Anomaly Detection and Inter-Sensor Transfer Learning on Smart Manufacturing Datasets

Mustafa Abdallah  
Purdue University  
abdalla0@purdue.edu  

Byung-Gun Joung  
Purdue University  
bjoung@purdue.edu  

Wo Jae Lee∗  
Amazon  
wojaelee@gmail.com  

Charilaos Mousoulis  
Purdue University  
cmousoul@purdue.edu  

John W. Sutherland  
Purdue University  
jwsuther@purdue.edu  

Saurabh Bagchi  
Purdue University  
sbagchi@purdue.edu  

Abstract

Smart manufacturing systems are being deployed at a growing rate because of their ability to interpret a wide variety of sensed information and act on the knowledge gleaned from system observations. In many cases, the principal goal of the smart manufacturing system is to rapidly detect (or anticipate) failures to reduce operational cost and eliminate downtime. This often boils down to detecting anomalies within the sensor data acquired from the system. The smart manufacturing application domain poses certain salient technical challenges. In particular, there are often multiple types of sensors with varying capabilities and costs. The sensor data characteristics change with the operating point of the environment or machines, such as, the RPM of the motor. The anomaly detection process therefore has to be calibrated near an operating point. In this paper, we analyze four datasets from sensors deployed from manufacturing testbeds. We evaluate the performance of several traditional and ML-based forecasting models for predicting the time series of sensor data. Then, considering the sparse data from one kind of sensor, we perform transfer learning from a high data rate sensor to perform defect type classification. Taken together, we show that predictive failure classification can be achieved, thus paving the way for predictive maintenance.

1 Introduction

Smart manufacturing application domain poses certain salient technical challenges for the use of ML-based models for anomaly detection. First, in smart manufacturing domain, there are multiple types of sensors concurrently generating data about the same (or overlapping) events. These sensors are of varying capabilities and costs. Second, the sensor data characteristics change with the operating point of the machines, such as, the RPM of the motor. The inferencing and the anomaly detection processes therefore have to be calibrated for the operating point. Thus, we need case studies of anomaly detection deployments on such systems — the need for such deployments and resultant analyses have been made for smart manufacturing systems [41, 36] (see also the survey [46] on the usage and challenges of deep learning in smart manufacturing systems). Most of the existing work has relied on classical models for anomaly detection and failure detection in such systems [45, 38, 31, 7]. While there is a rich literature on anomaly detection in many IoT-based systems [6, 34], there are few existing works that document the use of ML models for anomaly detection in smart manufacturing systems [4] (see [29] for a survey). In particular, most of the existing work focused on categorizing anomalies in the semi-conductor industry [39], windmill monitoring [22], and laser-based manufacturing [13].

∗The work conducted while the author was at Purdue University.

There is also important economic impetus for this kind of deployment and analysis. In a smart manufacturing system, various sensors (e.g., vibration, ultrasonic, pressure sensors) are applied for process control, automation, production plans, and equipment maintenance. For example, in equipment maintenance, the condition of operating equipment is continuously monitored using proxy measures (e.g., vibration and sound) to prevent unplanned downtime and to save maintenance costs [23]. Thus, the data from these sensors can be analyzed in a streaming, real-time manner to fill a critical role in predictive maintenance tasks, through the anomaly detection process [4][20][11]. Thus, we propose our anomaly detection technique for smart manufacturing systems [21]. Two notable exceptions to the lack of prior work in this domain are the recent works [23][3]. In [24], the authors proposed a kernel principal component analysis (KPCA)-based anomaly detection system to detect a cutting tool failure in a machining process. The work [3] provided a deep-learning based anomaly detection approach. However, they did not address the domain-specific challenges introduced above, did not propose any learning transfer across different manufacturing sensors as we propose here, and did not benchmark the performance of diverse forecasting models for the anomaly detection task.

Our Contribution:

In this paper, we study the maintenance problem of smart manufacturing systems by detecting failures and anomalies that would have an impact on the reliability and security of these systems. In such systems, the data are collected from different sensors via intermediate data collection points and finally aggregated to a server to further store, process, and perform useful data-analytics on the sensor readings [30][18]. We propose a temporal anomaly detection model, in which the temporal relationships between the readings of the sensors are captured via a time-series prediction model. Specifically, we consider two classes of time-series prediction models which are classical forecasting models (including Autoregressive Integrated Moving Average model (ARIMA) [10], Seasonal Naive [32], and Random Forest [27]) and new ML-based models (including Long Short-Term memory (LSTM) [15], Auto Encoder [9], and DeepAR [35]). These models are used to predict the expected future samples in certain time-frame given the near history of the readings. We first test our models on real data collected from deployed manufacturing sensors to detect anomalous data readings. We then analyze the performance of our models, and compare the algorithms of these time-series predictors for different testbeds. We observe that the best forecasting model is dataset-dependent with ML-based models giving better performance in the anomaly detection task.

Another problem in this domain is the prediction from models using sparse data, which is often the case because of limitations of the sensors or the cost of collecting data. One mitigating factor is that plentiful data may exist in a slightly different context, such as, from a different kind of sensor on the same equipment or the equipment being operated under a somewhat different operating condition in a different facility (such as a different RPM). Thus, the interesting research question in this context is: can we use a model trained on data from one kind of sensor (such as, a piezoelectric sensor, which has a high sampling frequency) to perform anomaly detection on data from a different kind of sensor (such as, a MEMS sensor, which has a low sampling frequency but is much cheaper)? In this regard, we propose a transfer-learning model that transfers learning across different instances of manufacturing vibration sensors. This transfer-learning model is based on sharing weights and feature transformations from a deep neural network (DNN) trained with data from the sensor that has a high sampling frequency. These features and weights are used in the classification problem of another sensor data, the one with lower sampling frequency. We show that the transfer-learning idea gives a relative improvement of 11.6% in the accuracy of classifying the defect type over the regular DNN model. We built variants of DNN models for the defect classification task, i.e., using a single RPM data for training and for testing across the entire operating environment, and using aggregations of data across multiple RPMs for training with interpolation within RPMs. One may wonder why we need to use sensors with much lower sampling rate; the reason is the significant price difference between the MEMS sensor and piezoelectric sensor. The former has much lower resolution (and also cost [19][2]—$8 versus $1,305). Therefore, the goal is to build a predictive maintenance model from the piezoelectric sensor and use it for the MEMS sensor.

\footnote{By classification problem here, we mean doing both re-training on the new sensor using the shared neural weights and the feature representation and then doing the defect type classification.}
In this paper, we test the following hypotheses related to anomaly detection in smart manufacturing.

**Hypothesis 1:** Deep learning-based anomaly detection technique is effective for smart manufacturing.

**Hypothesis 2:** Learning process for classifying failures is transferable across different sensor types.

Based on our analysis with real data, we have the following contributions:

1. **Anomaly Detection:** We build two classes for time series prediction models for temporal anomaly detection on real sensor data in smart manufacturing system for detecting anomaly readings collected from the deployed sensors. We test our models for temporal anomaly detection through four real-world datasets collected from manufacturing sensors (e.g., vibration data). We observe that the ML-based models outperform the classical models in the anomaly detection task.

2. **Defect Type Classification:** We detect the level of defect (i.e., normal operation, near-failure, failure) for each RPM data using deep learning (i.e., deep neural network multi-class classifier) and we transfer the learning across different instances of manufacturing sensors. We analyze the different parameters that affect the performance of prediction and classification models, such as the number of epochs, network size, prediction model, failure level, and sensor type.

3. **RPM Selection and Aggregation:** We show that training at some specific RPMs, for testing under a variety of operating conditions gives better accuracy of defect prediction. The takeaway is that careful selection of training data by aggregating multiple of predictive RPM values is beneficial.

4. **Benchmark Data:** We release our database corpus (4 datasets) and codes for the community to access it for anomaly detection and defect type classification and to build on it with new datasets and models. We are unveiling real failures of a pharmaceutical packaging manufacturer company.

## 2 Proposed Models

We now describe our proposed algorithms for the anomaly detection and defect type classification.

### 2.1 Temporal Anomaly Detection

Here, we describe our proposed algorithm for detecting anomalies from the sensor readings. First, we build time-series predictors, using different time-series predictor variants in our algorithm. We compare several state-of-the-art time-series forecasting models for our anomaly detection task on our manufacturing testbeds. They can be classified into the following two classes:

- **Classical forecasting models:** In this category, we included Autoregressive Integrated Moving Average model (ARIMA) [10], Seasonal Naive [32] (in which each forecast equals the last observed value from the same season), Random Forest (RF) [27] (which is a tree ensemble that combines the predictions made by many decision trees into a single model), and Auto-regression [26].

- **ML-based forecasting models:** We selected six popular time series forecasting models, including Recurrent Neural Network (RNN) [42], LSTM [15] (which is a better version than RNN and has been used in different applications [16]), Deep Neural Netowrk (DNN) [37], AutoEncoder [9], and the recent works DeepAR [35], DeepFactors [49].

For each model, we generated multiple variants by varying the values of hyperparameters and we chose the model variant with the best performance for each dataset. We describe the hyper-parameters and the libraries used for all forecasting models in Appendix E (in the supplementary material).

**Anomaly Detection Rule:** After using any of the above proposed time-series predictors, for each sample under test, we would have two values: the actual value (measured by the sensor) and the predicted value (predicted by our model). To flag an anomaly, we consider that \( \frac{\text{predicted value} - \text{actual value}}{\text{predicted value}} > \lambda \). In other words, the relative error between the actual value and the predicted value is more than \( \lambda \).^

^
URL for our database and codes is: https://drive.google.com/drive/u/2/folders/1QX3chnSTKd3PwEhi5k8df9WwsM0riJ8

In our work, we also used classifier-based model for anomaly detection of test samples (see Appendix F.2).
2.2 Transfer Learning across Sensor Types

We show our proposed model in Figure 1, which has two modes: In offline training, the sensor with a large amount of data (let us call it sensor type I) has its data entered to the feature extraction module that performs encoding and normalization of the input signals into numerical features. Second, a deep neural network (DNN) model is trained and tuned using these features and labels of the data (normal, near-failure or failure). We use the DNN as a multi-class classifier due to its discriminative power that is leveraged in different classification applications [51, 12]. Moreover, DNN is useful for both tasks of learning the level of defect for the same sensor type and for transfer learning across the different sensor types that we consider here. In online mode, any new sensor data under test (here, sensor type II) would have the same feature extraction process where the saved feature encoders are shared. Then, the classifier (after retraining) predicts the defect type (one of the three states mentioned earlier) given the trained model, and giving as output the probability of each class.

It is worth noting that sensor types I and II should be measuring the same physical quantity but can be from different manufacturers and with different characteristics. For instance, in our smart manufacturing domain, sensor type I is a piezoelectric sensor (of high cost but with high sampling resolution) while type II is a MEMS sensor (of lower cost but with lower sampling resolution). We propose the transfer learning for predictive maintenance i.e., predicting the level of defect of the MEMS sensor and whether it is in normal operation, near-failure (and needs maintenance), or failure (and needs replacement). We emphasize that although the two sensor types we consider for that task in our work generate different data distribution and have different sampling frequency, our transfer learning is efficient (see our evaluation in Section 4).

3 Anomaly Detection with Manufacturing Sensors

Anomalous data generally needs to be separated from machine failure as abnormal patterns of data do not necessarily imply machine or process failure [47]. We perform anomaly detection using vibration and process data to identify anomalous events and then attempt to label/link these events with machine failure information. This way, we aim to identify abnormal data and correlate the abnormal data to machine failure coming from manufacturing sensors. To achieve such goal, we build time-series models to predict (and detect) anomalies in the sensors. We first detail our datasets.

3.1 Deployment Details and Datasets Explanation

(1) Piezoelectric and MEMS datasets: To build these datasets, an experiment was conducted in the motor testbed (shown in Figure 2) to collect machine condition data (i.e., acceleration) for different health conditions. During the experiment, the acceleration signals were collected from both piezoelectric and MEMS sensors (Figure 3) at the same time with the sampling rate of 3.2 kHz and 10 Hz, respectively, for X, Y, and Z axes. Different levels of machine health condition can be induced by mounting a mass on the balancing disk (shown in Figure 4), thus different levels of mechanical imbalance are used to trigger failures. Failure condition can be classified as one of three possible
states - normal, near-failure, and failure. Acceleration data were collected at the ten rotational speeds (100, 200, 300, 320, 340, 360, 380, 400, 500, and 600 RPM) for each condition. While the motor is running, 50 samples were collected at 10 second interval, for each of the ten rotational speeds. We use this same data for defect-type classification and learning transfer tasks (Section 4).

(2) Process and Pharmaceutical Packaging datasets: In the production of injection molded plastic components, molten material is injected into a die. To increase the production rate (i.e., speed up the process), a coolant is circulated through a piping system embedded within the die to remove heat from the system. This accelerates the rate at which the die and plastic components cool and solidify, and reduces the cycle time. Of course, the coolant within this system must then have heat removed from it; this is often achieved with the aid of a chiller. Discussions with our company partner (makes plastic containers for the pharmaceutical industry) indicated that there might be concerns with the vibration of the chiller. Data were collected on the chiller vibration. We were also able to collect process related data that can potentially indicate the condition of machine operation. Such process data is being collected as part of the company’s standard statistical process control (SPC) activities; 49,706 samples of process data were collected for the period from Aug. 2021 - May 2022. One type of process data collected was the internal temperature of the chiller for the injection molding machines. In this paper, the chiller temperature was used for anomaly detection task. The chiller in the pharmaceutical process is designed to maintain the temperature of the cooling water used in the manufacturing process to around 53 Fahrenheit degree. When the chiller operation is down, the temperature of the process water varies with the ambient temperature. The sampling rate of the process data is 1 data point per 5 minutes when the SPC system is on service. As the pharmaceutical company operates non-stop from Sunday 11 pm to Friday 7 pm and shuts down from Friday 7 pm to Sunday 11 pm. When the chiller is failed, supply temperature can vary and goes up to 65 degrees.

Experimental Setup: The goal is to measure the performance of our time-series regression model to detect anomalies for the vibration sensors. We show the performance of our models in terms of the accuracy of detecting anomalies (measured by precision, recall, and F-1 score). We also use the root mean square error (RMSE) for evaluating the performance of different forecasting models on the four datasets. The goal of these time-series regression models is to extract the anomaly measures that are typically far from the predicted value of the regression model. For each proposed model, the training size was 66% of the total collected data while the testing size was 34%. We also varied the proportion of data used for training as a parameter and tested the performance of our model to check the least amount of data needed (which which was 30% of the data in our experiments) for the time-series regression model to predict acceptable values (within 10% error from the actual values). We trained the ten predictive models on specific RPM and tested on same RPM. The data contains different levels of defects (i.e., different labels for indicating normal operation, near-failure, and failure). These labels would be used in next section. In time-series prediction models, all data that have different levels of defects were tested. Specifically, the data was divided between training and testing equally. We stopped after 5 epochs as the total loss on training samples saturates.

Computing Resources: We performed anomaly detection experiments on an Intel i7 @2.60 GHz, 16GB RAM, 8-core workstation. The transfer learning experiments were performed on Dell Precision T3500 Workstation, with 8 CPU cores, each running at 3.2GHZ, 12GB RAM, and Ubuntu 16.04 OS.

3.2 Results and Insights

Performance: We first do bench-marking of the ten time-series forecasting models for each of the four datasets (described above in Section 3.1). Table 1 shows such comparison in terms of the RMSE.
We first observe that each dataset has a different best model (e.g., LSTM gave the best performance for Piezoelectric dataset while AutoEncoder was the best for Process data). Second, most of the ML-based forecasting models perform better than the traditional models. This is due to the fact that the deployments generate enough data for accurate training and due to the complex dependencies among the features of the datasets. Third, the linear models such as ARIMA and Auto-Regression were worse due to the non-linear nature of sensors’ data. We second compare the anomaly detection performance of our approach under the different forecasting models (represented by the typical metrics: Precision, Recall, and F-1 Score [5]). Table 2 shows the average performance for each metric across our four datasets. We observe that Random Forest and AutoEncoder give the first and second best anomaly detection performances, respectively (i.e., highest precision and recall). Also, Seasonal Naive and Auto Regression gave the worst performance.

**Table 1: Results for forecasting (RMSE; the lower the better) for every testbed. For each forecasting model, we choose the model with the best performance from all its model variants. We observe that the best forecasting model is task-dependent (i.e., the best model is varying depending on each dataset type).**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Seasonal Naive</th>
<th>DeepAR</th>
<th>Deep Factors</th>
<th>Random Forest</th>
<th>AutoEncoder</th>
<th>Auto-Regression</th>
<th>ARIMA</th>
<th>LSTM</th>
<th>RNN</th>
<th>DNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMS</td>
<td>0.1442</td>
<td>0.1346</td>
<td>0.2510</td>
<td>0.2295</td>
<td>0.2431</td>
<td>0.3865</td>
<td>0.2569</td>
<td>0.1450</td>
<td>0.1501</td>
<td>0.1554</td>
</tr>
<tr>
<td>Process Data</td>
<td>0.8943</td>
<td>0.8262</td>
<td>0.6468</td>
<td>0.5537</td>
<td>0.6560</td>
<td>1.1345</td>
<td>1.8240</td>
<td>0.6001</td>
<td>0.5811</td>
<td>0.7690</td>
</tr>
<tr>
<td>Pharm. Packag</td>
<td>0.5673</td>
<td>0.3654</td>
<td>0.3628</td>
<td>0.1957</td>
<td>0.1510</td>
<td>0.3962</td>
<td>1.3869</td>
<td>0.7031</td>
<td>0.7612</td>
<td>1.5820</td>
</tr>
</tbody>
</table>

**Table 2: Anomaly detection Results (Precision, Recall, and F-1 Score; the higher the better) for each forecasting model. Random Forest and AutoEncoder give the best anomaly detection performances.**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Seasonal Naive</th>
<th>DeepAR</th>
<th>Deep Factors</th>
<th>Random Forest</th>
<th>AutoEncoder</th>
<th>Auto-Regression</th>
<th>ARIMA</th>
<th>LSTM</th>
<th>RNN</th>
<th>DNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.4285</td>
<td>0.5609</td>
<td>0.3419</td>
<td>0.3835</td>
<td>0.7845</td>
<td>0.3146</td>
<td>0.7419</td>
<td>0.5135</td>
<td>0.5961</td>
<td>0.6085</td>
</tr>
<tr>
<td>Recall</td>
<td>0.4302</td>
<td>0.6571</td>
<td>0.6071</td>
<td>0.7813</td>
<td>0.7705</td>
<td>0.5945</td>
<td>0.6216</td>
<td>0.5938</td>
<td>0.6518</td>
<td>0.6744</td>
</tr>
<tr>
<td>F-1 Score</td>
<td>0.4301</td>
<td>0.6052</td>
<td>0.5721</td>
<td>0.8064</td>
<td>0.7209</td>
<td>0.5499</td>
<td>0.6764</td>
<td>0.5207</td>
<td>0.6361</td>
<td>0.6517</td>
</tr>
</tbody>
</table>

4 Transfer Learning across Vibration Sensors

In this section, we use our transfer-learning proposed model to detect the level of defect of the readings from the manufacturing sensors. In this context, we evaluate the performance of the model on two real datasets from our manufacturing sensors which are piezoelectric and MEMS vibration sensors. In other words, we perform data analytics on the data from the vibration sensors and infer one of three operational states (mentioned in Section 3) for the motor. We show the performance of our model in terms of the accuracy of detecting defect level as measured by the classification accuracy of the deep-learning prediction model on the test dataset which is the defined as the number of correctly classified samples to the total number of samples. We study different parameters and setups that affect the performance. We answer the following two research questions in this section:

- Can we detect the operational state effectively (i.e., with high accuracy)?
- Can we transfer the learned model across the two different types of sensors?

4.1 DNN Model Results

**Experimental Setup and Results:** We collected the data from 2 deployed sensors, i.e., piezoelectric and MEMS sensors mentioned earlier. Then, two DNN models were built on these two datasets. First, a normal model for each RPM was built where we train a DNN model on around 480K samples for the RPM. We have a sampling rate of 3.2 KHz (i.e., collect 3.2K data during 1 second) and we collect 50 samples and we have 3 axes. So, total data for one experiment is $3200 \times 50 \times 3 = 480K$ data points. For testing on same RPM, the training size was 70% of the total collected data while the testing size was 30%. The baseline DNN model consists of 50 neurons per layer, 2 hidden layers (with ReLU activation function for each hidden layer) and output layer with Softmax activation function. Following standard tuning of the model, we created different variants of the models to choose the best parameters (by comparing the performance of the multi-class classification problem). We built upon the Keras library [17] which is Python-based for creating the variants of our models. In our results, we call the two models DNN-R and DNN-TL where the first refer to training DNN regularly and testing on the same sensor while the latter means transfer learning model where training was performed on one sensor and classification was performed on a different sensor (using the design of shared weights and learned representations as described in Sec. 2.2). Specifically, for the DNN-TL, training was
Table 3: A comparison between regular DNN model and our transfer-learning DNN model on MEMS sensors. The transfer-learning model gives an absolute gain of 7.48% over regular DNN model.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Sensor Tested</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN-R</td>
<td>MEMS</td>
<td>64.23%</td>
</tr>
<tr>
<td>DNN-TL</td>
<td>MEMS</td>
<td>71.71%</td>
</tr>
<tr>
<td>DNN-R</td>
<td>Piezoelectric</td>
<td>80.01%</td>
</tr>
</tbody>
</table>

Table 4: The effect of parameter tuning on the accuracy of DNN-TL model. The parameter tuning gave an absolute gain of 13.71% over the baseline model.

<table>
<thead>
<tr>
<th>Tuning Factor</th>
<th>Accuracy</th>
<th>Tuning Factor</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>58.00%</td>
<td>Feature Selection</td>
<td>64.08%</td>
</tr>
<tr>
<td>Feature-normalization</td>
<td>68.66%</td>
<td>Neurons per layer (50-80-100)</td>
<td>69.41%</td>
</tr>
<tr>
<td>Number of Hidden layers (2-3)</td>
<td>70.32%</td>
<td>Number of Epochs (50-100)</td>
<td>70.75%</td>
</tr>
<tr>
<td>Batch Size (50-100)</td>
<td>71.71%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

done on the plentiful sensor data from the piezoelectric sensor and the prediction was done based on the MEMS sensor data. The comparison between regular DNN model and our transfer-learning DNN model on MEMS sensors in terms of the best achieved accuracy is shown in Table 3. We notice that the transfer-learning model gives a relative gain of 11.6% over the model trained only on the lower resolution MEMS sensor data. The intuition here is that the MEMS sensor data is only 2000 samples, due to very low sampling rate (10 Hz as opposed to 3.2 kHz with the piezoelectric sensor) and thus it cannot fit a good DNN-R model. On the other hand, we can train a DNN-TL model with sensor of different type (but still with vibration readings) with huge data and classify the failure of the sensor under test (i.e., MEMS with less data) with accuracy 71.71%.

Moreover, we show the effect of parameter-tuning on the performance of the models in Table 4. The parameter tuning gives an absolute gain of 13.71% over the baseline DNN-TL model. Delving into the specifics, the most effective tuning steps were feature-selection and normalization which give absolute increase of 10.66% in the accuracy over non-normalized features and increasing number of hidden layers and batch size which gave around 3.05% each on the performance. Note that increasing the epochs to 200 and hidden layers to more than 3 decreases the accuracy, due to over-fitting.

Feature Selection: We validate one idea that the vibration data in certain axis will not carry different information in normal and failure cases. The circular movement around the center of the motor is on X and Z axes so that they have vibration values that change with motor condition while the Y-axis has smaller vibration (the direction of the shaft). Thus, we compare the result of the model when the features are the three data axes in one setup (i.e., default setup) and the proposed idea when the features are extracted only from X-axis and Z-axis data vectors. According to the experimental setup shown in Fig. 2 as the motor rotates with the disk, which is imbalanced by the mounted mass, i.e., eccentric weight, the centripetal forces become unbalanced, and this causes repeated vibrations along multiple directions. Considering the circular movement around the center of the motor, the two directions, which are x-axis and z-axis in our case, are mainly vibrated while the y-axis (the direction along the shaft) show relatively smaller vibration, which may not show a distinguishable variation in the data pattern as machine health varies. We find that this feature selection process gave us a relative increase of 10.5% over the baseline model with all three features. Specifically, the accuracy is 58% using the model trained on default features compared to 64.08% using the model with feature selection. This kind of feature selection requires domain knowledge, specifically about the way the motor vibrates and the relative placement of the sensors. The intuition here is that redundant data features are affecting the model’s learning and therefore selecting the most discriminating features helps the neural network learning.

4.2 Data-augmentation Model Results

Experimental Setup: We used data-augmentation techniques (by both augmenting data from different RPMs and generating samples with interpolation within each RPM) and train DNN-R model on each sensor. For piezoelectric sensor, the data-augmentation model consists of 5M samples (480K samples collected from each rotational speed data for the available ten rotational speeds and 20K generated samples by interpolation within each RPM). For MEMS sensor, the data augmentation model consists of 15120 samples. We compare the average accuracy of the model over all RPMs
under the regular model (DNN-R) and the augmented model. The absolute increase in the accuracy using the augmentation techniques over the regular model is 9.76% for piezoelectric and 8.99% for MEMS, respectively. The data-augmentation techniques are useful for both piezoelectric and MEMS vibration sensors. Data-augmentation is useful for transfer learning across different RPMs.

**Confusion Matrices Comparison:** Here, we show the confusion matrix which compare the performance of our DNN-R models for each operational state separately. Table 5(a) shows such metric using data-augmentation. The best performance is for near-failure which exceeds 96%. This is good in practice since it gives early alarm (or warning) about the expected failure in future. Moreover, the model has good performance in normal operation which exceeds 70%. Finally, the failure accuracy is a little lower which is 61.67% however the confusion is with near-failure state which also gives alarm under such prediction. On the other hand, DNN-R model without data-augmentation has worse prediction in both normal and near-failure modes as shown in Table 5(b) (normal operation detection around 60.4% and near-failure is 93.86%) while much better for detecting failures where the accuracy is 75.00%. The intuition here is that detecting near-failure and normal-operation modes can be enhanced using data-augmentation techniques. On the contrary, detecting failure operational state is better without data-augmentation as failure nature can be specific for each RPM and thus creating single model for each RPM can be useful in that sense.

### 4.3 Relaxation of the Classification Problem

In some applications of the sensor data, the goal can be to detect only if the data from the deployed sensor is normal or not. Thus, we relax the defect classification problem into binary classification problem to test such application. In this subsection, the experimental data obtained under five rotation speeds, i.e., 300, 320, 340, 360, and 380 RPMs were considered to classify between normal and not-normal states. For the deep learning model, we use neural network, which consists of two layers. The models’ performances are summarized in Table 6. Compared to the original defect classification problem, the performance here is better due to the following reasons. First, the confusion is less in binary classification problem since we have only two classes. Second, the variation in the range between RPMs is less in this experiment.

### 4.4 Failure Detection Models

Failure detection Models: There have been several works to study a failure detection in manufacturing processes using single or multi-sensor data [24][40][25]. Specifically, the recent work [24], in which the kernel principal component analysis based anomaly detection system was proposed to detect a cutting tool failure in a machining process. In the study, multi-sensor signals were used to estimate the condition of a cutting tool, but a transfer learning between different sensor types was not considered. Also, in another recent study [25], the fault detection monitoring system was proposed to detect various failures in a DC motor such as a gear defect, misalignment, and looseness. In the study, a single sensor, i.e., accelerometer, was used to obtain machine condition data, and several convolutional neural network architectures were used to detect the targeted failures. However, different rotational speeds and sensors were not considered. Thus, these techniques must be applied

### Table 5: Confusion matrices for classifying operational conditions using DNN-R, where we have the following cases (a) with data-augmentation and (b) no data-augmentation.

<table>
<thead>
<tr>
<th>Normal</th>
<th>Near-failure</th>
<th>Failure</th>
<th>Normal</th>
<th>Near-failure</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM-300</td>
<td>70.22%</td>
<td>28.40%</td>
<td>1.38%</td>
<td>60.38%</td>
<td>18.52%</td>
</tr>
<tr>
<td>RPM-320</td>
<td>96.05%</td>
<td>0.14%</td>
<td>0.08%</td>
<td>93.86%</td>
<td>1.00%</td>
</tr>
<tr>
<td>RPM-340</td>
<td>61.67%</td>
<td>0%</td>
<td>28.00%</td>
<td>75.00%</td>
<td></td>
</tr>
<tr>
<td>RPM-360</td>
<td>97.33%</td>
<td>100%</td>
<td>97.58%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPM-380</td>
<td>99.77%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 6: Comparison on the performance of binary classifier detection model where the trained model is using one RPM and the tested data is from another RPM. The average accuracy is higher, compared to the three classes defect classification models.

<table>
<thead>
<tr>
<th>Trained RPM</th>
<th>RPM-300</th>
<th>RPM-320</th>
<th>RPM-340</th>
<th>RPM-360</th>
<th>RPM-380</th>
<th>Average (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM-300</td>
<td>100%</td>
<td>65.17%</td>
<td>58.17%</td>
<td>51.50%</td>
<td>50.63%</td>
<td>65.09%</td>
</tr>
<tr>
<td>RPM-320</td>
<td>99.75%</td>
<td>100%</td>
<td>97.58%</td>
<td>78.63%</td>
<td>68.17%</td>
<td>88.82%</td>
</tr>
<tr>
<td>RPM-340</td>
<td>96.60%</td>
<td>99.27%</td>
<td>100%</td>
<td>97.33%</td>
<td>82.43%</td>
<td>95.12%</td>
</tr>
<tr>
<td>RPM-360</td>
<td>96.60%</td>
<td>99.27%</td>
<td>99.67%</td>
<td>95.46%</td>
<td>84.43%</td>
<td>95.46%</td>
</tr>
<tr>
<td>RPM-380</td>
<td>61.05%</td>
<td>87.75%</td>
<td>96.93%</td>
<td>99.77%</td>
<td>89.07%</td>
<td>95.46%</td>
</tr>
</tbody>
</table>
again for each new sensor type. On the other hand, we consider the transfer learning between different sensor types. We also compare traditional and ML-based models for our anomaly detection task.

**Learning Transfer:** Transfer learning has been proposed to extract knowledge from one or more source tasks and apply the knowledge to a target task \[50,33,43\] with the advantage of intelligently applying knowledge learned previously to solve new problems faster. In the literature, transfer learning techniques have been applied successfully in many real-world data processing applications, such as cross-domain text classification, constructing informative priors, and large-scale document classification \[28,8\]. However, these works did not tackle the transfer the learning across different instances of sensors that we consider here and did not consider the smart manufacturing domain. In smart manufacturing systems, the existing works only considered calibration of sensors using neural network regression models \[48\] and multi-fault Bearing classification \[44\]. However, these works did not tackle the transfer of the learning across different instances of sensors that we consider here.

### 6 Discussion and Limitations

**Ethical concerns:** We do not see significant risks of security threats or human rights violations in our work or its potential applications. However, we do foresee that our work contributes to the field of smart manufacturing and anomaly detection fields overall. These efforts might eventually automate the detection process, leading to changes in the workforce structure. Hence, there is a general concern that automation may significantly reduce the demand for manufacturing human workers, and the industries would need to act proactively to avoid the social impact of such changes.

**Transfer Learning under Different Features:** In our transfer learning task, the sensor types I and II should be measuring the same physical quantity but can be from different manufacturers and with different characteristics. Another interesting question would be what happens if the two sensor types have overlapping but not identical features in the data that they generate? This requires more complex models which can do feature transformations, using possibly domain knowledge, and we leave such investigation for future work.

**Reproducibility:** We have publicly released our source codes and benchmark data to enable others reproduce our work. We are publicly releasing, with this submission, our smart manufacturing database corpus of 4 datasets. This resource will encourage the community to standardize efforts at benchmarking anomaly detection in this important domain. We encourage the community to expand this resource by contributing their new datasets and models. The website with our database and source codes is: https://drive.google.com/drive/u/2/folders/1QX3chSnSTK03PSh15k8dfWbMvDr1J8. The details of each dataset and the different categories of models are in Section 3. We provide the datasheet for the datasets in supplementary material (Appendix A). The hyper-parameter selections and the libraries used are presented in Appendix E.

### 7 Conclusion

This paper explored several interesting challenges to an important application area, smart manufacturing. We studied anomaly detection and failure classification for the predictive maintenance problem of smart manufacturing. We designed a temporal anomaly detection technique and an efficient defect-type classification technique for such application domain. We compared the traditional and ML-based models for anomaly detection. We observed that ML-based models lead to better anomaly detection prediction. We tested our findings on four real-world data-sets. We then proposed a transfer learning model for classifying failure on sensors with lower sampling rate (MEMs) using learning from sensors with huge data (piezoelectric) where the model can detect anomalies across operating regimes. Our findings indicate that the transfer learning model can considerably increase the accuracy of failure detection. We also studied the effects of several tuning parameters to enhance the failure classification. We release our database corpus and codes for the community to build on it with new datasets and models. Future avenues of research include leveraging the data from multiple sensors and detecting the device health by merging information from multiple, potentially different, sensors.
Acknowledgments and Disclosure of Funding

This work is supported by the Wabash Heartland Innovation Network (WHIN). The opinions expressed in this publication are those of the authors. They do not purport to reflect the opinions or views of sponsor. We thank Nithin Raghunathan and Ali Shakouri for their valuable feedback and suggestions on the work.

References


[38] Gul Shahzad, Heekwon Yang, Arbab Waheed Ahmad, and Chankil Lee. Energy-efficient intelligent street


[40] Sheng-Hsien Gary Teng and Shin-Yann Michael Ho. Failure mode and effects analysis. *International

[41] Tara Elizabeth Thomas, Jinkyu Koo, Somali Chaterji, and Saurabh Bagchi. Minerva: A reinforcement
learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations.
In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages


detection. In *2016 IEEE 30th International Conference on Advanced Information Networking and


Issue on Smart Manufacturing.


[49] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep

[50] Evan Wei Xiang, Bin Cao, Derek Hao Hu, and Qiang Yang. Bridging domains using world wide knowledge

[51] Yuchen Yuan, Yi Shi, Changyang Li, Jinman Kim, Weidong Cai, Zeguang Han, and David Dagan Feng.
Deepgene: an advanced cancer type classifier based on deep learning and somatic point mutations. *BMC
Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default [TODO] to [Yes], [No], or [N/A]. You are strongly encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets?
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The errors are negligible under different random seeds
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]