Gated recurrent unit autoencoder for fault detection in penicillin fermentation process.

PETROVSKI, A., ARIFEEN, M. and PETROVSKI, S.

2023

Gated Recurrent Unit Autoencoder for Fault Detection in Penicillin Fermentation Process

Andrei Petrovski[1(✉)], Murshedul Arifeen[2], and Sergei Petrovski[1]

[1] Samara State Technical University, Samara, Russia
ap45ap@gmail.com, petrovski.sv@samgtu.ru
[2] School of Computing, Robert Gordon University, Aberdeen, Scotland
d.arifeen@rgu.ac.uk

Abstract. The penicillin fermentation process is a fed-batch system to generate industrial-scale penicillin for antibiotic production. Any fault in the fermentation tank can lead to low-quality penicillin products, which may cause a severe impact on final antibiotic production. In this paper, we have developed a Gated Recurrent Unit-based Autoencoder deep learning model to detect faults in the batch data of the penicillin fermentation process. In particular, we have used the data shuffling strategy to minimize distribution discrepancy from different batches generated under various controlling conditions for training the deep learning model. We have also compared the model with the Feedforward Autoencoder and Long short-term memory Autoencoder model for fault detection. Experimental results show that our model trained on shuffled data from different batches outperformed the Feedforward and Long short-term memory Autoencoder model with an avergae fault detection rate of 94.74%.

Keywords: Fault detection · Autoencoder · Gated recurrent unit · Penicillin fermentation

1 Introduction

Penicillin produced from penicillin fungi is an effective antibiotic to treat various contagious diseases. This antibiotic is one of the earliest discoveries in clinical history by Alexander Flemming in 1928 [11]. It is the secondary metabolite of penicillin fungi used to cure infectious diseases from the early days [1]. The fermentation process of penicillin is a non-linear dynamic fed-batch process that involves three stages of antibiotic production [1,12]. The first stage is known as the growth stage of penicillin fungi cells, where the nutrients of the feeding materials are continuously smashed, and new cells are synthesized. In the second stage, penicillin synthesis starts, and the production of penicillin antibiotics begins. In the final autolysis stage, the pH of the fermentation broth increases, which reduces the capability of penicillin synthesis [1]. The complete fermentation process is controlled through numerous process variables, for instance, pH, temperature, substrate concentration, oxygen or nitrogen concentration, etc [11,12]. Any deviation or faults in process variables can significantly affect the quality of the

antibiotic generated from the penicillin fermentation tank [1]. Moreover, the low-quality product can directly affect the intended disease treatment procedure. Therefore, detecting and preventing faults in the fermentation process is highly required for enhancing quality and reliability of the penicillin product.

Fault detection methods can be categorized as mechanism-driven and data-driven approaches [14,17]. The efficacy of data driven approaches over the mechanism based models is that the data driven models do not require prior knowledge of the underlying examining system [17]. On the contrary, mechanisms or mathematical models require domain knowledge, and acquiring domain knowledge is critical for a non-linear dynamic process like the penicillin fermentation process. However, IoT or sensor networks integrated with distributed control systems can provide data samples related to control variables to build data-driven models [5,17]. Data-driven approaches can be further classified as statistical and neural network-based deep learning methods [18]. Statistical models like principal component analysis (PCA), independent component analysis (ICA), partial least square regression (PLS), or support vector machines are suitable for the dataset with linear patterns [17,18]. However, penicillin fermentation is a complex, non-linear, and time-varying system. Due to the dynamic nature of the system, statistical approaches will fail to model the control parameters and detect faults. On the contrary, the advancements in deep learning can help model non-linear and dynamic control variables of the deep tank penicillin fermentation process [6]. Deep learning models can be supervised and unsupervised based on the data pattern.

Although supervised deep learning models are superior to statistical models to classify normal and faulty conditions for a highly non-linear system like the penicillin fermentation process, these supervised deep learning models require a large dataset comprising normal and faulty samples to learn the patterns [16]. However, abnormal operating conditions are rare in a real industrial environment, and generating anomalous samples is not cost-effective. Therefore, unsupervised modeling techniques trained on only normal data are suitable for detecting and alerting faults in complex non-linear industrial plants. Moreover, due to the varying operating conditions of the penicillin fed-batch fermentation process, the batches of dataset from different conditions vary in the probability distribution. The dynamic distribution nature of the batches creates difficulty in training a deep learning model since a deep learning model assumes that the data samples of a training dataset come from a single source with certain distribution parameters [6,14]. Therefore, it is required to minimize the distribution discrepancy among the data samples of different batches before preparing the training data for a deep learning model [14]. In this work, we have used the data shuffling strategy to shuffle the data samples from different batches of varying distribution to prepare the training data, which minimizes the distribution discrepancy among the samples of the training dataset. The shuffled training data is then used to train a Gated Recurrent Unit (GRU) based deep Autoencoder(AE)(GRU-AE) for fault detection.

1. We have proposed a GRU-AE for fault detection in penicillin fermentation process by shuffling the data samples from different batches. The GRU-AE is compared with Long short term memory based Autoencoder (LSTM-AE) and Feedforward Autoencoder (FF-AE) to detect faults.
2. We have randomly shuffled the data instances from multiple batches generated under different normal operating conditions to reduce the distribution discrepancy of the training dataset.
3. The trained GRU-AE on the shuffled training dataset is tested on ten different faulty batches.
4. Experimental findings show that the GRU-AE model performed better than conventional GRU-AE (trained without data shuffling), LSTM-AE and FF-AE with randomly sampled data samples from different batches of varying operating conditions.

Section 2 discusses recent literature on fault detection based on domain adaptation. Section 4 presents the GRU based AE model and fault detection system. In Sect. 5, we have described the experimental settings and outcomes of the experiments and finally, Sect. 6 concludes the paper.

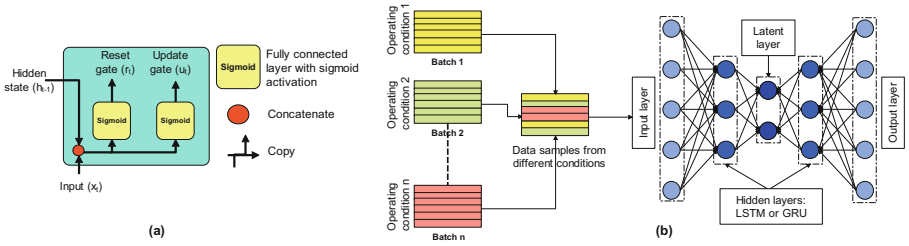

Fig. 1. (a)– illustrates a GRU cell with internal architecture. (b)– shows the data shuffling process and the architecture of an AE model.

2 Literature Review

A novel deep adversarial perceptual domain adaptation method is proposed in [19] for bearing fault detection and diagnosis in the industrial manufacturing system. This novel method overcomes the equilibrium issues of adversarial domain adaptation of the deep learning model training process. A novel perceptual loss function is proposed to force the source and target domain to have the same probability distribution. Maximum mean discrepancy loss is used to build a deep learning-based domain adaptation method in the semiconductor industry [6]. A deep Convolutional Neural network (CNN) is used to extract features and classify the fault conditions. On the contrary, maximum mean discrepancy loss is used to optimize data distributions among

different sets. A new deep trans-fer learning-based adaptive joint distribution adaptation method is proposed for domain adaptive fault detection and diagnosis in several application scenarios [13]. A CNN-based feature extractor is built to extract discriminative features from the process data. A gradient penalty factor is used to make the model converge easily. The experimental findings show that the model can easily adapt to varying work conditions.

3 Domain Adaptation

The performance of the fault detection models based on machine learning depends on the training dataset. If the datasets are imperfect, then the performance of the models degrades. Domain shift is a problem in machine learning, where a model trained on one dataset may not perform well on a slightly different (like change in a probability distribution) dataset. Domain adaptation is a technique to make the models capable to perform well in different datasets [7]. In this work, data shuffling is used to overcome the domain shift problem in various batches of penicillin fermentation dataset.

4 Deep Learning Models

4.1 Gated Recurrent Unit

Gated Recurrent Unit (GRU) [2] is an enhanced variant of Recurrent Neural Network (RNN), which comprises a reset gate and an update gate (Fig. 1). These two gates control the flow of information throughout the unit using a sigmoid activation function. The reset gate controls how much information from the previous state needs to be remembered in the current state. On the contrary, the update gate controls how much information about the new state is a copy of the old state. The two gates are formed using fully connected layers with sigmoid activation. For timestamp t, lets consider the input is $x_t \in \mathbb{R}^{n \times m}$ (n = number of data samples, m = number of features) and the hidden state from previous time step is $h_{t-1} \in \mathbb{R}^{n \times h}$ (h = number of hidden features). Then, we can compute the reset gate ($r_t \in \mathbb{R}^{n \times h}$) and update gate ($u_t \in \mathbb{R}^{n \times h}$) as follows-

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r)$$
$$u_t = \sigma(x_t W_{xu} + h_{t-1} W_{hu} + b_u)$$

where $W_{xr}, W_{xu}, W_{hr}, W_{hu}$ denote weights of the corresponding layers and b_r, b_u are the biases to the reset and update gate respectively.

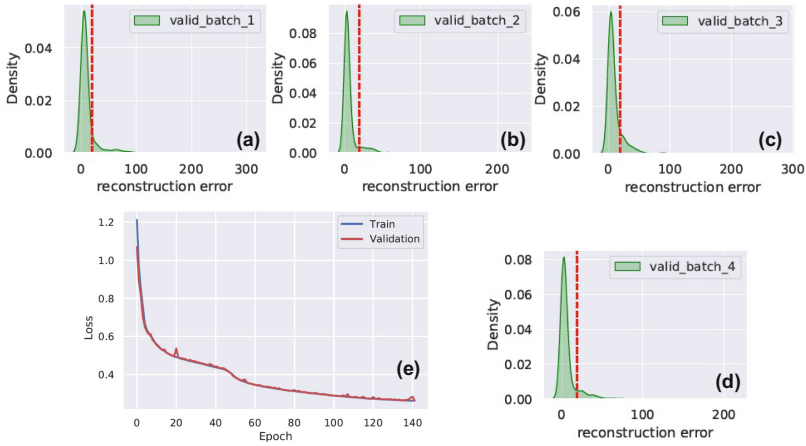

Fig. 2. (e)–GRU-AE(s) training curve (log_cosh against the epoch numbers). (a)–(d) demonstrates the kernel density plot for the four validation batches with the threshold (red dotted vertical line).

4.2 GRU Based Autoencoder

In this paper, we have developed a GRU based AE for fault detection in penicillin fermentation process. Also, the data samples from different dataset are shuffled to eliminate the distribution discrepancy. An AE is a deep learning model which can be trained to extract informative representations of the data by learning to reconstruct the input samples well enough [8]. The extracted information then can be used for other applications such as classification or regression. The basic structure of an AE consists of an encoder unit, bottleneck layer, and a decoder unit, where the encoders and the decoders are neural networks [3,4,15]. But the last layer of the encoder is called the bottleneck layer. Here, we have used the GRU layer to construct the encoder and decoder models so that the GRU based AE can extract time dependent latent information.

AE uses unlabeled data to train itself. If we have an unlabeled training dataset D with N number of samples x_i from $i = 1, ..., N$. Mathematically, the unlabeled training data can be represented as, $D = \{x_i | i = 1, ..., N\}; x_i \in \mathbb{R}^n; n \in \mathbb{N}$. The encoder function can be written as $h_i = g(x_i)$, where the h_i is the latent representation layer with the dimension of q, i.e. $h_i \in \mathbb{R}^q$. Then the goal of the encoder is to reduce the dimension of the input data from dimension n to q, i.e. $g : \mathbb{R}^n \rightarrow \mathbb{R}^q$. On the contrary, the decoder with a function say $f(.)$ reconstructs the input data from h_i. Mathematically it can be denoted as $\bar{x} = f(h_i) = f(g(x_i))$. A loss function representing the reconstructed samples $\bar{x}$ and original samples x is minimized through a learning algorithm by the AE to learn the latent representation of the data. Typically, a deterministic AE follows

mean square error (MSE) [15], as a loss function, i.e. $Loss = \frac{1}{N}\sum_{i} |x_i - \bar{x}_i|^2$. However, in this work, we have used log hyperbolic cosine function [9] to minimize the error instead of MSE.

$$L = \sum_{i}^{n} log(cosh(y_p - y_t))$$

where y_p denotes the predicted value and y_t denotes the ground truth value.

4.3 GRU-AE Based Fault Detection

An under-complete AE extracts the features by learning to reconstruct the input data. Therefore, an AE is a highly data-specific deep learning algorithm. If a trained AE is given a new dataset outside its training data domain, then the trained AE will not be able to reconstruct the new dataset. The high reconstruction error for the new dataset indicates that the features comprise faulty samples. Moreover, from the training data, a threshold can be inferred and can be used to determine faulty data samples.

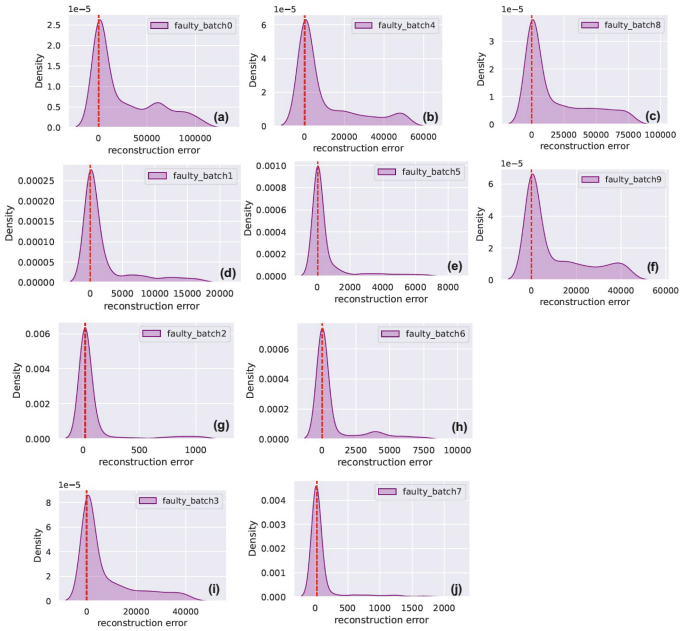

Fig. 3. (a)–(j) demonstrates the kernel density plot for the ten validation batches with the threshold (red dotted vertical line) identified from the first validation batch.

5 Experiments and Results

5.1 Dataset

The dataset comes from an industrial-scale penicillin simulator called IndPenSim [10]. The entire database comprises 90 batches (a batch can be understood as a dataset of particular operating conditions) of normal operating conditions and 10 batches of faulty conditions. Each batch has 38 features, including manually and automatically controllable variables. Also, each batch contains around 1000 data instances and is set to run for 230 h.

5.2 Data Preprocessing

To improve the data quality before applying the GRU-based Autoencoder, we have applied the data preprocessing step. First, we imputed the missing values. Then feature selection is performed to eliminate unnecessary features which reduces model complexity. The features named Time (h), Agitator RPM (RPM: RPM), Ammonia shots(NH3_shots: kgs), Fault reference(Fault_ref: Fault ref), 0 - Recipe driven 1 Operator controlled(Control_ref: Control ref), 1- No Raman spec, 1-Raman spec recorded, Batch reference(Batch ref: Batch ref), 2-PAT control(PAT ref:PAT ref), Batch ID and Fault flag are discarded from each of the batch. The data samples from the 86 out of 90 normal batches are shuffled to make one single large dataset for training the GRU-AE deep learning model. The combined and shuffled dataset is then normalized using the Python Scikit learns *StandardScaler*() library. The *StandardScaler*() method transforms the features into the range about the mean 0 and standard deviation 1, that is each value from a feature is normalized by subtracting the mean of the corresponding feature and divided by the standard deviation. However, the normalized dataset is not directly applicable to train the GRU-AE deep learning model since the GRU layer requires the training dataset to be structured as (batch size, time step, number of features). Therefore, the normalized training data is then reshaped to fit the GRU layers.

5.3 Model Training

Since the GRU-AE deep learning model is a one-class learning model, only the batches of normal operating conditions are used for the training process, and the faulty samples are used for testing the model. From the 90 normal batches 86 batches are used for the training, and the remaining 4 batches are used for validation. The first validation batch is used to determine the threshold for detecting faults in the test faulty batches.

In this work, we have used the GRU layers in the encoder and decoder part of an Autoencoder to learn the time-varying nature of the penicillin fermentation dataset. The bottleneck layer of the Autoencoder is responsible for extracting meaningful features from the high-dimensional dataset. However, selecting the size of a bottleneck layer is a crucial task for an Autoencoder. Therefore, we

have tried several GRU-based Autoencoders with 3 different bottleneck sizes, including 16, 8, and 4. The first hidden layer of the GRU-Autoencoder consists of 32 neurons with the return sequences True, sigmoid recurrent activation function, glorot uniform kernel initializer, orthogonal recurrent initializer, and regularizer 0.001. The second and the third hidden layer follows the first hidden layer for setting the parameters. However, the second and the third hidden layer consists of 16 and 8 neurons, respectively. After the third hidden layer i.e. the bottleneck layer, we have added the Repeat Vector layer to reshape the compressed data to feed into the decoder. The hidden layers in the decoder only comprise the neurons (in reverse order of encoder hidden layers), activation function, and return sequences. All the layers use tanh as the activation function.

The model is trained for 300 epochs with batch size 512, optimizer Adam, learning rate 0.001, and loss function *log cosh*. The *log cosh* improves the AE training process by behaving as L_2 loss for small values and L_1 loss for large values [9]. Also, it keeps balance between reconstruction and generation.

5.4 Results

We have developed a GRU-based AE to detect faults in the penicillin fermentation process. The GRU-AE is first trained with the batches from normal operating conditions. Therefore, the GRU-AE has learned to reconstruct data samples coming from the fermentation process under a normal controlling environment. Moreover, the GRU-AE is trained on the normal batches without (GRU-AE) and with data shuffling (GRU-AE(s)). Figure 2(e) shows the learning curve of the GRU-AE(s) training process. The first of the four validation batches is used to determine the threshold using equation (1) for differentiating the normal data samples from the abnormal or faulty data samples. Figure 2(a), 2(b), 2(c), 2(d), shows the kernel density plot of the GRU-AE(s) reconstruction error for the four validation batches with the threshold. It is evident from these figures that the kernel density of the reconstruction errors for the four validation batches are on the left side of the threshold, meaning the GRU-AE(s) identifies these batches as normal batches. After training, we tested the trained GRU based AE(s) on ten faulty batches to identify faults. Figure 3 depicts the kernel density plot of the reconstruction errors of the faulty batches against the thresholds and we can observe that noteworthy portion of all the faulty batches are on the right side of the threshold, meaning these batches are detected as faulty batch.

Table 1 shows the results of the fault detection rate (FDR) for ten faulty batches. We have also compared the GRU-based AE model with Feed Forward AE (FF-AE) and LSTM-AE models. From Table 1 we can observe that GRU-AE(s) outperformed LSTM-AE and FF-AE for detecting faults. It is also evident from this table that the trained GRU-AE(s) on the shuffled data outperformed all the models including the conventional GRU-AE (trained without shuffling the data from different batches) model with average FDR 94.74%.

$$Threshold = median \pm \frac{inter\_quartile\_range}{1.35} \tag{1}$$

Table 1. Fault detection rates of the deep learning models on penicillin fermentation dataset

Faulty batches	FF-AE	FF-AE(S)	LSTM-AE	LSTM-AE(S)	GRU-AE	GRU-AE(S)
fault batch 0	34.48	41.87	60.59	76.35	76.85	100
fault batch 1	87.70	100	95.08	94.26	99.18	99.18
fault batch 2	67.33	75.24	74.26	74.25	78.22	87.12
fault batch 3	100.00	100	100.00	100.0	100.00	100
fault batch 4	30.32	31.38	62.77	79.25	76.60	97.87
fault batch 5	62.38	68.31	69.31	68.31	69.31	69.30
fault batch 6	25.12	10.34	15.76	48.27	52.71	95.56
fault batch 7	86.07	98.36	94.26	93.44	96.72	98.36
fault batch 8	100.00	100	100.00	100.0	100.00	100
fault batch 9	100.00	100	100.00	100.0	100.00	100
average	69.34	72.55	77.20	83.41	84.96	94.74

Legend: FF-AE– Feedforward Autoencoder, LSTM-AE–Long short term memory Autoencoder, GRU-AE– Gated recurrent unit Autoencoder, GRU-AE(s)– Gated recurrent unit Autoencoder on shuffled data

6 Conclusion

In this work, we have proposed a GRU-AE deep learning-based fault detection model to detect faults in the Penicillin Fermentation Process. Due to varying controlling conditions, the batches of the dataset generated from the fermentation tank vary in distribution. The distribution discrepancy creates difficulty in training a deep learning model. We have randomly shuffled the data instances from different batches of normal operating conditions to train the GRU-AE model. The GRU-AE model trained on shuffled data achieved 94.74% average FDR, whereas, the GRU-AE trained on unshuffled data from multiple batches achieved 84.96% FDR. Moreover, the GRU-AE model on shuffled data outperformed the LSTM-AE and FF-AE models. Code for this paper will be available at - https://github.com/ArifeenDipto/GRU-Autoencoder-FaultDetection

References

1. Abbasi, M.A., Khan, A.Q., Mustafa, G., Abid, M., Khan, A.S., Ullah, N.: Data-driven fault diagnostics for industrial processes: an application to penicillin fermentation process. IEEE Access **9**, 65977–65987 (2021)
2. Anon.: Gated recurrent units (gru). webpage (2023). https://d2l.ai/chapter_recurrent-modern/gru.html
3. Arifeen, M., Ghosh, T., Islam, R., Ashiquzzaman, A., Yoon, J., Kim, J.: Autoencoder based consensus mechanism for blockchain-enabled industrial internet of things. Internet Things **19**, 100575 (2022)
4. Arifeen, M., Petrovski, A.: Bayesian optimized autoencoder for predictive maintenance of smart packaging machines. In: 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS), pp. 1–6. IEEE (2023)

5. Arifeen, M., Petrovski, A., Petrovski, S.: Automated microsegmentation for lateral movement prevention in industrial internet of things (iiot). In: 2021 14th International Conference on Security of Information and Networks (SIN), vol. 1, pp. 1–6. IEEE (2021)
6. Azamfar, M., Li, X., Lee, J.: Deep learning-based domain adaptation method for fault diagnosis in semiconductor manufacturing. IEEE Trans. Semicond. Manuf. **33**(3), 445–453 (2020)
7. Bandaru, R.: Domain adaptation. webpage (2023). https://rohitbandaru.github.io/blog/2021/08/09/Domain-Adaptation.html
8. Bank, D., Koenigstein, N., Giryes, R.: Autoencoders. arXiv preprint arXiv:2003.05991 (2020)
9. Chen, P., Chen, G., Zhang, S.: Log hyperbolic cosine loss improves variational auto-encoder (2018)
10. Goldrick, S., Duran-Villalobos, C.A., Jankauskas, K., Lovett, D., Farid, S.S., Lennox, B.: Modern day monitoring and control challenges outlined on an industrial-scale benchmark fermentation process. Comput. Chem. Eng. **130**, 106471 (2019)
11. Khan, A.R., Khan, A.Q., Raza, M.T., Abid, M., Mustafa, G.: Design of robust fault detection scheme for penicillin fermentation process. IFAC-PapersOnLine **48**(21), 589–594 (2015)
12. Li, K., Feng, J.: Grouping multi-rate sampling fault detection method for penicillin fermentation process. Can. J. Chem. Eng. **98**(6), 1319–1327 (2020)
13. Li, S., Yu, J.: Deep transfer network with adaptive joint distribution adaptation: a new process fault diagnosis model. IEEE Trans. Instrum. Meas. **71**, 1–13 (2022)
14. Li, S., Yu, J.: A multi-source domain adaptation network for process fault diagnosis under different working conditions. IEEE Trans. Ind. Electron. **70**, 6272–683 (2022)
15. Michelucci, U.: An introduction to autoencoders. arXiv preprint arXiv:2201.03898 (2022)
16. Sarker, I.H.: Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Comput. Sci. **2**(6), 420 (2021)
17. Taqvi, S.A.A., Zabiri, H., Tufa, L.D., Uddin, F., Fatima, S.A., Maulud, A.S.: A review on data-driven learning approaches for fault detection and diagnosis in chemical processes. ChemBioEng Rev. **8**(3), 239–259 (2021)
18. Wu, H., Zhao, J.: Fault detection and diagnosis based on transfer learning for multimode chemical processes. Comput. Chem. Eng. **135**, 106731 (2020)
19. Xia, B., Wang, K., Xu, A., Zeng, P., Yang, N., Li, B.: Intelligent fault diagnosis for bearings of industrial robot joints under varying working conditions based on deep adversarial domain adaptation. IEEE Trans. Instrum. Meas. **71**, 1–13 (2022)