Fuzzy Systems and Data Mining VII
A.J. Tallón-Ballesteros (Ed.)

doi:10.3233/FAIA210180

Edge Analytics for Bearing Fault Diagnosis Based on Convolution Neural Network

Valentin PERMINOV [a] and Vladislav ERMAKOV [a] and Dmitry KORZUN [a,1]
[a] *Petrozavodsk State University, Russia*

Abstract. Advanced technologies of Sensorics and Internet of Things (IoT) enable real-time data analytics based on multiple sensors covering the target industrial production system and its manufacturing processes. The rolling bearings fault diagnosis is one of the most urgent problems and can be solved by using convolution neural networks and edge artificial intelligence (edge AI) devices. The limitations of the hardware platform must be taken into account to achieve maximum performance. In this paper, we analyze efficient CNN architecture for bearings fault diagnosis that is able to process data in real-time on edge AI devices. We observe that the accuracy of the proposed CNN is unsatisfactory for practical use, and better accuracy is possible with increasing the number of bearings in the training dataset.

Keywords. Fault diagnosis, convolutional neural network, edge computing, vibration diagnostics, bearing fault diagnosis, edge AI

Introduction

Continuous condition monitoring of industrial equipment enables early detection of malfunctions of machinery and its units, thus increasing the efficiency of staff scheduling, reducing costs, and preventing accidents. In this work, we extend our previous study on applying neural network data analysis to diagnose industrial rotary machinery failures using edge artificial intelligence (edge AI) devices [1]. We focus on the rolling bearings fault diagnosis, as an important practical problem. Up to 40% of rotary machinery failures caused by bearings faults [2]. There are many studies aimed to the problem of bearings fault diagnosis and a number of datasets have been collected for training and evaluating machine learning and deep learning models [3].

In order to increase the autonomy of the monitoring system and reduce the data traffic, it is preferable to analyze the data near the place of its acquisition in accordance with the edge computing paradigm. When building an industrial monitoring system, placing high-performance servers near the monitored equipment is usually difficult due to severe operating conditions, such as a wide range of temperatures, dust, and vibration. In such conditions, it is preferable to use compact, low-power devices.

[1]Corresponding Author: Dmitry Korzun, Petrozavodsk State University, Lenin Ave. 33, Petrozavodsk, Republic of Karelia, 185910, Russia; E-mail: dkorzun@cs.karelia.ru.

The balance between the computational complexity of the applied data processing methods and the performance of the used devices should provide real-time analysis, since equipment malfunctions must be detected immediately. Recently, among edge computing devices, the neural network accelerators – edge artificial intelligence (edge AI) devices – have been developed, optimized for intelligent data processing with neural networks usage [4]. At the same time, there has been great progress in the use of neural networks for rolling bearings fault diagnosis [3]. In this paper, we combine these two research directions to assess the practical applicability of existing solutions. Our contribution is the following.

- We propose a CNN architecture that addresses the hardware limitations of the edge AI device and ensures efficient use of hardware resources.
- We evaluate the proposed CNN accuracy in accordance with a protocol close to the conditions of a real industrial monitoring system deployment.
- We evaluate the performance of the proposed CNN on the low-capacity edge AI device and show that can perform real-time sensory data analysis in industrial monitoring tasks.

The rest of the paper is organized as follows. Section 1 considers existing approaches to rotating machinery fault diagnosis and edge neural network computing applications. Section 2 describes our solution to the bearing fault diagnosis using the edge AI device, as well as introduces the dataset and data preprocessing technique. Section 3 shows the results of our experiments. Finally, Section 4 summarizes this study.

1. Related work

1.1. Condition monitoring and bearing fault diagnosis

Bearing defects can be diagnosed using frequency spectral analysis, based on bearings characteristic fault frequencies, which are related to the defect type, bearing geometry and operating mode through a well-defined mechanical model. However, this method requires human expert to make a decision. Machine learning (ML) and deep learning (DL) techniques have been explored to automate bearing diagnostics and enable continuous condition monitoring (CM). The ML includes such methods as artificial neural networks (ANN), principal component analysis (PCA), k-nearest neighbors (k-NN), support vector machines (SVM) and others [2]. These methods perform classification task based on a set of features that describe the original data, such as mean, standard deviation, root mean square, kurtosis, crest factor, etc., calculated from the raw signal, spectrum, envelope spectrum or their intervals. The set of these features is determined by the researcher and affects the accuracy of the resulting method. Therefore, a lot of human effort and proficient domain knowledge are required.

Several DL methods have been developed, such as convolutional neural network (CNN), auto-encoder (AE), deep belief network (DBN), recurrent neural network (RNN), generative adversarial network (GAN). Applying the DL methods to bearing diagnostics has been investigated in [2]. The benefit is end-to-end learning, eliminating the need for manual feature engineering and selection. The CNN shows the state-of-the-art performance in many data classification problems and is therefore of particular interest among researchers involved in solving the problem of bearing diagnostics [3].

The input data of the CNN could be the raw vibration signal, spectrum, spectrogram, envelope spectrum, wavelet scaleogram. The advantage of using a raw vibration signal as input data for a CNN is that there is no need for data preprocessing by other methods, which is especially important when using neural network accelerators, since preprocessing methods may not be supported by such accelerators. Consequently, it becomes necessary to use additional hardware accelerators (for example, FFT), or perform preprocessing on the CPU, which can increase the total processing time, compared to the processing performed entirely on the neural network accelerator. The hardware used can also limit the choice of neural network architectures. In order to enable hardware acceleration for applied in our research hardware, we use 2D CNN (see Section 2).

The use of 2D CNN for raw vibration signal bearing diagnostics has been explored in papers [5,6,7]. Wen et al. [5] apply LeNet consists of four convolution + pooling layers and two fully connected layers to classify bearings fault type by raw vibration signal fitted row-by-row in 64x64 matrix. The obtained model achieves accuracy of 99.79% on Case Western Reserve University (CWRU) dataset. Guo et al. [6] applied a similar approach, differing in that they split the tasks of fault type classification and fault size estimation among separate CNNs. One CNN was used to classify the fault type and other three separate CNNs to estimate the fault size for each type of fault. They demonstrate 97.9% accuracy in fault type classification in tenfold cross validation.

Liu et al. [7] have proposed a dislocated time series convolutional neural network (DTS-CNN), which features the transformation of 1D raw signal into the 2D matrix by row-by-row placing 1D raw signal into the rows of the matrix in such a way that each row shifts relative to the previous one. Wherein the offset step increases with each row. This approach aims to allow CNN to extract periodic fault information between non-adjacent signals. The authors demonstrated up to 6.7% improvement in DTS-CNN accuracy over CNN in the induction motor fault diagnosis task.

Pandhare et al. [8] evaluate the performance of CNN with a 2D input formed from a 1D raw vibration signal. The sizes of the convolution kernels and poolings are one along the first dimension. That is, they cover only one row. Thus, such CNN is equivalent to a 1D CNN that processes a one-dimensional signal by one-dimensional kernels. Cross-validation on Paderborn University dataset was done by splitting the training and test datasets by bearing instances, so CNN performance was evaluated on bearings that were not present in the training dataset. The authors demonstrate that the average accuracy of CNN with raw vibration signal input is superior to other considered alternatives, but does not exceed 61.86%, which is an unsatisfactory result for practical use.

Researches [5,6,7,8] demonstrate that CNN with raw vibration input outperforms machine learning methods based on hand-crafted features. However, in the context of the vibration diagnostics for the edge computing conditions, the mentioned works have the following disadvantages. First, the CNN accuracy evaluation methodology in papers [5,6,7] does not match the actual model performance that would be expected during deployment. The accuracy is evaluated on the same bearings, which was used in training. While when deploying the model to production, CNN will have to determine the condition of the bearing, which was absent in the training dataset. Second, the CNN models proposed in works [5,6,7] do not allow full utilization of hardware acceleration on the edge AI device , since they include an unsupported kernel and pooling sizes.

1.2. Edge neural network computing

The data processing near the place of its acquisition, by the edge computing paradigm, can provide reduced latency and traffic and increased privacy and autonomy. With increased interest in artificial intelligence and its applications, edge AI devices have evolved to accelerate the inference of neural networks.

A comprehensive study of the performance of edge AI devices was carried out in [4]. The authors note that the inference time on neural network accelerators is not directly related to the number of operations, in contrast to the inference on the Central Processing Unit (CPU). It is shown that, in some cases, state-of-the-art lightweight neural networks are computed more slowly than more computationally complex but conformed with the target hardware platform ones. The authors emphasize the need to develop an individual neural network architecture for each accelerator, taking into account the features of the target hardware platform and using such sets of operations, layers, and their parameters that ensure the most efficient utilization of hardware resources.

There are a number of works using edge AI devices for various practical applications, such as face mask detection [9], resilient image compression for IoT cameras [10], mineral granulometric analysis [11], conveyor belt longitudinal rip detection [12]. Most of the work focuses on the problems of image analysis. In this article, we explore the applicability of edge AI devices for analyzing time-domain signals from sensors, specifically, for analyzing vibration signals in order to identify bearing faults.

2. Experimental setup

2.1. Edge neural network computing device

We use the Kendryte K210 system-on-chip as an edge neural network computing device. This system-on-chip has a dual-core CPU, interfaces for connecting sensors and data transmission modules, and a hardware accelerator unit for CNN inference, suiting well for edge AI applications. We simulate the data of vibration sensor by transmitting signal fragments from the dataset via Universal Asynchronous Receiver-Transmitter (UART).

The hardware platform imposes the following restrictions on neural networks used. Only CNN could be hardware accelerated, and its size should not exceed 5 MB. According to the nncase neural network compiler documentation, which is used to deploy CNN to the Kendryte K210, only a sequence of convolution, batch normalization, activation, and pooling operations could be hardware accelerated. The following restrictions are imposed on the parameters of these operations to be accelerated:

- 2D convolution or 2D depthwise convolution;
- kernel size 1x1 or 3x3;
- stride 1 or 2;
- channels number from 1 to 1024;
- input feature map size up to 320x240;
- output feature map size no less than 4x4;
- same symmetric zero padding;
- pooling size 2x2, 4x4, or without pooling;
- pooling type: max or average.

Also, fully connected (dense) layers could be accelerated through conversion to convolution operation. However, such a conversion requires tensor transpose and padding operations that are performed on the CPU and add data transfer operations between the CPU and the hardware accelerator.

We use Kendryte K210 standalone SDK to program Kendryte K210, nncase neural network compiler to deploy CNN to the Kendryte K210 with hardware acceleration and kflash utility for firmware uploading. We measure CNN inference time through the system clock. To profile inference time layer-by-layer we compile firmware with the preprocessor directive `NNCASE_DEBUG` set to `1`, which enables the layer execution time to be output to the debug serial port. The CNN inference time measurement and inference profiling were performed separately, using different firmware compilations, since printing the layer execution time to the serial port slows down inference.

Our experiments show the hardware acceleration on Kendryte K210.

- The difference in inference time with kernels 3×3 and 1×1 is less than 5%.
- The layer (as a set of operations convolution+activation+pooling) inference time does not depend on stride, activation function type, pooling size and type.

Considering limitations and features mentioned above, we develop a CNN for rotating machinery fault diagnosis, described in the next section.

2.2. CNN for condition monitoring

In our previous work [1], we used 1D CNN and showed that it was able to run on Kendryte K210. However, the hardware acceleration was not utilized, and all computations were performed on the CPU. Based on this, the measured neural network execution time was approximately 1.66 times larger than the size of the input signal frame. In this paper, we take into account the constraints, which have to be met and solved to enable hardware acceleration (see Section 2.1). Thus, we focus on the 2D CNN.

The input data is the raw vibration signal placed in a 2D tensor (Section 2.3). The convolution of this data with a 2D kernel (for example, 3x3 kernel) could be thought of as a kind of atrous (dilated) 1D convolution. The atrous convolution application to bearing fault diagnosis was studied in [13]. This type of convolution enables filters field of view enlarging without increasing the number of parameters or the amount of computation.

We have chosen the sequence of convolution, batch normalization, activation, and pooling operations as the main building blocks of our CNN and will refer to them as "KPUConv2D" layer. The same denotation is used by nncase neural network compiler in internal representation of computational graph. This choice is justified by the fact that in the used hardware platform, this sequence of operations is an elementary hardware-accelerated operation. The proposed CNN consists of a sequence of KPUConv2D layers followed by a global average pooling layer and a fully connected (dense) layer with a softmax activation function. The detailed description of the architecture of the proposed CNN is shown in Table 3. For all convolutions, we used kernel size of 3x3 and stride 1x1. These values of kernel size and stride have been selected because they are expected to provide the largest number of degrees of freedom of a neural network while slightly affect the inference time compared with other allowed ones (see Section 2.1). The padding type of convolution was set to "same" to comply with restrictions of used hardware. The Keras with TensorFlow backend is used to implement and train CNN. Deploying CNN on Kendryte K210 was done using the nncase compiler.

Table 1. Categorization of dataset.

Fold No.	*Healthy (Class 1)*	*Outer ring damage (Class 2)*	*Inner ring damage (Class 3)*
1	K001	KA04	KI04
2	K002	KA15	KI14
3	K003	KA16	KI16
4	K004	KA22	KI18
5	K005	KA30	KI21

2.3. Condition monitoring dataset

The Paderborn University Bearing Dataset is used [14] to evaluate the performance of proposed method. The dataset includes 32 bearings, 12 of which have artificial damages and 14 have natural damages caused by accelerated lifetime tests. The rest are the baseline without damages. There are 80 vibration signal fragments for each bearing with duration of 4 s sampled at 64 kHz and acquired in different operating conditions.

We split the dataset into three parts: train, validation, and test datasets. We select different bearings for the test and training datasets based on the fact that when deploying a real monitoring system, bearings are monitored that were not in the training dataset. We evaluate the model in a cross-validation manner, choosing one fold for test and the other four folds for train and validation, as shown in Table 1. The files from train and validation folds were randomly split in the following proportion: 80% as train and 20% as validation dataset. Hence, the validation dataset is similar to the ones that are used in works [7,5] and shows whether the CNN is able to classify the same bearings on which it was trained. The test dataset is the same as in works [8,15] and shows do the CNN can classify new bearings. Thus test dataset shows the applicability of the model in production, where CNN is intended to classify new bearings, which was absent in the training dataset. Further, we will denote a particular combination of train, validation, and test data by the fold number used for the test dataset.

We normalize raw vibration signal from the original dataset to mean and standard deviation. To match 1D raw vibration signal with 2D CNN input, we apply the method suggested in [5]: 1D raw vibration signal is fitted into a 2D array line by line. In production, this transformation would not require any additional operations with data since the raw data from the sensor could accumulate in a memory buffer in the required layout and then be ready to be passed to the 2D CNN input. Since in the Paderborn University Bearing Dataset each file contains a signal fragment with a duration of 4 seconds, but CNN input length is smaller, at each training step, a random frame is selected from a random file from the dataset. At each training epoch, 7680 training, 1920 validation, and 2400 test samples are generated.

3. Results and Discussion

3.1. Rolling bearing fault classification

The neural network was trained for 100 epochs. Each epoch included 7680 training, 1920 validation, and 2400 test unique samples, generated as described in section 2.3. We use

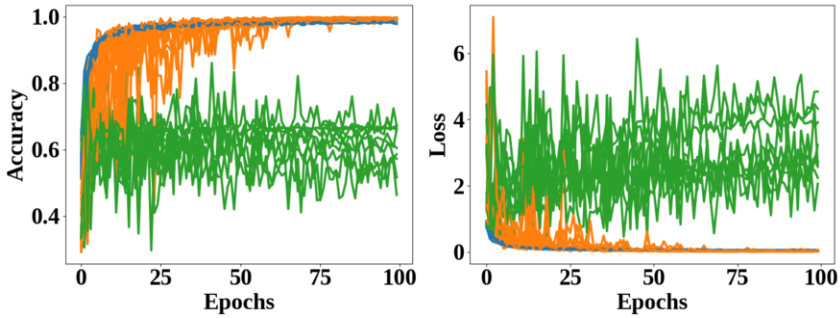

Figure 1. The accuracy (left) and loss (right) curves: training (blue), validation (orange), and test (green).

Adam optimizer with batch size of 32, initial learning rate of 0.001, and exponential learning rate decay after ten epochs with exponent 0.01. The training was performed ten times for each fold to test the stability of the training process. The obtained learning curves of CNN accuracy and loss for fold 1 are shown in Figure 1. The CNN accuracy across all five folds is summarized in Table 2. The accuracy mean and standard deviation were calculated across all trials for each fold and across all folds and trials.

The large fluctuations of accuracy and loss on the validation dataset and even larger on the test dataset are observed. These fluctuations are observed both within an individual training trial and between trials. While the fluctuations on the validation dataset decrease with each epoch as the learning rate decreases and therefore could be caused by the stochastic nature of the training process, the fluctuations on the test dataset are most likely to be related to the characteristics of the test data. The fluctuations on the test dataset within an individual training trial indicate the unrepresentativeness of the test dataset. The fluctuations on the test dataset between trials were caused by random weights initialization at the beginning of each trial and the stochastic nature of the training process. These fluctuations indicate that loss reaches different local minima, and CNN learned to extract different features, which have varying degrees of generalization. This, in turn, indicates the unrepresentativeness of the training dataset.

Table 2 shows that CNN reaches high accuracy on the validation dataset. However, classification accuracy on test dataset is low. Similar results were observed in [8,15], where cross-validation by bearing instance had been used with Paderborn university dataset. The test accuracy drop is most likely due to the unrepresentativeness of the training dataset. The modes of naturally occurring defects would be very diverse. Hence, the training dataset should contain many samples of different damaged bearings to ensure that the CNN can learn to extract the most representative features. The folds 3 and 4 outstand among the others and achieve accuracy of 81.93% and 80.32%, respectively. This indicates that with these combinations of training and test data, the neural network

Table 2. CNN accuracy

Type	*Fold 1*	*Fold 2*	*Fold 3*	*Fold 4*	*Fold 5*	*Average*
Val. mean	99.27	99.31	99.08	98.01	99.67	99.07
Val. std	0.39	0.36	0.36	0.64	0.16	0.38
Test mean	60.70	28.57	81.93	80.32	37.83	57.87
Test std	6.93	5.73	4.64	7.99	4.92	6.04

is able to learn to extract features that describe well those bearings on which the neural network was tested. With an increase in the number of bearings in the training dataset, the accuracy of the CNN will increase and reach a level suitable in practice.

3.2. CNN performance evaluation on edge device

The CNN had been deployed to the Kendryte K210 edge AI device to evaluate its performance. After CNN compilation to KModel format by nncase compiler, the CNN consumes 17 580 bytes of storage and 81 920 bytes of working memory. The measured CNN inference time was 7987 us with a standard deviation of 3 us. Taking into account that the input size of CNN is 128x128 and the sampling frequency of the input signal is 64 kHz, one input sample covers a 256 ms time slice, which is more than 30 times larger than CNN inference time. This enables vibration signal processing and fault diagnosis in real-time with a large margin. However, additional computation resources would be used to acquire the vibration signal, normalize it and handle the result of processing.

Comparing the results with obtained earlier [1], the number of FLOPS increases from 3.3 MFLOPS to 5.6 MFLOPS, while inference time decrease from 212 to 8 ms and the ratio of the incoming data flow intensity to the time of its processing becomes 50 times greater. The performance gain is due to the use of the hardware acceleration of convolution. We profile CNN inference time layer-by-layer (see Table 3). The operation names are taken from the nncase interpreter. The KPUConv2D layer includes a sequence of convolution, batch normalization, activation, and pooling layers.

The KPUConv2D operation is hardware accelerated and therefore consumes less than 10% of the total inference time, despite the fact that it includes more than 99% of the computations. Half of the inference time is spent loading data into the hardware acceleration unit. Unloading data from the hardware acceleration unit also takes additional time, which is included in the last KPUConv2D operation. The quantization and dequantization operations are necessary because the hardware acceleration unit operates with numbers in the uint8 format, while the information at the input and output of the neural network is represented by float32 numbers. These operations take about 30% of the total inference time. The total inference time summed up from the layer-by-layer profile is about 3 ms less than the measured neural network interpreter invocation time. Apparently, this time is spent on auxiliary operations, such as data copying between buffers, DMA and interrupts setting up. This implies that for small neural networks, auxiliary operations take up most of the inference time, and the number of convolution operations does not increase the inference time that much as when executing on the CPU.

The KPUConv2D operation on the hardware accelerator of Kendryte K210 is performed in unsigned integers to increase performance. The quantization of weights and activations can lead to problems with the inference accuracy of the neural network. The test data of one fold were processed on the Kendryte K210 by obtained CNN to assess the accuracy after deployment. Results showed a slight decrease in accuracy of about 1%. However, this degradation of accuracy is negligible compared to misclassification on the test dataset, which is observed even at the stage of training the CNN.

Table 3. CNN architecture and time profiling

Layer	*Output Shape*	*Parameters*		*Operation*	*Time, us*
Input	(128, 128, 1)			Quantize	1397
				KPUUpload	2598
KPUConv2D	(64, 64, 2)	Filters, Pooling	2, 2x2	KPUConv2D	25
KPUConv2D	(64, 64, 2)	Filters, Pooling	2, –	KPUConv2D	19
KPUConv2D	(64, 64, 2)	Filters, Pooling	2, –	KPUConv2D	18
KPUConv2D	(64, 64, 2)	Filters, Pooling	2, –	KPUConv2D	18
KPUConv2D	(64, 64, 2)	Filters, Pooling	2, –	KPUConv2D	18
KPUConv2D	(32, 32, 4)	Filters, Pooling	4, 2x2	KPUConv2D	21
KPUConv2D	(32, 32, 4)	Filters, Pooling	4, –	KPUConv2D	18
KPUConv2D	(32, 32, 4)	Filters, Pooling	4, –	KPUConv2D	19
KPUConv2D	(32, 32, 4)	Filters, Pooling	4, –	KPUConv2D	18
KPUConv2D	(16, 16, 8)	Filters, Pooling	8, 2x2	KPUConv2D	21
KPUConv2D	(16, 16, 8)	Filters, Pooling	8, –	KPUConv2D	19
KPUConv2D	(16, 16, 8)	Filters, Pooling	8, –	KPUConv2D	20
KPUConv2D	(8, 8, 16)	Filters, Pooling	16, 2x2	KPUConv2D	23
KPUConv2D	(8, 8, 16)	Filters, Pooling	16, –	KPUConv2D	191
Average Pooling Layer	(1, 1, 16)			Dequantize	50
				Reduce	547
Dropout	(16)	Dropout rate	30%		
Dense	(3)	Activation	Softmax	MatMul	14
				Reduce	12
				Binary	17
				Quantize	10
				TableLookup1D	10
				Dequantize	9
				Reduce	11
				Binary	15
				Total	5138

4. Conclusion

This paper considered opportunities of the edge analytics for fault diagnostics in industrial rotary machinery based on CNN methods. We propose a CNN architecture that addresses the hardware features of the edge AI device and ensures efficient use of hardware resources. We show that low-capacity edge AI devices are able to perform real-time CNN-based sensor data analysis in industrial monitoring tasks. We evaluated the proposed CNN accuracy in environment close to real industrial monitoring. Basically, we observed the unsatisfactory accuracy of CNN for practical use. The suggested option for better accuracy is increasing the number of faulty and healthy bearings in the training dataset. Therefore, with the development of Industrial Internet and Big Data larger datasets should be collected for particular machinery equipment.

Acknowledgements

This work is implemented in Petrozavodsk State University (PetrSU) with financial support by the Ministry of Science and Higher Education of Russia within Agreement no. 075-15-2021-1007 on the topic "Software and hardware methods of sensorics and machine perception for robotic systems with autonomous movement". In part of data computing infrastructure, this study was performed using the Unique Scientific Unit (UNU)—Multicomponent software and hardware system for automated collection, storage, markup of research and clinical biomedical data, their unification and analysis based on Data Center with Artificial Intelligence technologies (UNU for support of medical decision-making) (reg. number: 2075518).

References

[1] V. Perminov, V. Ermakov, and D. Korzun, "Fault diagnosis for industrial rotary machinery based on edge computing and neural networking," in *UBICOMM 2020: The Fourteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2020, pp. 1–6.

[2] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, "Deep learning algorithms for bearing fault diagnostics—a comprehensive review," *IEEE Access*, vol. 8, pp. 29 857–29 881, 2020.

[3] J. Jiao, M. Zhao, J. Lin, and K. Liang, "A comprehensive review on convolutional neural network in machine fault diagnosis," *Neurocomputing*, vol. 417, pp. 36–63, 2020.

[4] X. Tang, S. Han, L. L. Zhang, T. Cao, and Y. Liu, "To bridge neural network design and real-world performance: A behaviour study for neural networks," *Proceedings of Machine Learning and Systems*, vol. 3, 2021.

[5] L. Wen, X. Li, L. Gao, and Y. Zhang, "A new convolutional neural network-based data-driven fault diagnosis method," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5990–5998, 2017.

[6] X. Guo, L. Chen, and C. Shen, "Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis," *Measurement*, vol. 93, pp. 490–502, 2016.

[7] R. Liu, G. Meng, B. Yang, C. Sun, and X. Chen, "Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1310–1320, 2016.

[8] V. Pandhare, J. Singh, and J. Lee, "Convolutional neural network based rolling-element bearing fault diagnosis for naturally occurring and progressing defects using time-frequency domain features," in *2019 Prognostics and System Health Management Conference (PHM-Paris)*. IEEE, 2019, pp. 320–326.

[9] E. Torres-Sánchez, J. Alastruey-Benedé, and E. Torres-Moreno, "Developing an ai iot application with open software on a risc-v soc," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*. IEEE, 2020, pp. 1–6.

[10] P. Hu, J. Im, Z. Asgar, and S. Katti, "Starfish: resilient image compression for aiot cameras," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 395–408.

[11] N. F. d. C. Meira, M. C. Silva, R. A. Oliveira, A. Souza, T. D'Angelo, and C. B. Vieira, "Edge deep learning applied to granulometric analysis on quasi-particles from the hybrid pelletized sinter (hps) process," in *23rd International Conference on Enterprise Information Systems*, 2021.

[12] E. Klippel, R. A. R. Oliveira, D. Maslov, A. G. C. Bianchi, S. E. D. Silva, and C. T. B. Garrocho, "Conveyor belt longitudinal rip detection implementation with edge ai," 2021.

[13] Y. Chen, G. Peng, C. Xie, W. Zhang, C. Li, and S. Liu, "Acdin: Bridging the gap between artificial and real bearing damages for bearing fault diagnosis," *Neurocomputing*, vol. 294, pp. 61–71, 2018.

[14] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro, "Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification," in *Proceedings of the European conference of the prognostics and health management society*, 2016, pp. 05–08.

[15] V. Perminov and D. Korzun, "On applying convolutional neural network to bearing fault detection," in *Conference of Open Innovations Association, FRUCT*, no. 29. FRUCT Oy, 2021, pp. 475–480.