Generative Adversarial Networks for Failure Prediction

Shuai Zheng, Ahmed Farahat, and Chetan Gupta
Industrial AI Lab, Hitachi America Ltd
Santa Clara, CA, USA
{Shuai.Zheng, Ahmed.Farahat, Chetan.Gupta}@hal.hitachi.com

Abstract. Prognostics and Health Management (PHM) is an emerging engineering discipline which is concerned with the analysis and prediction of equipment health and performance. One of the key challenges in PHM is to accurately predict impending failures in the equipment. In recent years, solutions for failure prediction have evolved from building complex physical models to the use of machine learning algorithms that leverage the data generated by the equipment. However, failure prediction problems pose a set of unique challenges that make direct application of traditional classification and prediction algorithms impractical. These challenges include the highly imbalanced training data, the extremely high cost of collecting more failure samples, and the complexity of the failure patterns. Traditional oversampling techniques will not be able to capture such complexity and accordingly result in overfitting the training data. This paper addresses these challenges by proposing a novel algorithm for failure prediction using Generative Adversarial Networks (GAN-FP). GAN-FP first utilizes two GAN networks to simultaneously generate training samples and build an inference network that can be used to predict failures for new samples. GAN-FP first adopts an info-GAN to generate realistic failure and non-failure samples, and initialize the weights of the first few layers of the inference network. The inference network is then tuned by optimizing a weighted loss objective using only real failure and non-failure samples. The inference network is further tuned using a second GAN whose purpose is to guarantee the consistency between the generated samples and corresponding labels. GAN-FP can be used for other imbalanced classification problems as well. Empirical evaluation on several benchmark datasets demonstrates that GAN-FP significantly outperforms existing approaches, including under-sampling, SMOTE, ADASYN, weighted loss, and infoGAN augmented training.

Keywords: Generative Adversarial Networks · Failure Prediction · Imbalanced Classification

1 Introduction

Reliability of industrial systems, products and equipment is critical not only to manufacturers, operating companies, but also to the entire society. For example,
in 2017, due to electrical fault in a refrigerator, Grenfell Tower fire in London killed 72 people, hospitalized 74 and caused 200 million to 1 billion GBP property damage [1]. Because of the profound impact and extreme costs associated with system failures, methods that can predict and prevent such catastrophes have long been investigated. These methodologies can be grouped under the framework of Prognostics and Health Management (PHM), where prognostics is the process of predicting the future reliability of a product by assessing the extent of deviation or degradation of the product while the product is still working properly; health management is the process of real time measuring and monitoring. The benefits of accurate PHM approaches include: 1) providing advance warning of failures; 2) minimizing unnecessary maintenance, extending maintenance cycles, and maintaining effectiveness through timely repair actions; 3) reducing cost related to inspection and maintenance, reducing cost related to system downtime and inventory by scheduling replacement parts in the right time; and 4) improving the design of future systems [2,3]. Failure prediction is one of the main tasks in PHM.

Failure prediction approaches can be categorized into model-based approaches and data-driven approaches [4]. Model-based approaches use mathematical equations to incorporate a physical understanding of the system, and include both system modeling and physics-of-failure (PoF) modeling. The limitation is that the development of these models requires detailed knowledge of the underlying physical processes that lead to system failure. Furthermore, the physical models are often unable to model environmental interactions. Alternatively, data-driven techniques are gaining popularity. There are several reasons: 1) data-driven methods learn the behavior of the system based on monitored data and can work with incomplete domain knowledge; 2) data-driven methods can learn correlations between parameters and work well in complex systems, such as aircraft engines, HPC systems [5], large manufacturing systems; 3) with the development of IoT systems, large amount of data is being collected in real time, which makes real time monitoring and alerts for PHM possible.

However, data-driven techniques for failure prediction have a set of unique challenges. Firstly, for many systems and components, there is not enough failure examples in the training data. Physical equipment and systems are engineered not to fail and as a result failure data is rare and difficult to collect. Secondly, complex physical systems have multiple failure and degradation modes, often depending upon varying operating conditions. One way to overcome these challenges is to artificially generate failure data such that different failure modes and operating conditions are adequately covered and machine learning models can be learned over this augmented data. Traditionally, oversampling has been used to generate more training samples. However, oversampling cannot capture the complexity of the failure patterns and can easily introduce undesirable noise with overfitting risks due to the limitation of oversampling models. With the successful application of Generative Adversarial Networks (GANs) [6] in other domains, GANs provide a natural way to generate additional data. For example, in computer vision, GANs are used to generate realistic images to improve performance...
in applications, such as, biomedical imaging [7], person re-identification [8] and image enhancement [9]. In addition, GANs have been used to augment classification problems by using semi-supervised learning [10,11] or domain adaptation [12]. However, GAN methods cannot guarantee the consistency of the generated samples and their corresponding labels. For example, infoGAN is claimed to have 5% error rate in generating MNIST digits [13].

In this work, we propose a novel algorithm that utilizes GANs for Failure Prediction (GAN-FP). Compared to existing work, our contributions are:

1. We propose GAN-FP in which three different modules work collaboratively to train an inference network: (1) In one module, realistic failure and non-failure samples are generated using infoGAN. (2) In another module, the weighted loss objective is adopted to train inference network using real failure and non-failure samples. In our design, this inference network shares the weights of the first few layers with the discriminator network of the infoGAN. (3) In the third module, the inference network is further tuned using a second GAN by enforcing consistency between the output of the first GAN and label generated by the inference network.

2. We design a collaborative mini-batch training scheme for GANs and the inference network;

3. We conduct several experiments that show significant improvement over existing approaches according to different evaluation criteria. Through visualization, we verify that GAN-FP generates realistic sensor data, and captures discriminative features of failure and non-failure samples.

4. Failure prediction is the motivation and typical use case of our design. For broader applications, GAN-FP can be applied to other general imbalanced classification problems as well.

2 Background

2.1 Imbalanced classification for failure prediction

Existing approaches to handle imbalanced data can be categorized into two groups: re-sampling (oversampling/undersampling) and cost-sensitive learning. **Re-sampling** method aims to balance the class priors by undersampling the majority non-failure class or oversampling the minority failure class (or both) [14]. Chawla et al. [15] proposed SMOTE oversampling, which generates new synthetic examples from the minority class between the closest neighbors from this class. He et al. [16] proposed ADASYN oversampling, which uses a weighted distribution for different minority class examples according to their level of difficulty in learning. Inspired by the success of boosting algorithms and ensemble learning, re-sampling techniques have been integrated into ensemble learning [14]. **Cost-sensitive learning** assigns higher misclassification costs to the failure class than to the non-failure class [17]. Zhang et al. [18] proposed an evolutionary cost-sensitive deep belief network for imbalanced classification, which uses adaptive differential evolution to optimize the misclassification costs. Using
Fig. 1: InfoGAN architecture: latent code $c$ and noise vector $z$ are combined as input for generator $G$. $x' = G(z, c)$ is generated data, $x$ is real data, discriminator $D$ tries to distinguish generated data from real data, network $Q$ is used to maximize $L_{\text{mutual}}$ (Eq. (2)), loss $L_1$ is given in Eq. (3).

weighted softmax loss function, Jia et al. [19] proposed a framework called Deep Normalized CNN for imbalanced fault classification of machinery to overcome data imbalanced distribution. Many hybrid methods combine both re-sampling and cost-sensitive learning [20]. However, limitations exist in both categories. For instance, oversampling can easily introduce undesirable noise with overfitting risks, and undersampling removes valuable information due to data loss. Cost-sensitive learning requires a good insight into the modified learning algorithms and a precise identification of reasons for failure in mining skewed distributions. Data with highly skewed classes also pose a challenge to traditional discriminant algorithms, such as subspace and feature representation learning [21,22,23,24,25,26]. This makes it further difficult to achieve ideal classification accuracy in failure prediction tasks.

2.2 GAN

Recently, generative models such as Generative Adversarial Networks (GANs) have attracted a lot of interest from researchers and industrial practitioners. Goodfellow et al. formulated GAN into a minimax two-player game, where they simultaneously train two models: a generator network $G$ that captures the data distribution, and a discriminator network $D$ that estimates the probability that a sample comes from the true data rather than the generator network $G$. The goal is to learn the generator distribution $p(x')$ of $G$ over data $x'$ that matches the real data distribution $p(x)$. The generator network $G$ generates samples by transforming a noise vector $z \sim p(z)$ into a generated sample $G(z)$. This generator is trained by playing against the discriminator network $D$ that aims to distinguish between samples from true data distribution $p(x)$ and the generator distribution $p(x')$. Formally, the minimax game is given by the following expression:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log (1 - D(G(z)))].$$ (1)

InfoGAN [13] is an information-theoretic extension to GAN. InfoGAN decomposes the input noise vector into two parts: incompressible noise vector $z$ and latent code vector $c$. The latent code vector $c$ targets the salient structured
GANs for Failure Prediction

3 GAN-FP: GAN for Failure Prediction

3.1 Motivation

In failure prediction problems, we collect a lot of training data $\mathbf{x}$ and the corresponding labels $y$. Training data $\mathbf{x}$ usually is sensor data coming from equipment,
but can be image, acoustics data as well. Label $y$ contains a lot of non-failure label 0s and very few failure label 1s.

Given a failure prediction problem, one choice is to construct a deep inference neural network and adopt weighted loss objective. As there are not enough real failure samples, test samples with failure labels are often misclassified to the prevalent non-failure class. As mentioned earlier, in this work, we propose the use of GANs to generate realistic failure samples.

To control the class labels of generated samples, we can choose Conditional GAN (CGAN) or infoGAN. CGAN was shown to mainly capture class-level features [13,28]. In addition to capturing class-level features, infoGAN captures fine variations of features that are continuous in nature using continuous latent code. As mentioned earlier, PHM data has multiple failure modes, and is continuous in nature, hence we use infoGAN as a basic building block in our design.

One problem with simply using infoGAN, is that it cannot guarantee that the generated sample is from a desired class. This means that some generated samples might end up having the wrong label. For example, infoGAN is claimed to have 5% error rate in generating MNIST digits [13]. When we have a 2-class highly imbalanced classification problem like failure prediction, this can have significant negative impact on the usefulness of this approach. In order to alleviate this problem, we propose the use of a second GAN to enforce the consistency of data-label pairs. In the second GAN, we use the inference network $P$ as a label generator.

Once we have generated data, a traditional approach is to use both the generated and real samples to train a classifier. However, since we are sharing layers between the inference network and the discriminator network in the first GAN, and training all three modules simultaneously, we can directly use this inference network to achieve higher inference accuracy.

During building the model, we alternate between the following steps:

1. Update the infoGAN to generate realistic samples for failure and non-failure labels.
2. Update the inference network $P$ using real data. We bootstrap $P$ using the weights of the first few layers of the discriminator of the infoGAN. This is a common approach to save training time and utilize the ability of GAN to extract features.
3. Update inference network $P$ along with the discriminator in the second GAN to make sure that the generated samples and corresponding labels are consistent. This will increase the discriminative power of inference network $P$.

3.2 GAN-FP model

Figure 3 shows the design of GAN-FP.

Module 1 adopts an infoGAN to generate class-balanced samples. For the input categorical latent code $c$, we randomly generate labels 0s (non-failure) and 1s (failure) with equal probability. The continuous latent code $c$ and noise vector $z$ is generated using uniform random process within range $[0, 1]$. Generator
network $G$ is a deep neural network with input $(z, c)$, and outputs generated sample $x'$, where $x'$ has the same size as real data $x$. Discriminator network $D$ aims to distinguish generated sample $x'$ from real sample $x$. Network $Q$ aims to maximize the mutual information between latent code $c$ and generated sample $x'$. By jointly training network $G$, $D$ and $Q$, module 1 solves the minimax problem denoted in Eq. (3). The first few layers of the discriminative layer $D + Q$ will capture a lot of implicit features about the data. In order to reduce the overall training time, we are going to reuse these weights while training the inference network in the Module 2.

Module 2 consists of a deep neural network $P$ and solves a binary classification problem with weighted loss based on real data and real label. Network $P$ shares the first several layers with $D$ and takes as input real data $x$ and outputs a probability (denoted as $P(x)$) within range $[0, 1]$ indicating the chance that $x$ is a failure sample. The real label is denoted as $y$ (0 or 1). In our design, the loss function $L_2$ for module 2 is cross entropy:

$$
\min_P L_2(P) = \mathbb{E}_{(x, y) \sim p(x, y)}[-wy \log(P(x)) - (1 - y) \log(1 - P(x))],
$$

(5)

where weight $w = \frac{\text{number of non-failure samples}}{\text{number of failure samples}} > 1$. Note at this step, the input for network $P$ is class-imbalanced real data and labels. Loss $L_2$ is a weighted version which emphasizes more on failure sample prediction. In the training of Module 3, the weights of inference network $P$ will be further tuned using generated data and labels.
Algorithm 1 Mini-batch SGD solving GAN-FP.

**Input:** Real data and label pairs \(\{x_i, y_i\}\), where \(i = 1, 2, ..., n\), hyperparameter \(\lambda_G, \lambda_D, \lambda_P, \lambda_{D_2}, \lambda_{L_2}\), batch size \(b\).

**Output:** Network parameters \(\theta_G, \theta_D, \theta_Q, \theta_P, \theta_{D_2}\) for networks \(G, D, Q, P, D_2\) respectively.

1. Initialize \(\theta_G, \theta_D, \theta_Q, \theta_P, \theta_{D_2}\).
2. repeat
   3. Randomly choose \(b\) data and label pairs from \(\{x_i, y_i\}\).
   4. Randomly generate \(b\) latent code \(c\) and noise \(z\), where \(c\) is class-balanced, noise \(z\) is uniform random variables.
   5. Update Module 1 discriminator network \(\theta_D\) by ascending along its stochastic gradient w.r.t. max \(D\lambda_D L_1(D)\) and share the weights of the first few layers with \(P\).
   6. Update Module 1 generator and \(Q\)-network \(\theta_G, \theta_Q\) by descending along its stochastic gradient w.r.t. min \(G,Q\lambda_G L_1(G, Q)\).
   7. Update inference network \(\theta_P\) by descending along its stochastic gradient w.r.t. min \(P\lambda_P L_2(P)\) and use \(P\) as the generator of Module 3.
   8. Update Module 3 discriminator network \(\theta_{D_2}\) by ascending along its stochastic gradient w.r.t. max \(\lambda_{D_2} L_3(D_2)\).
   9. Update Module 3 generator network \(\theta_P\) by descending along its stochastic gradient w.r.t. min \(\lambda_P P\).
3. until Convergence

**Module 3** consists of network \(P\) and \(D_2\) and enforces generated data-label pair \((x', y')\) to look like real data-label pair \((x, y)\). \(P\) serves as the generator network. Given \(x'\), the generated label \(y' = P(x')\) needs to be as correct as possible. \(D_2\) tries to distinguish the generated data-label pair from real pair. The minimax objective for module 3 is given as:

\[
\min_P \max_{D_2} L_3(P, D_2) = \mathbb{E}_{(x, y) \sim p(x, y)} [\log D_2(x, y)] \\
+ \mathbb{E}_{x' \sim p(x')} [\log (1 - D_2([x', P(x')]))]. \tag{6}
\]

While training this module, the weights of the inference network \(P\) will be further tuned to increase the discrimination between failure and non-failure labels. The effectiveness of Module 3 to improve inference network \(P\) will be validated by comparing the performance of GAN-FP with infoGAN augmented training (denoted as InfoGAN AUG in experiments), where we train the inference network \(P\) with generated data without using Module 3.

### 3.3 Algorithm

Algorithm 1 summarizes the procedure for training GAN-FP. The input data includes real data-label pairs \((x_i, y_i)\), where \(i = 1, 2, ..., n\), and hyperparameter \(\lambda_G, \lambda_D, \lambda_P, \lambda_{D_2}, \lambda_{L_2}\), which control the weights of different losses, as in traditional regularization approaches [29,30]. The output of this algorithm is the trained neural network parameters \(\theta_G, \theta_D, \theta_Q, \theta_P, \theta_{D_2}\) for network \(G, D, Q, P, D_2\).
D₂ respectively. Step 1 initializes network parameters. Then we run the mini-batch loop until \( L₁ \), \( L₂ \) and \( L₃ \) converge. In each mini-batch loop, Step 3 first randomly chooses a batch of real data-label pairs. Step 4 generates batch size of latent code \( c \) and \( z \). Step 5 to 9 update all 3 modules.

4 Visualization of generated samples

We take CMAPSS FD001 data as an example to visualize the generated samples and examine if the proposed GAN-FP can generate realistic enough fake samples for failure prediction task. CMAPSS FD001 data contains failure and non-failure data for turbofan engines. Each engine sample includes 21 sensors and their readings are in a continuous time window with 15 time steps. Detailed description of CMAPSS FD001 is given in Section 5.1. Due to space limitations, we chose 4 sensors (s1, s2, s3, s4) and plotted them from real samples and generated samples. In Figure 4, we visualize 2 real failure samples and 2 real non-failure samples. As we can see, for failure samples, sensor s1 has higher values than other sensors, sensor s2 and s3 have lower values than s1 and s4, especially for time from 6 to 14; for non-failure samples, sensor s4 has lower values, sensor s2 and s3 have higher values than sensor s1 and s4. In Figure 5, we visualize the same 4 sensors from 2 generated failure samples and 2 generated non-failure samples. We observe similar visual properties as in Figure 4 for failure samples, sensor s2 and s3 have lower values than s1 and s4, especially for time from 6 to 14; for non-failure samples, sensor s2 and s3 have higher values than sensor s1 and s4. We also observe that noises exist in generated samples. For example, at time
Table 1: Data sets.

| Name               | Dimension | Failure sample # | Non-failure sample # | Operating condition # | Fault condition # |
|--------------------|-----------|------------------|----------------------|-----------------------|-------------------|
| APS                | 170       | 1,000            | 59,000               | N/A                   | N/A               |
| CMAPSS FD001       | 315       | 2,000            | 12,031               | 1                     | 1                 |
| CMAPSS FD002       | 315       | 5,200            | 31,399               | 6                     | 1                 |
| CMAPSS FD003       | 315       | 2,000            | 16,210               | 1                     | 2                 |
| CMAPSS FD004       | 315       | 4,980            | 39,835               | 6                     | 2                 |

Table 2: Network structures.

| Network | APS | CMAPSS |
|---------|-----|--------|
| G       | 64, 64, 170 | 64, 256, 500, 500, 315 |
| D       | 170, 64, 1 | 315, 500, 500, 256, 1 |
| Q       | 170, 64, 1 | 315, 500, 500, 256, 1 |
| P       | 170, 64, 1 | 315, 500, 500, 256, 1 |
| $D_2$   | 170, 64, 1 | 315, 500, 500, 256, 1 |

11, sensor s4 in Figure [5b] has a big drop, but at time 12, s4 increases back to a higher value. Though real sensor data seems more smooth than generated sensor data, GAN-FP is able to capture the major properties for this failure prediction task. This shows that GAN-FP can generate very good failure and non-failure samples and different levels of variations exist in the generated samples.

5 Experiments

5.1 Data

We conduct experiments on one Air Pressure System (APS) data set from trucks [31] and four turbofan engine degradation data sets from NASA CMAPSS (Commercial Modular Aero-Propulsion System Simulation) [32]. For APS data, air pressure system generates pressured air that are utilized in various functions in a truck, such as braking and gear changes. The failure class consists of component failures for a specific component of the APS system. The non-failure class consists of samples not related to APS failures. The CMAPSS data consists of four subsets: FD001, FD002, FD003, and FD0004. Data attributes are summarized in Table 1 in each subset, the data records a snapshot of turbofan engine sensor data at each time cycle, which includes 26 columns: 1st column represents engine ID, 2nd column represents the current operational cycle number, 3-5 columns are the three operational settings that have a substantial effect on engine performance, 6-26 columns represent the 21 sensor values. The engine is operating normally at the start of each time series, and develops a fault at some point in time. The fault grows in magnitude until a system failure. The four CMAPSS data sets have different number of operating conditions and fault conditions. For example, FD001 data has one operating condition and one fault condition, and FD002 has six operating conditions and one fault condition. CMAPSS data set is considered benchmark for predictive maintenance [33].
5.2 Experimental setup and evaluation criteria

We use fully connected layers for all the networks. Table 2 shows the network structures for both APS and CMAPSS data. For example, G network for APS data consists of 3 layers, with the first layer 64 nodes, second layer 64 nodes and last layer 170 nodes. For APS data, network Q and P share the first two layers with network D. For CMAPSS data, network Q and P share the first four layers with network D. For both APS and CMAPSS data, the noise vector $z$ is a 60-dimensional vector with uniform random values within $[0, 1]$. Latent code $c$ includes 1-dimensional categorical code and 3-dimensional continuous code with uniform random values within $[0, 1]$. The activation function is rectified linear unit by default.

For evaluation, we use AUC (Area Under Curve), (precision, recall, F1) with macro average, micro average, and for the failure class only. All compared methods output the probability that a sample is a failure sample. We then compute the precision and recall curve and calculate AUC. We then can compute both failure and non-failure class precision, recall and F1. Larger values indicate better performance in all these metrics. More about these metrics can be found in [34].

We compare GAN-FP with 17 other methods. For the first 16 methods, we conduct experiments using 4 classifiers in 4 different sampling settings. The 4 classifiers are DNN, SVM (Support Vector Machines), RF (Random Forests) and DT (Decision Trees). The 4 sampling settings are: undersampling, weighted loss, SMOTE oversampling and ADASYN oversampling. The structure of DNN is the same as network $P$ in GAN-FP. Parameters of SVM, RF and DT are tuned to achieve the best accuracy. For undersampling, we fix the failure samples and randomly draw equal size number of non-failure samples for training. For each experiment, we perform 10 times random undersampling. For weighted loss objective, we assign the same class weights as used in Eq.(5). Lastly, we compare with infoGAN augmented DNN (denoted as InfoGAN AUG), which uses infoGAN to generate more failure samples to make the class distribution balanced and then train a DNN for classification. InfoGAN AUG is used to validate the effectiveness of Module 3. Experiments were performed in a 5-fold cross-validation fashion.

5.3 Algorithm convergence

To examine the training convergence, we take CMAPSS FD001 data as an example and plot the loss changes along the training process. From Eq.(3), we know that Module 1 loss consists of three parts: discriminator ($D$) loss $\min_D -V(D)$, generator ($G$) loss $\min_G V(G)$ and mutual information ($Q$) loss $\min_{G,Q} -L_{\text{mutual}}(G,Q)$. Figure 6a shows that D loss and G loss converge along the training. Mutual information ($Q$) loss is minimized and converged after about 2,000 batches. Advanced accelerating algorithm can reduce training time furthermore [35]. Figure 6b shows that $L_2$ loss Eq.(5) is decreasing from 0 to 3,000 batches. Figure 6c shows that $D_2$ loss and generator P loss of Module 3 converge along the training. Overall, this shows the effectiveness of Algorithm 1.
5.4 Effect of class imbalance

We compare the classification performance of different approaches when the number of majority non-failure samples is decreased. Figure 7 shows that, for
### Table 4: CMAPSS FD001.

| Method   | Precision | Recall | F1 | Macro Precision | Recall | F1 | Micro Precision | Recall | F1 | Failure Precision | Recall | F1 |
|----------|-----------|--------|----|-----------------|--------|----|-----------------|--------|----|------------------|--------|----|
| **DNN**  | 0.6381    | 0.7525 | 0.7895 | 0.8785 | 0.8785 | 0.8785 | 0.5624 | 0.0650 | 0.6094 |
| Undersampling | 0.6331 | 0.7592 | 0.7200 | 0.8824 | 0.8824 | 0.8824 | 0.5825 | 0.0175 | 0.5995 |
| Weighted loss | 0.6498 | 0.7485 | 0.7972 | 0.8757 | 0.8757 | 0.8757 | 0.5511 | 0.0875 | 0.6118 |
| SMOTE    | 0.6295    | 0.7491 | 0.7822 | 0.8767 | 0.8767 | 0.8767 | 0.5579 | 0.6500 | 0.6905 |
| ADASYN   | 0.6224    | 0.7461 | 0.7893 | 0.8746 | 0.8746 | 0.8746 | 0.5492 | 0.6700 | 0.6936 |
| **SVM**  | 0.6030    | 0.7327 | 0.7614 | 0.8678 | 0.8678 | 0.8678 | 0.5315 | 0.0125 | 0.5601 |
| Undersampling | 0.5531 | 0.7046 | 0.7466 | 0.8497 | 0.8497 | 0.8497 | 0.4782 | 0.6025 | 0.5332 |
| Weighted loss | 0.5378 | 0.7067 | 0.7304 | 0.8507 | 0.8507 | 0.8507 | 0.4813 | 0.6100 | 0.5380 |
| SMOTE    | 0.5701    | 0.7486 | 0.7355 | 0.8771 | 0.8771 | 0.8771 | 0.5733 | 0.5733 | 0.5733 |
| ADASYN   | 0.5238    | 0.7322 | 0.7134 | 0.8606 | 0.8606 | 0.8606 | 0.5470 | 0.4950 | 0.5197 |
| **RF**   | 0.5503    | 0.6996 | 0.7501 | 0.8193 | 0.8193 | 0.8193 | 0.4601 | 0.6279 | 0.5363 |
| Undersampling | 0.5212 | 0.6888 | 0.7057 | 0.8336 | 0.8336 | 0.8336 | 0.4200 | 0.4400 | 0.4298 |
| Weighted loss | 0.5199 | 0.6869 | 0.7021 | 0.8151 | 0.8151 | 0.8151 | 0.3758 | 0.4500 | 0.4096 |
| SMOTE    | 0.5377    | 0.6910 | 0.7666 | 0.8331 | 0.8331 | 0.8331 | 0.4427 | 0.6760 | 0.5350 |
| ADASYN   | 0.4471    | 0.6373 | 0.6596 | 0.8130 | 0.8130 | 0.8130 | 0.3701 | 0.4450 | 0.4041 |
| **DT**   | 0.5429    | 0.6693 | 0.7097 | 0.8497 | 0.8497 | 0.8497 | 0.4782 | 0.6025 | 0.5332 |
| Undersampling | 0.4969 | 0.6631 | 0.6965 | 0.8336 | 0.8336 | 0.8336 | 0.4200 | 0.4400 | 0.4298 |
| Weighted loss | 0.4521 | 0.6460 | 0.6629 | 0.8151 | 0.8151 | 0.8151 | 0.3758 | 0.4500 | 0.4096 |
| SMOTE    | 0.4471    | 0.6373 | 0.6596 | 0.8130 | 0.8130 | 0.8130 | 0.3701 | 0.4450 | 0.4041 |
| ADASYN   | 0.4471    | 0.6373 | 0.6596 | 0.8130 | 0.8130 | 0.8130 | 0.3701 | 0.4450 | 0.4041 |
| **InfoGAN AUG** | 0.5484 | 0.6945 | 0.7455 | 0.6583 | 0.6583 | 0.6583 | 0.4200 | 0.4400 | 0.4298 |
| **GAN-FP** | 0.5666 | 0.7081 | 0.7485 | 0.8746 | 0.8746 | 0.8746 | 0.5492 | 0.6700 | 0.6936 |

CMAPSS FD001, there is no significant performance loss when the number of majority non-failure samples is decreased. Among all experiments, GAN-FP gives the best performance in terms of the four metrics.

### 5.5 Comparison with other methods

Table 3 shows the result for APS data. Table 4, 5, 6 and 7 show the results for CMAPSS. The best performing methods in each column is in bold. Note that CMAPSS data sets have different levels of difficulty since they have different number of operating conditions and fault conditions. Overall, GAN-FP shows...
### Table 6: CMAPSS FD003.

| Method   | DNN | SVM | RF  | DT  | InfoGAN AUG | GAN-FP |
|----------|-----|-----|-----|-----|-------------|--------|
| AUC      |     |     |     |     |             |        |
| Macro    |     |     |     |     |             |        |
| Micro    |     |     |     |     |             |        |
| Failure  |     |     |     |     |             |        |
| Precision|     |     |     |     |             |        |
| Recall   |     |     |     |     |             |        |
| F1       |     |     |     |     |             |        |

| Undersampling | 0.6211 | 0.5718 | 0.5152 | 0.5460 | 0.4826 | 0.7093 |
| Weighted loss | 0.6035 | 0.6338 | 0.5693 | 0.5460 | 0.5065 | 0.6538 |
| SMOTE       | 0.6207 | 0.6383 | 0.5266 | 0.4947 | 0.5360 | 0.6161 |
| ADASYN      | 0.6199 | 0.6113 | 0.5246 | 0.4685 | 0.4364 | 0.5703 |

### Table 7: CMAPSS FD004.

| Method   | DNN | SVM | RF  | DT  | InfoGAN AUG | GAN-FP |
|----------|-----|-----|-----|-----|-------------|--------|
| AUC      |     |     |     |     |             |        |
| Macro    |     |     |     |     |             |        |
| Micro    |     |     |     |     |             |        |
| Failure  |     |     |     |     |             |        |
| Precision|     |     |     |     |             |        |
| Recall   |     |     |     |     |             |        |
| F1       |     |     |     |     |             |        |

| Undersampling | 0.4826 | 0.4553 | 0.5152 | 0.4885 | 0.4826 | 0.7093 |
| Weighted loss | 0.5065 | 0.5453 | 0.5693 | 0.5460 | 0.5065 | 0.6538 |
| SMOTE       | 0.5360 | 0.4983 | 0.4978 | 0.4947 | 0.5360 | 0.6161 |
| ADASYN      | 0.5398 | 0.4983 | 0.4978 | 0.4947 | 0.5398 | 0.6161 |

better results in terms of AUC and F1 score compared to its counterparts. The fact that GAN-FP outperforms InfoGAN AUG shows the effectiveness of Module 3.

### 6 Conclusion

In conclusion, we proposed a novel model GAN-FP for imbalanced classification and failure prediction, and experimented it on industrial data. This novel design not only improves modeling performance, but also can have significant potential economical and social values.
References

1. Monaghan, A.: Hotpoint tells customers to check fridge-freezers after grenfell tower fire. The Guardian (2017)
2. Vichare, N.M., Pecht, M.G.: Prognostics and health management of electronics. IEEE transactions on components and packaging technologies 29(1), 222–229 (2006)
3. Mosallam, A., Medjaher, K., Zerhouni, N.: Data-driven prognostic method based on bayesian approaches for direct remaining useful life prediction. Journal of Intelligent Manufacturing 27(5), 1037–1048 (2016)
4. Pecht, M.G.: A prognostics and health management roadmap for information and electronics-rich systems. IEICE ESS Fundamentals Review 3(4), 425–432 (2010)
5. Zheng, S., Shaq, Z.Y., Zhang, X., Jamjoom, H., Fong, L.: Analysis and modeling of social influence in high performance computing workloads. In: European Conference on Parallel Processing. pp. 193–204. Springer Berlin Heidelberg (2011)
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
7. Calimeri, F., Marzullo, A., Stabile, C., Terracina, G.: Biomedical data augmentation using generative adversarial neural networks. In: International Conference on Artificial Neural Networks. pp. 626–634. Springer (2017)
8. Zhong, Z., Zheng, L., Zheng, Z., Li, S., Yang, Y.: Camera style adaptation for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5157–5166 (2018)
9. Yun, K., Bustos, J., Lu, T.: Predicting rapid fire growth (flashover) using conditional generative adversarial networks. Electronic Imaging 2018(9), 1–4 (2018)
10. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.R.: Good semi-supervised learning that requires a bad gan. In: Advances in Neural Information Processing Systems. pp. 6510–6520 (2017)
11. Tran, T., Phan, T., Carneiro, G., Palmer, L., Reid, I.: A bayesian data augmentation approach for learning deep models. In: Advances in Neural Information Processing Systems. pp. 2707–2806 (2017)
12. Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 1, p. 7 (2017)
13. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems. pp. 2172–2180 (2016)
14. Nejadzian, S., Parvin, H., Faraji, E.: Using sub-sampling and ensemble clustering techniques to improve performance of imbalanced classification. Neurocomputing 276, 55–66 (2018)
15. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research 16, 321–357 (2002)
16. He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). pp. 1322–1328. IEEE (2008)
17. Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3982–3991 (2015)
18. Zhang, C., Tan, K.C., Li, H., Hong, G.S.: A cost-sensitive deep belief network for imbalanced classification. IEEE Transactions on Neural Networks and Learning Systems (2018)
19. Jia, F., Lei, Y., Lu, N., Xing, S.: Deep normalized convolutional neural network for imbalanced fault classification of machinery and its understanding via visualization. Mechanical Systems and Signal Processing 110, 349–367 (2018)
20. Tang, Y., Zhang, Y.Q., Chawla, N.V., Krasser, S.: Svms modeling for highly imbalanced classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39(1), 281–288 (2009)
21. Zheng, S., Ding, C.: Kernel alignment inspired linear discriminant analysis. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 401–416. Springer Berlin Heidelberg (2014)
22. Zheng, S., Cai, X., Ding, C.H., Nie, F., Huang, H.: A closed form solution to multi-view low-rank regression. In: AAAI. pp. 1973–1979 (2015)
23. Zheng, S., Nie, F., Ding, C., Huang, H.: A harmonic mean linear discriminant analysis for robust image classification. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI). pp. 402–409. IEEE (2016)
24. Zheng, S.: Machine Learning: Several Advances in Linear Discriminant Analysis, Multi-View Regression and Support Vector Machine. Ph.D. thesis, The University of Texas at Arlington (2017)
25. Zheng, S., Ding, C., Nie, F., Huang, H.: Harmonic mean linear discriminant analysis. IEEE Transactions on Knowledge and Data Engineering (2018). https://doi.org/10.1109/TKDE.2018.2861858
26. Zheng, S., Ding, C.: Sparse classification using group matching pursuit. Neurocomputing 338, 83–91 (2019). https://doi.org/10.1016/j.neucom.2019.02.001
27. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
28. Lee, M., Seok, J.: Controllable generative adversarial network. arXiv preprint arXiv:1708.00598 (2017)
29. Zheng, S., Ding, C., Nie, F.: Regularized singular value decomposition and application to recommender system. arXiv preprint arXiv:1804.05090 (2018)
30. Zheng, S., Ding, C.: Minimal support vector machine. arXiv preprint arXiv:1804.02370 (2018)
31. Dua, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
32. Saxena, A., Goebel, K., Simon, D., Eklund, N.: Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 international conference on prognostics and health management. pp. 1–9. IEEE (2008)
33. Zheng, S., Ristovski, K., Farahat, A., Gupta, C.: Long short-term memory network for remaining useful life estimation. In: Prognostics and Health Management (ICPHM), 2017 IEEE International Conference on. pp. 88–95. IEEE (2017)
34. Han, J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)
35. Zheng, S., Vishnu, A., Ding, C.: Accelerating deep learning with shrinkage and recall. In: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). pp. 963–970. IEEE (2016)