Proceedings of CONF-MLA 2026 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2026.LD38567

Post Training Quantization of a Compact 1D CNN for Bearing Fault Diagnosis

Zixuan Qiu

Shanghai Qibao Dwight High School, Shanghai, China
daniel605.qiu@gmail.com

Abstract. This study evaluates the static post-training INT8 quantization in a compact one-dimensional convolutional neural network to diagnose bearing fault. The model has 9,188 trainable parameters, and the four bearing states are classified according to normalized vibration windows. CWRU records at 1 hp, 2 hp and 3 hp are used for training, validation and testing, respectively. Each partition has 472 windows. Five models are trained from scratch, and only training windows are used for calibration. The saved state size is reduced from 38.75 KiB to 18.95 KiB after quantization, a decrease of 51.11%. The accuracy of the test decreases from 99.45% to 96.44%. After quantization, test accuracy varies more across training seeds. Newly introduced errors are concentrated in the outer-race fault classification. The mean CPU latency per window increases from 0.316 milliseconds to 0.623 milliseconds. The test results show storage benefits, but there is no speed improvement in the tested configuration. The shared bearing configuration and limited recording limit the generalization claim. Quantization needs separate evaluation of storage, accuracy, and hardware latency, even if the starting model is relatively compact.

Keywords: bearing fault diagnosis, 1D CNN, post-training quantization, CWRU, model storage

1. Introduction

Bearing conditions can be studied by vibration measurement. Learning-based classifiers use labeled recordings to associate signals with bearing states. Eren [1] shows that a one-dimensional convolutional network combines feature learning with bearing fault detection in a model.

For resource-limited devices, classification accuracy is one of several requirements. The model needs to fit available storage and operate within a suitable time frame. Small saved files cannot directly measure runtime memory, fewer bits cannot guarantee lower latency in every implementation.

This study explores how static INT8 post-training quantization affects the saved state size and classification accuracy of compact 1D CNNs and the CPU inference latency. In each FP32-INT8 pair, the architecture and training checkpoint are fixed. Five training seeds were reported to show how the conversion of learned weights varies.

This study does not propose new convolution operations, nor does it claim that the 1D CNN is better than the image-based method. Holdout load experiments provide the main evidence, and

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.LD38567

previous random window experiments are retained as supplementary checks.

2. Background and related work

2.1. Learning from one dimensional vibration signals

Eren [1] applied a 1D CNN to the raw bearing vibration signal, combined feature extraction and classification, so that direct signal input can be selected.

Convolution uses the same learned filters at different positions in the sequence. Sharing weights avoids assigning parameters to each input position. Lin et al. [2] used global average pooling as part of Network in Network. In our model, global average pooling is followed by a small 4-output linear classifier.

2.2. Compact design and model compression

Models can be made small through architecture or changes after training. Ruan et al. [3] combined neural architecture search and weight pruning to carry out bearing diagnosis and remaining useful life prediction. This study uses fixed three-layer architectures instead, and the measured change is attributed to the quantization of each training pair.

Han et al. [4] used pruning, trained quantization, and Huffman coding to carry out compression. Hinton et al. [5] proposed transferring predictive knowledge to a model that is easy to deploy. In this experiment, pruning and knowledge distillation were not used.

2.3. Post training quantization

Quantization uses a restricted range of levels to represent numerical values. In common affine formulations, the real value x is estimated by s(q - z), q is an integer, s is scale, and z is zero point. Rounding and clipping lead to approximation errors. Jacob et al. [6] combined quantization with integer inference training. Krishnamoorthi [7] examined post-training quantization and quantization-aware training, including accuracy and latency trade-offs.

Static post-training quantization first trains floating-point models, and then uses calibration input to estimate the activation range before converting. Quantization-aware training models the quantized effect during optimization. The current study uses the post-training conversion method to evaluate the changes without further training.

2.4. Data partitioning and research scope

Rosa and others [8] pointed out that random segments of the CWRU recording have optimistic evaluation. Separate motor loads keep the recordings separated, and can also reuse the same bearings in different partitions. The main experiment is held-out load testing, and it does not establish a general basis for independent bearings.

3. Methodology

3.1. Data and evaluation scope

Drive-end vibration recordings were obtained from the CWRU Bearing Data Center [9, 10]. There are four categories: normal, inner-race fault, ball fault, and outer-race fault. Fault recordings have a

DOI: 10.54254/2755-2721/2026.LD38567

defect of 0.007 inches, the outer-race defects are at six o'clock. Table 1 lists selected files.

Table 1. Recording identifiers and fixed data partitions

Partition	Load	Normal	Inner	Ball	Outer
Training	1 hp	98	106	119	131
Validation	2 hp	99	107	120	132
Test	3 hp	100	108	121	133

Normal recordings are resampled from 48 kHz to 12 kHz using scipy.signal.resample_poly, and the downsampling factor is four. Anti-alias filtering is applied during resampling of normal recordings, fault recordings are retained at 12 kHz. Rosa et al. [8] supported the treatment of normal recordings. Select the exact drive end variable matching the file number.

3.2. Window preparation

The first 118 non-overlapping windows are retained from each recording, each window has 1,024 points, each partition has 472 windows, and the total number of windows is 1,416. At 12kHz, each window covers about 85.3 milliseconds. Each window is independently normalized as $(x - \mu)/(\sigma + 10^{-8})$, and its mean and standard deviation are used. Signals are one-dimensional, and no spectrogram is generated.

There is no recording or exactly the same processed window in multiple partitions. All recordings come from the test rig and bearing configuration. This evaluates held-out loads, not unseen bearings or machines. The 472 test windows are not 472 independent bearings.

3.3. Compact network

The network uses three one-dimensional convolutions to process the vibration window. After each convolution, ReLU introduces nonlinear operations. There are two max pooling layers to shorten the feature sequence. The global average pool reduces each final feature channel to one value, and the linear layer outputs four classification scores. The predicted class is the class with the highest score.

Table 2. Compact 1D CNN architecture and trainable parameters

Layer	Channels or output	Kernel	Parameters
Conv1 + ReLU + max pool	1 → 16	7	128
Conv2 + ReLU + max pool	16 → 32	5	2592
Conv3 + ReLU	32 → 64	3	6208
Global average pool	64 values	—	0
Linear classifier	4 scores	—	260

As shown in Table 2, the convolution padding is 3, 2, and 1, respectively. Each max pooling layer halves the sequence length. The total number of trainable parameters is 9,188. The small channel count and global average pooling limit the storage of parameter values, ReLU does not reduce the number of feature values.

DOI: 10.54254/2755-2721/2026.LD38567

3.4. Training and quantization

Five models are trained from seeds 42-46. All use the same data partition. Training uses Adam, the learning rate is 0.001, there is a loss of cross-entropy, the batch size is 32, and training runs for 30 epochs. Select the checkpoint with the highest FP32 validation accuracy, ties retain the earliest epoch. The seeds change model initialization and training data order, not the actual test data.

Each selected checkpoint is converted using static post-training quantization with the default configuration of PyTorch (2.6.0) on x86. The Convolution module and ReLU module are fused. Calibration uses all training windows. The weights of all three convolution layers and the linear layer are checked to use qint8 after conversion. No quantization-aware training, pruning, or post-conversion fine-tuning was used. Reloading the saved quantized state preserves its predictions.

3.5. Measurement

On the fixed test set, the accuracy, macro F1, and class recall are evaluated. The results are the mean and sample standard deviations of the five training seeds, respectively. The serialization of FP32 and INT8 state dictionaries is carried out. The sizes include metadata, and 1 KiB is 1,024 bytes.

Latency is measured using an Intel Xeon 8272CL CPU, with one thread and a batch size of one. After 50 warmups, ten rounds process all test windows once per round, and the model order is alternated. The reported latency of each model is the median of the ten round means. The timing includes model calls and quantization boundaries but excludes loading or signal preparation. Deterministic algorithms are enabled for training and disabled for quantized operations because a required quantized operation lacks deterministic support.

4. Results

4.1. Accuracy across training seeds

Table 3 reports each paired run. The FP32 accuracy is relatively high. Quantized accuracy relies more on trained checkpoints: two runs have the same accuracy, two are worse, and one is slightly better.

Table 3. Correct predictions out of 472 test windows. pp refers to percentage points

Seed	FP32 correct	INT8 correct	FP32 %	INT8 %	Change pp
42	470	470	99.58	99.58	+0.00
43	470	444	99.58	94.07	-5.51
44	470	470	99.58	99.58	+0.00
45	471	424	99.79	89.83	-9.96
46	466	468	98.73	99.15	+0.42

Table 4. Main experiment summary across five training seeds

Metric	FP32	INT8
Accuracy %	99.45 ± 0.41	96.44 ± 4.37
Macro F1	0.9945 ± 0.0041	0.9633 ± 0.0455
Saved state KiB	38.75	18.95

DOI: 10.54254/2755-2721/2026.LD38567

Table 4. (continued)

Latency ms per window	0.316 ± 0.011	0.623 ± 0.040

Mean accuracy decreases by 3.01 percentage points (Table 4). The standard deviation is the variation of training runs on a fixed test set, it is not a confidence interval for the performance of new bearings.

4.2. Storage and latency

The saved state is reduced from 39,680 bytes to 19,400 bytes, and the reduction is 51.11%. This is smaller than a theoretical 75% reduction of replacing 32 bits with 8 bits because serialized states have other values or metadata. Quantization changes the numerical representation of the network without reducing the number of channels.

The Mean Latency (Mean Lat) increased from 0.316 milliseconds to 0.623 milliseconds, which is about 1.97 times that of FP32. This result has storage benefits, but not speed benefits. Operator profiling is needed to find the source of slowdown. Peak memory consumption, energy use, and actual edge device performance are not measured.

4.3. Errors introduced by quantization

Analyze the prediction window by window. A new error is a window classified correctly by FP32 but incorrectly by INT8. The corrected error is the FP32 - wrong window correctly classified by INT8, which distinguishes the changes in individual prediction from the overall accuracy.

Table 5. Errors introduced and corrected by quantization

Seed	New errors	Corrected errors	INT8 errors
42	0	0	2
43	26	0	28
44	0	0	2
45	47	0	48
46	0	2	4

There are 28 INT8 errors in seed 43, all from outer-race windows, 26 are predicted as ball faults, and two are predicted as inner-race faults. 26 errors are newly introduced through quantization. In Seed 45, 48 outer-race windows are predicted to be ball faults, with 47 new errors. The recall rate of the outer-race window is 76.27%, and 59.32%, respectively (Table 5).

There are two misclassified ball-fault windows in each of seeds 42 and 44. Seed 46 corrects two existing errors, and there are four ball fault windows predicted as outer race faults. There was no misclassification of the normal or inner race test window in either representation.

Table 6. Class recall in the main experiment, mean ± sample SD in percent

Class	FP32 recall	INT8 recall
Normal	100.00 ± 0.00	100.00 ± 0.00
Inner race	100.00 ± 0.00	100.00 ± 0.00
Ball	98.31 ± 2.08	98.64 ± 1.42

DOI: 10.54254/2755-2721/2026.LD38567

Table 6. (continued)

Outer race	99.49 ± 0.76	87.12 ± 18.63

These observations in Table 6 show that accuracy loss mainly occurs in outer-race classification in the affected runs, but do not establish which layer or activation range causes it. Errors summed in seeds are repeated predictions in the same window, and should not be regarded as new samples.

4.4. Supplementary random window experiment

There were 472 windows in the earlier dataset, 330 training windows, 71 validation windows and 71 test windows were used. The FP32 accuracy of five retrained models is 100%, the INT8 accuracy is 89.58±9.53%. All INT8 errors are outer-race windows predicted as ball faults, which gives a similar error pattern. This experiment has weak evidence, because the partitions share source recordings and the comparability of sampling rate is not established. It is different from the main data volume and preparation experiments, and this difference is not only due to the splitting strategy.

5. Conclusion

The static INT8 quantization reduces the saved state of a 1D CNN with 9,188 parameters by 51.11% in the held-out-load experiment. In five training seeds, the mean accuracy decreases by 3.01 percentage points, and the CPU latency increases to about 1.97 times that of FP32. In two runs, there were additional classification errors in the outer-race window. These results do not demonstrate faster inference or successful deployment on edge devices. In this compact model, quantization value depends on the application's storage budget, tolerance for errors, and target hardware performance. Reporting only seeds 42 or 44 would suggest that quantization preserves accuracy. The results of all five runs show that quantization does not consistently preserve accuracy.

Only one architecture, one quantization configuration and one CPU were tested. The data has four bearing states, a fault size, and an outer race position. Load separated partitions still share bearing configurations, which Rosa et al. pointed out in [8]. This study does not explore the performance of unseen bearings, machines or naturally developing faults in noisy fields. Each partition has four original recordings. The correlation between windows limits the effectiveness of evaluation. Five training seeds are used to measure sensitivity to training randomness, and they do not replace independent data collection.

Future target device measurements would test whether storage reduction helps within actual deployment budgets. These measurements need to include complete signal preparation and peak memory use. Future tests with independent bearing/machine data would examine whether the classifier generalizes beyond the current rig/bearing configuration. Check the activation ranges of the layer and quantization errors before testing alternative calibration/quantization-aware training. The choice of improved methods needs to rely on validation data, and then evaluate on a fresh hold-out test set, because the test results have been inspected.

References

[1] Eren, L. (2017). Bearing fault detection by one-dimensional convolutional neural networks. *Mathematical Problems in Engineering*, 8617315. https://doi.org/10.1155/2017/8617315

[2] Lin, M., Chen, Q., & Yan, S. (2014). Network in network. ICLR. https://arxiv.org/abs/1312.4400

[3] Ruan, D., Han, J., Yan, J., & Gühmann, C. (2023). Light convolutional neural network by neural architecture search and model pruning for bearing fault diagnosis and remaining useful life prediction. *Scientific Reports, 13*, 5484.

DOI: 10.54254/2755-2721/2026.LD38567

https://doi.org/10.1038/s41598-023-31532-9

[4] Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. ICLR. https://arxiv.org/abs/1510.00149

[5] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531. https://arxiv.org/abs/1503.02531

[6] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR*, 2704–2713. https://doi.org/10.1109/CVPR.2018.00286

[7] Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv:1806.08342. https://arxiv.org/abs/1806.08342

[8] Rosa, R. K., Braga, D., & Silva, D. (2024). Benchmarking deep learning models for bearing fault diagnosis using the CWRU dataset: A multi-label approach. arXiv:2407.14625v1. https://arxiv.org/html/2407.14625v1

[9] Case Western Reserve University. 12k drive end bearing fault data. https://engineering.case.edu/bearingdatacenter/12k-drive-end-bearing-fault-data

[10] Case Western Reserve University. Normal baseline data. https://engineering.case.edu/bearingdatacenter/normal-baseline-data