of operation units, each of which is the time duration for a pair *UT* and *DT*, in a year and the average uptime per cycle:

$$\text{Expected uptime in a year} = \frac{total\ hours\ in\ a\ year}{UT+DT} * UT$$

In addition to the SC- or PCA-based predictive maintenance policy, we consider two conventional replacement policies: (1) run-to-failure replacement policy, under which the particle filter will be used until it fails (i.e., $H = 0$), and (2) preventive maintenance policy, under which the particle filter will be replaced once the number of executed runs exceeds the average number of runs in training cycles.

Figure 6 compares uptime against the FPR under different replacement policies for the test data. The X-axis is the FPR of the PCA+SVM and SC+SVM methods. The Y-axis is the annual uptime of the MOCVD equipment. To facilitate the discussion, the total number of particle filters for which no alarm was raised under a replacement policy before they

failed is denoted as *#misses*. For example, under the run-to-failure replacement policy, *#misses* is 7 since these filters are used until they fail; under the preventive maintenance policy, only *one* filter is not replaced before it fails, so *#misses* is 1. To study cost-benefit effects of *#misses* for the PCA+SVM and SC+SVM methods, we consider the three FPR subintervals shown in Figure 6. In these FPR regions various methods exhibit their relative strengths.

Interval$_1$ (*#misses* is larger than 1 for both PCA+SVM and SC+SVM). When we allow a very low FPR, it is unlikely that an alarm will be raised. Then the performance of both the baseline method and the proposed method is worse than that of preventive maintenance in terms of uptime (as shown in the lower-right zoomed-in panel of Figure 6).

Interval$_2$ (*#misses* is 1 for SC+SVM, and *#misses* is either 2 or 3 for PCA+SCM). We compare the proposed method with preventive maintenance under the same #misses. We can see that the proposed SC-based method outperforms the preventive maintenance strategy with an increased uptime of 300+ hours (as shown in the upper-right zoomed-in panel of Figure 6).

Interval$_3$ (*#misses* is 0 for SC+SVM, and *#misses* is either 1 or 2 for PCA+SVM). The proposed SC-based method successfully raises alarms before any particle filter fails, and therefore achieves the best uptime among all methods.

In summary, the proposed SC-based method outperforms the other replacement policies in Interval$_2$ and Interval$_3$. Particularly, the proposed SC+SVM method outperforms the preventive maintenance policy in annual uptime by 600+ hours under FPR equal to 5% (as shown in the upper-right zoomed-in panel of Figure 6). This suggests that under the given data set, when the proposed SC-based prediction method is used, the target FPR should be set at 5%. Note that if FPR is set too high (e.g., 50%), the proposed SC-based failure predictor would incur many false alarms, leading to a shortened uptime.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a sparse coding-based failure prediction method for the particle filter in MOCVD equipment. Using a real-world dataset, our proposed SC-based method raises fewer false alarms than a PCA-based baseline method under the same TPR. Compared with the preventative maintenance strategy, the proposed SC-based method increases the annual uptime of MOCVD equipment by 600+ hours. To the best of our knowledge, this work is the first application of sparse coding to the prediction of component failure in MOCVD equipment, with performance demonstrated using a real-world dataset.

The paper focuses on classifiers rather than their ensembles. It is generally true that ensemble methods are often more accurate than their individual classifiers provided that the latter are accurate and diverse (Dietterich, 2000). As a future work, we plan to study ensemble methods based on the proposed SC-based classifiers of this paper.

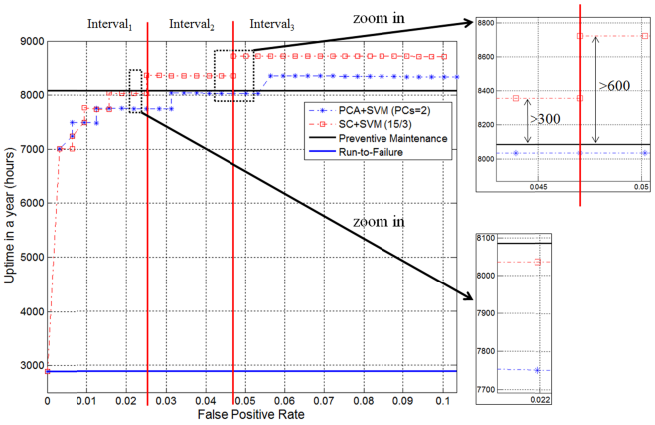

Figure 6. Annual uptime of MOCVD equipment operation under different replacement policies.

ACKNOWLEDGEMENT

This work is partially supported by the project of Big Data Technologies and Applications, Computational Intelligence Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan. H. T. Kung's research at Harvard for this paper was supported in part by gifts from the Intel Corporation and in part by the Naval Postgraduate School Agreement No. N00244-15-0050 awarded by the Naval Supply Systems Command. The views expressed in this paper do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

REFERENCES

Chang C. C., & Lin. C. J. (2011). LIBSVM: a library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, Vol. 2, No. 3, pp.1-27.

Chen H. C., Comiter M., Kung H. T., & McDanel B. (2015). Sparse Coding Trees with Application to Emotion Recognition, *In Proceedings of IEEE Workshop on Analysis and Modeling of Faces and Gestures*, pp. 77-86.

Dietterich T. G. (2000). Ensemble Methods in Machine Learning, *Multiple Classifier Systems, Lecture Notes in Computer Science*, Vol. 1857, pp. 1-15.

Efron B., Hastie T., Johnstone I., & Tibshirani, R. (2004). Least angle regression, *The Annals of Statistics*, Vol. 32, No. 2, pp. 407-499.

Huang K., & Aviyente S. (2006). Sparse Representation for Signal Classification, *Advances in Neural Information Processing Systems*, Vol. 19, pp. 131-138.

Lee J., Wu F., Zhao W., Ghaffari M., Liao L., & Siegel D. (2014). Prognostics and Health Management Design for Rotary Machinery Systems—Reviews, Methodology and Applications, *Mechanical Systems and Signal Processing*, Vol. 42, pp. 314-334.

Koh K., Kim S. J., & Boyd S. (2007). An Interior-Point Method for Large-Scale l1-Regularized Logistic Regression, *The Journal of Machine Learning Research*, Vol. 8, pp.1519-1555.

Mairal J., Bach F., Ponce J., & Sapiro G. (2009). Online Dictionary Learning for Sparse Coding, *In Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 689-696.

Mairal J., Bach F., Ponce J., & Sapiro G. (2010). Online Learning for Matrix Factorization and Sparse Coding, *The Journal of Machine Learning Research*, Vol. 11, pp. 19-60.

Orchard M. E., & Vachtsevanos G. J. (2007). A Particle Filtering-based Framework for Real-time Fault Diagnosis and Failure Prognosis in a Turbine Engine, *Mediterranean Conference on Control and Automation*, pp. 1-6.

Saxena A., Roychoudhury I., Celaya J., Saha S., Saha B., & Goebel K. (2010). Requirements Specifications for Prognostics: An Overview, *AIAA Infotech@Aerospace Conference*.

Salfner F., Lenk M., & Malek M. (2010). A survey of Online Failure Prediction Methods, *ACM Computing Surveys*, Vol. 42, No. 3, pp. 1-42.

Scoville J. (2011). Predictability as a Key Component of Productivity, *ISMI Manufacturing Week* 2011, Austin, TX.

Tarsa S. J., Comiter M., Crouse M., McDanel B., & Kung H. T. (2015). Taming Wireless Fluctuations by Predictive Queuing Using a Sparse-Coding Link-State Model, *ACM MobiHoc*, pp. 287-296.

Wright J., Yang A. Y., Ganesh A., Sastry S. S., & Yi M. (2009). Robust Face Recognition via Sparse Representation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 31, No. 2, pp. 210-227.

Wright J., Yi M., Mairal J., Sapiro G., Huang, T. S., & Yan S. (2010). Sparse Representation for Computer Vision and Pattern Recognition, *Proceedings of the IEEE*, Vol. 98, No. 6, pp. 1031-1044.

Zhu J., Nostrand T., Spiegel C., & Morton B. (2014). Mechanical Diagnostics System Engineering in IMS HUMS, *In Proceedings of Annual Conference of the Prognostics and Health Management Society*, pp. 635-647.

BIOGRAPHIES

Jia-Min Ren received his Ph.D. from the Computer Science Department, National Tsing Hua University, Hsinchu, Taiwan. He currently works at the Computational Intelligence Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan. His research interests include machine learning, semantic analysis of musical signals, music information retrieval, and analysis of manufacturing data.

Chuang-Hua Chueh received his Ph.D. from the Computer Science Department, National Cheng Kung University, Tainan, Taiwan. He currently works at the Computational Intelligence Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan. His research interests include machine learning, pattern recognition and speech recognition.

H. T. Kung is William H. Gates Professor of Computer Science and Electrical Engineering at Harvard University. He is interested in computing, communications and sensing, with a current focus on machine learning, compressive sampling and the Internet of Things. He has been a consultant to major companies in these areas. Prior to joining Harvard in 1992, he taught at Carnegie Mellon University for 19 years after receiving his Ph.D. there. Professor Kung is well-known for his pioneering work on systolic arrays in parallel processing and optimistic concurrency control in database systems. His academic honors include membership in the National Academy of Engineering in the US and the Academia Sinica in Taiwan.