On the Post-Training Quantization of Deep Convolutional Neural Network for Time Series Classification

Cyril Meyer[1 [0000−0001−7262−999X]] (✉), Ali Ismail-Fawaz[1 [0000−0001−5385−3339]], and Germain Forestier[1,2 [0000−0002−4960−7554]]

[1] Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
{first-name.last-name}@uha.fr
[2] DSAI, Monash University, Melbourne, Australia
{first-name.last-name}@monash.edu

Abstract. Time series classification has benefited substantially from advances in deep learning, particularly convolutional architectures such as InceptionTime and LITE. However, despite their strong predictive performance, deploying these models on resource-constrained platforms remains challenging because of their computational and memory requirements. Model quantization offers a promising route to more efficient inference by reducing numerical precision, thereby lowering storage and memory costs. Among quantization approaches, post-training quantization is especially attractive because it does not require retraining. Yet its impact on deep convolutional models for time series classification remains insufficiently studied.

In this paper, we investigate the effect of post-training quantization on InceptionTime-based classifiers for time series classification. Our evaluation covers three benchmark archives: UCR, UEA, and MONSTER. We first conduct a quantization sensitivity analysis by emulating uniform quantization over a broad range of precisions, from 32-bit to 2-bit representations. We then evaluate a practical deployment scenario based on half-precision floating-point weights. This dual evaluation allows us to assess both the empirical limits of reduced numerical precision and the practical viability of quantization for model deployment.

Our results show that InceptionTime-based classifiers are largely robust to reduced numerical precision across a wide range of datasets. We identify precision regimes that preserve classification performance while substantially reducing weight precision, as well as lower-precision regimes that lead to significant degradation. Overall, our study shows that meaningful reductions in numerical precision can often be achieved without sacrificing predictive performance. These findings provide practical guidance for the efficient deployment of deep time series classifiers and offer new insights into the relationship between quantization and performance in convolutional architectures for time series analysis.

Keywords: Time Series Classification · Deep Learning · Quantization.

1 Introduction

Time series classification (TSC) [20, 6, 19] has benefited from advances in deep learning, with convolutional architectures such as InceptionTime or LITE [14, 15, 12]. However, despite their good classification performances, the deployment of these models on resource constrained platforms remains challenging due to their computational and memory requirements. Model quantization offers a promising solution, by reducing numerical precision, lowering storage and memory usage while improving inference efficiency. Post-training quantization offers a practical solution by reducing numerical precision without requiring model retraining. Yet, its impact on deep convolutional TSC models remains unexplored.

In this paper, we investigate the effect of post-training quantization on InceptionTime based architectures for TSC across three benchmark archives: UCR (univariate) [4], UEA (multivariate) [1], and MONSTER (mix, large data) [5]. We conduct a quantization sensitivity analysis by emulating uniform quantization over a wide range of precisions, from 32-bit to 2-bit representations. In addition, we evaluate a practical deployment scenario based on half-precision floating-point weights. This dual evaluation enables the assessment of both the theoretical limits of reduced numerical precision and the realistic deployment of quantization strategies.

Our results provide a study on the robustness of the models with reduced numerical precision. We identify precision regimes that preserve classification performance while reducing weights precision and identified limits leading to significant degradation. Our work demonstrates that reductions in numerical precision can be achieved without sacrificing predictive performance, providing practical guidelines for the efficient deployment of deep TSC models and new insights into the relationship between quantization and performance in convolutional architectures for time series analysis.

2 Related Work

This section reviews the methodologies proposed over the last decade for both Time Series Classification (TSC) and model quantization.

2.1 Time Series Classification

Since the release of the UCR archive [4], a significant amount of research has focused on advancing the state of the art in TSC. Early approaches adapted machine learning techniques originally developed for tabular data to account for the temporal ordering in time series. Notably, several methods relied on the nearest neighbor algorithm using time-series-specific similarity measures.

As the field matured, researchers increasingly proposed methodologies specifically designed for time series data rather than adapting existing tabular-data techniques. The taxonomy introduced in [20] categorizes TSC algorithms into

eight main families: distance-based, feature-based, dictionary-based, convolution-based, interval-based, shapelet-based, hybrid-based, and deep-learning-based methods [14]. The increasing availability of large-scale TSC datasets, particularly through the multivariate UEA archive [1] and the recently introduced MONSTER archive [5], has further accelerated research on deep learning approaches. Beginning with relatively simple convolutional neural network architectures such as FCN and ResNet [24], increasingly sophisticated models have been proposed to improve classification performance. Examples include InceptionTime [15] and its extension H-InceptionTime [13] and their lighter version LITETime [12]. Transformer-based architectures have also gained increasing attention for TSC, notably ConvTran [7] by combining convolutions and self-attention mechanism.

2.2 Model Quantization

Model quantization is a compression technique that reduces the numerical precision of a model's parameters, typically by converting floating-point values into lower-bit integer or float representations. The roots of quantization [22] can be traced back to signal processing where it was used to map continuous values to discrete levels for efficient transmission of information, digital storage, fixed-point arithmetic and embedded systems [25, 21]. With the significant increase of work around machine learning, researchers questioned whether reducing model complexity while preserving performance is a possible solution for model deployment. This is where quantization played a key role for reducing memory footprint, inference latency and energy consumption while preserving performance for models such as decision tree [23].

Early research on quantization in deep learning has addressed CNNs, where [10] demonstrated minimal accuracy degradation for 8-bit integer inference compared to full-precision models. Moreover, binarized neural networks [11] demonstrated effective usage in model deployment on resource-constrained devices. Subsequently, two major methodologies emerged: Quantization Aware Training (QAT) and Post-Training Quantization (PTQ). QAT applies quantization methods during the training stage of the model, while PTQ quantized a pre-trained model without re-training. While QAT, in theory, achieves better performance, PTQ has gained significant attention due to its lower computational cost and easy deployment [16].

With the success of transformer-based architectures in natural language processing and Large Language Models (LLMs), its impact on quantization research has been significant. Modern LLMs contain billions of parameters, making it challenging to deploy them because of the limited memory and computational limitations of a significant amount of hardware. In order to address this issue, recent PTQ methods such as OPTQ [8], AWQ [17] and SmoothQuant [26] have been proposed and demonstrated that transformer-based models can be compressed to 8-bit or even 4-bit precision with minimal performance degradation. More recently, 1-bit precision LLMs have been proved possible with the BitNet family models [18].

Today, quantization remains one of the most widely adopted model compression strategies for both deep learning and LLM deployment. Current research focuses on reducing precision further while preserving model accuracy, robustness, and reasoning capabilities, making quantization a critical enabler for deploying increasingly large models on edge, mobile, and resource-constrained platforms [9].

3 Methods

In this work, our objective is to evaluate the impact of reducing the numerical precision of deep convolutional neural network weights for TSC. For this purpose, we propose to approach the problem from the post-training quantization point of view. We focus on weight quantization, as it is the primary contributor to memory and computational overhead in deep convolutional architectures. We propose two evaluation methods which are philosophically opposed. First, a theoretical strategy is used to systematically evaluate the accuracy difference related to the number of bits used to represent the weights. Second, a more realistic and application oriented approach, using half-precision (16-bits) floats in mixed precision policies pipeline. This second view allows us to evaluate performance change on accuracy, but also on computational time required.

3.1 Uniform quantization

The first method we propose is to use a uniform quantization with a single scaling factor per layer. Uniform quantization maps real-valued weights to a discrete set of integer values, scaled to fit a predefined range. This quantization method assumes a zero centered distribution, scaling the weights to fit a symmetric range around zero. The quantization is applied by the function Q as defined in equation 2, where b denotes the number of bits used and s_b the computed scaling factor.

$$s_b = \frac{\max |x|}{2^{b-1} - 1} \tag{1}$$

$$Q_b(x) = \left\lfloor \frac{x}{s_b} \right\rceil s_b \tag{2}$$

Where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

We only apply this quantization to the convolution kernel weights which represent 98.5% of the model weights. For this uniform integer quantization, we apply the same bit-width to all convolutional layers of our model. Practically, this quantization scheme is implemented using a simulated quantization before inference, as described in Figure 1 where the quantized values are computed and returned using full precision arithmetic.

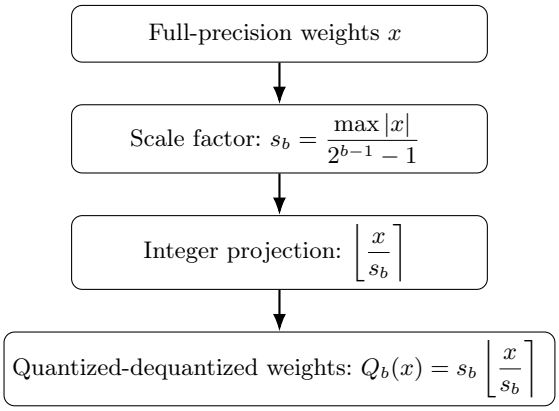

Fig. 1. Uniform post-training quantization of full-precision weights. Weights are scaled, rounded to the nearest integer, and then mapped back to floating-point values using the same scale factor.

3.2 Half-precision and mixed precision policy

In addition to the integer based uniform quantization, we evaluate mixed-precision policy with half-precision floating-point (FP16) quantization, which is supported in modern hardware. FP16 reduces memory usage by 50% compared to 32-bit floating-point (FP32) while maintaining a floating-point representation, which can better preserve the dynamic range of the weights. We evaluated 2 parameters, each with 2 choices making 4 different cases of precision policy.

The first parameter is the type of floating-point representation used. The possible choices are the classical FP16 and its brain floating point (BF16) counterpart [2]. Compared to FP16, BF16 used the same number of bits but is a shorter version of FP32 bits, result of a truncate operation. This choice makes the BF16 format to keep the dynamic of FP32 which makes it simpler to convert, but also does not require loss scaling when training models. On the other hand, the FP16 is more precise for small value changes.

The second parameter is the use of mixed precision mode for activations. In mixed precision mode, the weights are stored as FP32 weights (sometimes referred as master weights) but the computations are made using half-precision (FP16 or BF16). In our experiments, half-precision quantization is applied to all the model weights when mixed precision is disabled, and inference is always performed using half-precision arithmetic. This allows us to accurately measure the impact of the different variations of the reduced precision.

Unlike our theoretical uniform quantization approach, this evaluation makes it a practical choice for deployment scenarios. Quantized models are evaluated on the UCR, UEA, and MONSTER benchmarks without any fine-tuning or retraining. Half-precision and mixed precision policies are implemented using TensorFlow.

4 Experiments

In this section, we present the experimental setup and results obtained from evaluating the post-training quantization methods detailed in the previous sections. First, we evaluate the weight precision reduction as a search for a practical limit. The aim is to quantify how far it is possible to achieve a reasonable accuracy with limited bits per weight. Second, we evaluate the weight precision reduction using half-precision floating point, also providing evaluation time to show the potential speed gain. Finally, we provide an insight on the potential speed gain using the mixed precision quantization policy during the training phase.

4.1 Experimental Setup

We conduct our experiments on three TSC benchmarks. For univariate TSC tasks, we used the 128 datasets from the UCR archive [3]. For multivariate TSC tasks, we used the 26 datasets from the UEA archive [1]. For larger dataset, we use 27 datasets from the MONSTER archive [5], excluding the two variations of the S2Agri dataset already represented in S2Agri-10pc. This exclusion is done for reasons of time complexity as S2Agri represents more than 60% of the MONSTER archive.

For each dataset, we implement a consistent experimental protocol to ensure fair and comparable results. All our experimentations have been run on the same computer equipped with an RTX 4090 and using TensorFlow/Keras for the deep learning backend. We systematically evaluate each quantization method using the Inception [15] architecture. We use a single model rather than an ensemble of models with default hyperparameters. The hyperparameters used for all our experiments are summarized in Table 1. Concerning the number of epochs used to train the model on the MONSTER archive, Inception is trained for 100 epochs on the datasets that require less than a minute per epoch and 10 epochs the rest of the datasets.

4.2 Experimental Results

In this section, we present the experimental results obtained from the different quantization and mixed-precision strategies applied. Each part of this section focuses on either the predictive performance degradation or the computational time improvement. This allows us to characterize the trade-off between accuracy degradation and speed gain.

Affine quantization. Figures 2 and 3 report the accuracy variation induced by affine quantization across different integer bit-width configurations, for both the $32 \rightarrow 12$ and $12 \rightarrow 2$ regimes. Overall, the results show that the model is highly robust to moderate quantization levels.

For the higher precision regime ($32 \rightarrow 12$, figure 2), the degradation is essentially negligible. The mean accuracy variation at 12 bits is $< 0.01\%$, with a

Hyperparameter	**Value / Description**
Batch size	$\min\left(\frac{\lvert D\rvert}{10},\ 16\right)$ where $\lvert D\rvert$ is the number of training samples
Epochs	2000 (UCR/UEA) 100 (16/MONSTER) 10 (11/MONSTER)
Model selection	Best model selected based on training set performance
Learning rate schedule	Reduce on plateau: factor $= \frac{1}{2}$ patience $= 50$ $\min_{LR} = 10^{-4}$

Table 1. Training hyperparameters and settings.

maximum deviation of only 0.44%. Interestingly, even for $12 \rightarrow 8$ bits, the mean remains $< 0.01\%$, indicating no systematic loss in performance. Moreover, we note that the amount of datasets with improved performance is almost the same as the amount of datasets with degradation in performance. These differences are mostly due to noise introduced by minor weights changes.

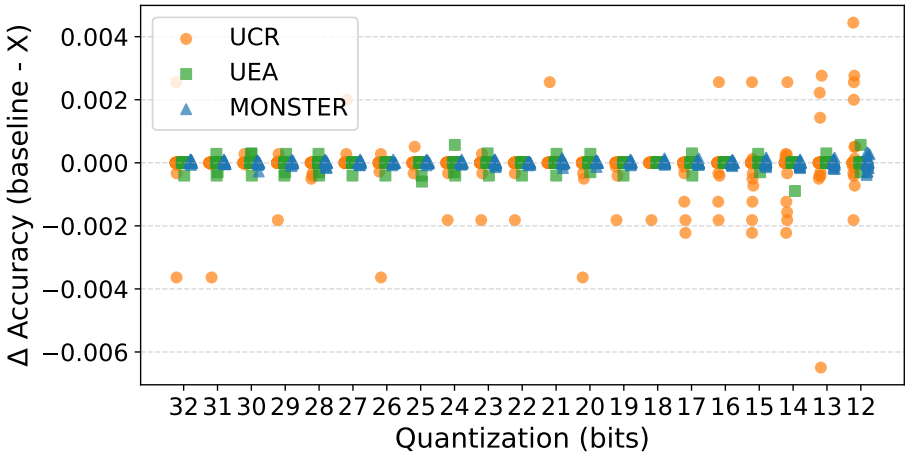

Fig. 2. Accuracy drop compared to baseline using uniform affine quantization.

For lower bit widths (Figure 3), degradation becomes progressively more significant. At 5 bits, the mean change is still below 1%, while at 4 bits the mean remains relatively small (4%) but with a significantly larger maximum

drop (67.8%), indicating dataset dependent instability. At 2–3 bits, performance becomes highly unreliable, with substantial degradation.

These results confirm that accuracy remains stable (within 1%) down to approximately 5-bit quantization, and that no more than 1% average variation is observed for bit-widths ≥ 5, while stability at the dataset level is strongly reduced below this threshold. From a safety point of view, 12-bit quantization is a safe approach with a 1% worst case degradation.

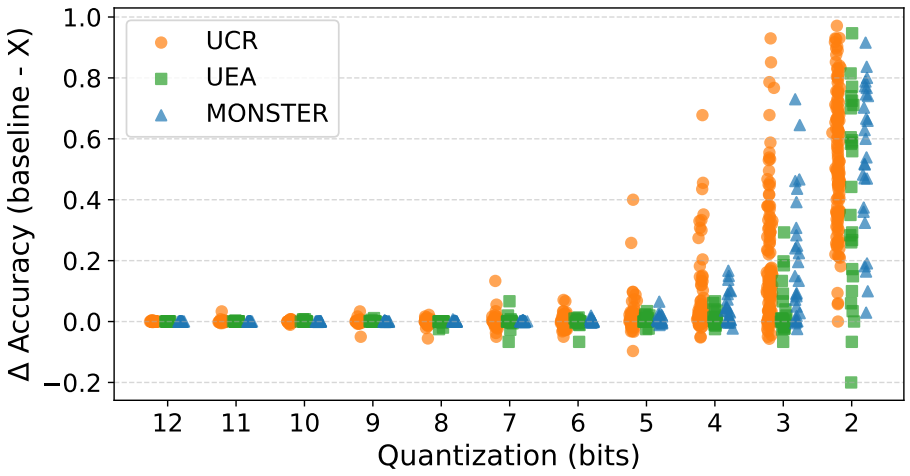

Fig. 3. Accuracy drop compared to baseline using uniform affine quantization.

Half-precision quantization. We then evaluate half-precision strategies using floating-point formats and mixed precision policies (Figure 4). In contrast to low-bits integer quantization, these approaches do not introduce a consistent degradation pattern.

Across all used configurations, the observed accuracy differences relative to the baseline are minimal, with average difference being close to zero. This suggests that 16-bits floating point inference preserves predictive performance almost entirely, even when aggressive precision reduction is applied. These results are consistent with previous results. No systematic accuracy loss is observed, and the small fluctuations are consistent with numerical noise rather than structural degradation.

Moreover, mixed-precision policies do not provide any measurable accuracy advantage over their corresponding full half-precision counterparts. Although FP16 appears marginally more stable than BF16, the observed differences remain negligible in practice. Consequently, from an accuracy perspective, mixed-precision policies do not appear to offer a compelling benefit for the considered

TSC benchmarks, as similar predictive performance can be achieved using a full half-precision representation.

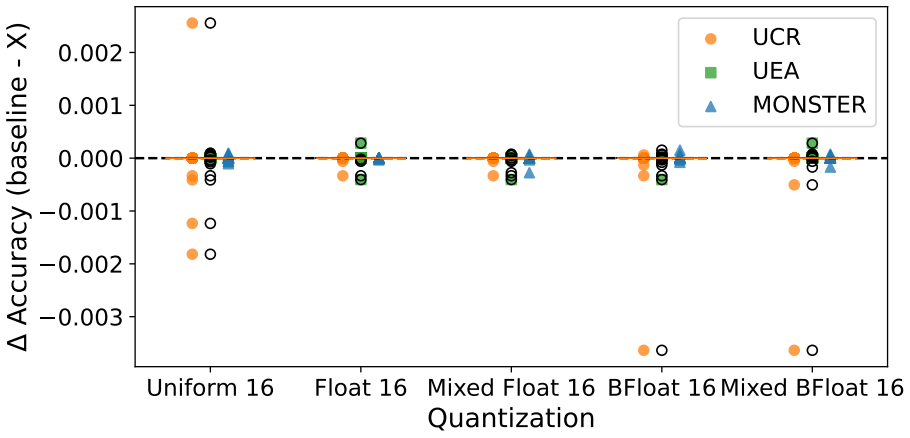

Fig. 4. Accuracy drop compared to baseline using half-precision policies.

Half-precision evaluation time. Figure 5 reports the percentage of speed improvement during evaluation under different half-precision strategies (FP16, BF16, and their mixed-precision variants). The results reveal a heterogeneous behavior across datasets.

While the mean improvements are positive, ranging from approximately 6.0% (BF16) to 17.4% (FP16), the distributions are extremely wide. In particular, all methods exhibit strong negative minimum values (down to approximately −70%), indicating cases where inference becomes significantly slower than the baseline. Conversely, extreme positive outliers exceeding 300% speedup suggest that certain configurations benefit disproportionately from hardware acceleration or favorable memory patterns.

Mixed-precision strategies do not outperform pure half-precision. For instance, mixed BF16 shows a slightly negative median (−2.0%), indicating that half of the datasets actually experience a slowdown. This highlights that evaluation time gains are not stable across datasets and depend heavily on runtime and hardware efficiency characteristics rather than model behavior alone.

Half-precision training time (MONSTER archive). Finally, Figure 6 reports the training time improvement obtained with half-precision strategies on the MONSTER archive, expressed as a percentage relative to the baseline training time.

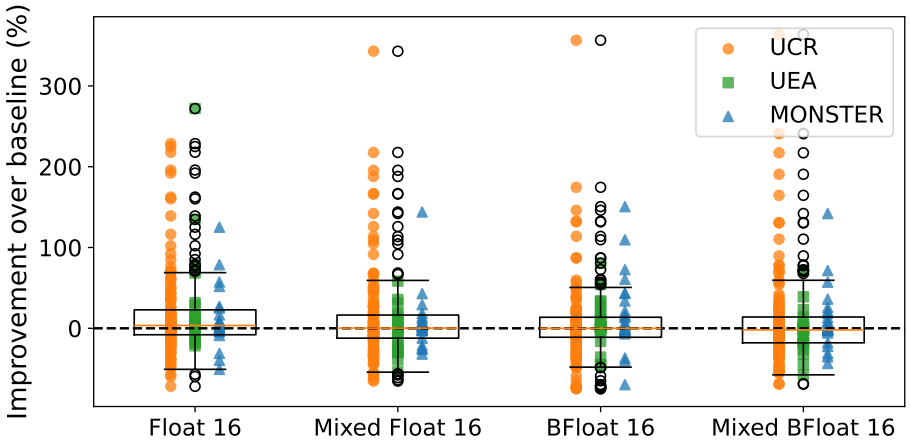

Fig. 5. Percentage of speed improvement for evaluation using half-precision strategies. Under the dashed line signify slower than baseline.

In contrast to the evaluation time results, the training speed improvements exhibit a much clearer pattern. FP16-based policies consistently accelerate training across all considered datasets. Standard FP16 achieves an average speedup of 37.8%, while even the worst observed case still benefits from a 20.7% reduction in training time. Mixed FP16 also provides significant gains, with an average improvement of 31.0%. BF16-based policies do not benefit from the same improvement in our experiments. Both standard and mixed BF16 exhibit negative average speedups (-1.8% and -4.6%, respectively), indicating a slight slowdown compared to the baseline. The mixed policies do not outperform the full half-precision configuration. This observation is consistent with the accuracy results presented previously, where mixed precision did not provide any measurable advantage.

In this experiment, FP16 appeared to be faster during training. However, this observation should be interpreted with caution, as we only conducted accuracy evaluation on the post-training quantization and did not specifically optimize or investigate half-precision trained model behavior. Moreover, the corresponding training runs showed poor convergence and predictive performance.

A detailed analysis of these effects was outside the scope of this work, which primarily focuses on post-training evaluation efficiency. Thus, those results will not be discussed in the following discussion and conclusion section as this is not the purpose of this paper.

Overall, our results indicate that post-training quantization with FP16 provides the most favorable trade-off between efficiency and accuracy.

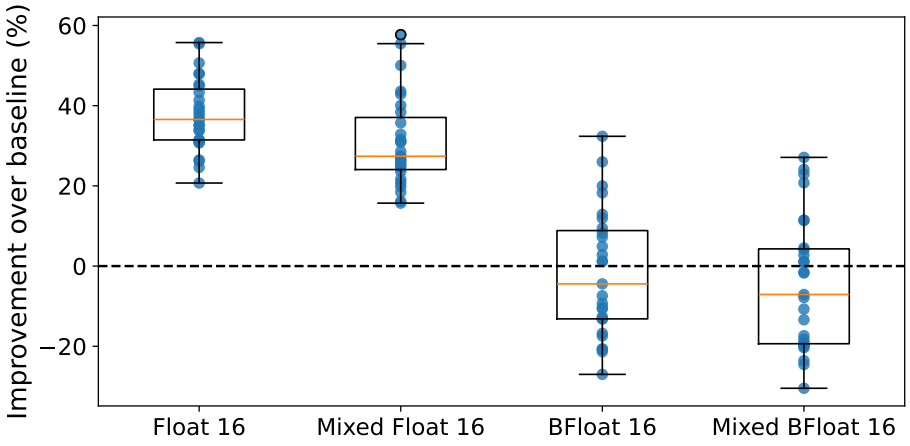

Fig. 6. Percentage of speed improvement for training using half-precision strategies for MONSTER archive.

4.3 Discussion

The experimental results reveal key insights into the practical application of quantization for time series classification (TSC).

First, the Inception architecture is robust to weight precision reduction. For bit-widths down to 5-bits, the average accuracy degradation remains below 1%, confirming that moderate quantization can be applied without significant performance loss. The larger degradation and higher variance below 5-bits suggests a practical lower bound for stable performance. This threshold effect is consistent with findings in other domains, where very low precision often requires specialized techniques. While we cannot define 5-bits as a general limit, we can assume that a limit close to this range remains an ideal spot for TSC with the considered architecture.

The evaluation of half-precision formats shows negligible accuracy loss across all configurations, confirming that half-precision floating-point is sufficient for TSC inference. However, the evaluation speed improvements seems small compared to baseline. The heterogeneous evaluation time results further emphasize that quantization benefits are dataset and context specific.

Contrary to expectations, mixed-precision policies do not outperform their uniform half-precision counterparts in terms of accuracy or speed. Those configurations show similar or negative accuracy and speed improvements. The lack of consistent benefits, combined with increased implementation complexity, suggests that mixed-precision is not currently advantageous for TSC.

These results demonstrate that reduced-precision strategies offer a practical way of reducing model memory footprint without sacrificing accuracy. The robustness of the Inception architecture to quantization suggests that TSC tasks may be particularly prone to precision reduction. However, the hardware depen-

dent nature of floating-point performance and the heterogeneous speed improvements across datasets highlight the need for careful, context specific evaluation.

5 Conclusion

This work presents a comprehensive evaluation of post-training quantization and mixed-precision strategies for TSC, using datasets from the UCR, UEA, and MONSTER archives. Our empirical results demonstrate that quantization is viable down to 5 bits with less than 1% average accuracy loss, while 12-bit quantization ensures robustness with negligible degradation. FP16 emerges as the most effective half-precision format, delivering both evaluation and training speedups with no measurable accuracy loss. Mixed-precision policies, overall, do not provide clear advantages over uniform half-precision for TSC given current hardware. In practice, FP16 represents an optimal choice for TSC with our setup, offering the best trade-off between efficiency and accuracy.

As a final contribution, in an effort to promote transparency and support future work, all of our experimental results and source code are publicly available at https://github.com/MSD-IRIMAS/QTZ4TSC. This ensures the reproducibility of our study and allows the community to use and verify our results.

Future work could explore several promising directions, such as extending the evaluation to additional architectures like LITE, investigating the impact of extended training on the MONSTER archive, or assessing more aggressive quantization schemes, particularly int8 and int4 as used in large language models community. Additionally, evaluating these methods on real resource constrained systems would provide valuable insights into their applicability in real world, low power environments.

Acknowledgments. This work was supported by a welcome contract funding from the Université de Haute-Alsace. This work was supported by the ANR Terminus project (ANR-25-CE23-2879) of the French Agence Nationale de la Recherche. We acknowledge the High Performance Computing Center of the University of Strasbourg for supporting this work by providing scientific support and access to computing resources. Part of the computing resources were funded by the Equipex Equip@Meso project (Programme Investissements d'Avenir) and the CPER Alsacalcul/Big Data.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The UEA multivariate time series classification archive, 2018 (Oct 2018). https://doi.org/10.48550/arXiv.1811.00075
2. Burgess, N., Milanovic, J., Stephens, N., Monachopoulos, K., Mansell, D.: Bfloat16 Processing for Neural Networks. In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH). pp. 88–91 (Jun 2019). https://doi.org/10.1109/ARITH.2019.00022
3. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (Nov 2019). https://doi.org/10.1109/JAS.2019.1911747
4. Dau, H.A., Keogh, E., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., Hexagon-ML: The UCR time series classification archive (Oct 2018)
5. Dempster, A., Foumani, N.M., Tan, C.W., Miller, L., Mishra, A., Salehi, M., Pelletier, C., Schmidt, D.F., Webb, G.I.: Monster: Monash scalable time series evaluation repository. arXiv:2502.15122 (2025)
6. Dempster, A., Tan, C.W., Miller, L., Foumani, N.M., Schmidt, D.F., Webb, G.I.: Highly Scalable Time Series Classification for Very Large Datasets. In: Lemaire, V., Ifrim, G., Bagnall, A., Guyet, T., Malinowski, S., Schäfer, P., Tavenard, R. (eds.) Advanced Analytics and Learning on Temporal Data. pp. 80–95. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-77066-1_5
7. Foumani, N.M., Tan, C.W., Webb, G.I., Salehi, M.: Improving position encoding of transformers for multivariate time series classification. arXiv preprint arXiv:2305.16642 (2023)
8. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: OPTQ: Accurate quantization for generative pre-trained transformers. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=tcbBPnfwxS
9. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-power computer vision, pp. 291–326. Chapman and Hall/CRC (2022)
10. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016), https://arxiv.org/abs/1510.00149
11. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 4114–4122. NIPS'16, Curran Associates Inc., Red Hook, NY, USA (Dec 2016)
12. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Look into the LITE in deep learning for time series classification. International Journal of Data Science and Analytics (Jan 2025). https://doi.org/10.1007/s41060-024-00708-5
13. Ismail-Fawaz, A., Devanne, M., Weber, J., Forestier, G.: Deep learning for time series classification using new hand-crafted convolution filters. In: 2022 IEEE International conference on big data (Big Data). pp. 972–981. IEEE (2022)
14. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: A review. Data Mining and Knowledge Discovery **33**(4), 917–963 (Jul 2019). https://doi.org/10.1007/s10618-019-00619-1

15. Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (Nov 2020). https://doi.org/10.1007/s10618-020-00710-y
16. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2704–2713 (2018)
17. Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.M., Wang, W.C., Xiao, G., Dang, X., Gan, C., Han, S.: Awq: Activation-aware weight quantization for on-device llm compression and acceleration. Proceedings of machine learning and systems **6**, 87–100 (2024)
18. Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., Wei, F.: The era of 1-bit llms: All large language models are in 1.58 bits. arXiv preprint arXiv:2402.17764 (2024)
19. Middlehurst, M., Bagnall, A.: Extracting Features from Random Subseries: A Hybrid Pipeline for Time Series Classification and Extrinsic Regression. In: Ifrim, G., Tavenard, R., Bagnall, A., Schaefer, P., Malinowski, S., Guyet, T., Lemaire, V. (eds.) Advanced Analytics and Learning on Temporal Data. pp. 113–126. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-49896-1_8
20. Middlehurst, M., Schäfer, P., Bagnall, A.: Bake off redux: A review and experimental evaluation of recent time series classification algorithms. Data Mining and Knowledge Discovery **38**(4), 1958–2031 (Jul 2024). https://doi.org/10.1007/s10618-024-01022-1
21. Padgett, W.T., Anderson, D.V.: Fixed-Point Signal Processing. Synthesis Lectures on Signal Processing, Morgan & Claypool Publishers (2009). https://doi.org/10.2200/S00220ED1V01Y200909SPR009
22. Shannon, C.E.: A mathematical theory of communication. The Bell system technical journal **27**(3), 379–423 (1948)
23. Shi, Y., Ke, G., Chen, Z., Zheng, S., Liu, T.Y.: Quantized training of gradient boosting decision trees. Advances in neural information processing systems **35**, 18822–18833 (2022)
24. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 1578–1585 (May 2017). https://doi.org/10.1109/IJCNN.2017.7966039
25. Widrow, B., Kollár, I.: Quantization noise: roundoff error in digital computation, signal processing, control, and communications. Cambridge University Press (2008)
26. Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., Han, S.: Smoothquant: Accurate and efficient post-training quantization for large language models. In: International conference on machine learning. pp. 38087–38099. PMLR (2023)