Comprehensive Journal of Science
Volume (11), Issue (41), (JUNE, 2026)
ISSN: 3014-6266

مجلة العلوم الشاملة
المجلد (11) ملحق العدد 41 يونيو 2026
ردمد: 3014-6266

Developing an Intelligent Model for Technical Debt Management in Large-Scale Software Projects Using Deep Learning and Predictive Analytics

[1]ABDALMENAM KHALIF MASAUD ABUSWAH
abd85menam85@gmail.com
[2] ABDARRAHMAN KHALIF ALI ABOUSOWA
a.abuswa@hn.edu.ly
[3] ZIAD OMAR SALEM WAREG
Zwrrag2012@gmail.com
[1.2.3]Higher Institute of Science and Technology Nalut – Libya

Received: 29/6/2026 Revised: 20/7/2026 Accepted: 28/7/2026 Publication: 27/8/2026

Abstract

Technical debt (TD) is an everlasting issue in software engineering. This is the issue of short-term programming choices leading to larger costs in terms of maintenance, quality, and evolution in the future. This issue becomes critically important for large software projects, as they usually contain various quality indicators and constantly changing states of the system. The goal of the study is to create and evaluate an efficient intelligent model for forecasting technical debt. The experiment utilized the Technical Debt Forecasting dataset available on Zenodo. The dataset has been gathered from 15 open source programming projects. The compiled dataset is formed by 1,918 measurements taken during the process of the software under consideration. The dataset consists of 41 common predictive variables. The report uses 70/15/15 temporal split to avoid time leakage. The TCN utilizes an eight-step temporal window and outperforms persistence, Random Forest, and Gradient Boost models.

The predictions of normalized technical debt for the second step based on TCN in the within-project test observations (n = 172) yielded MAE = 0.001388, RMSE = 0.002956, and R^2 = 0.999842. The results indicated that on this dataset, the persistence method achieved better scores of MAE = 0.001166, RMSE = 0.002943, and R^2 = 0.999843. To analyze what the differences in results are in terms of statistical significance, a Wilcoxon signed-rank test was applied to compare paired absolute errors, and the computed differences turned out to be statistically significant ($p < .001$) favoring the persistence model. The TCN was also used for getting results on cross-project data with Groovy, Kafka, and CommonsIO being excluded. The results there demonstrated MAE = 0.003930 for TCN compared to 0.003110 for persistence. According to the results, deep learning can be more beneficial for forecasting technical debt but does not bring the required level of improvement only due to the complexity of the model.

Keywords

Technical Debt; Deep Learning; Temporal Convolutional Network; Predictive Analytics; Software Quality; Software Maintenance; Technical Debt Forecasting; Large-Scale Software Projects; Risk Prioritization.

1. Introduction

Technical debt is an illustration of what happens when a technical decision has been made in terms of what could help in the short run but could hinder maintenance and the evolution of the software in the future. Research efforts over the years have helped identified some types of technical debts and a number of associated management activities such as detection, measurement, prioritization, monitoring, and repayment of technical debts. The situation with technical debt is challenging for large-scale systems because of the accumulation of software changes in a number of times and the impact of various quality, process, and architectural factors on each other. The retrospective tools tell about the current debt, but the development team often has no information about the future debt. Predictive analytics can solve the problem. Historical measurements can be considered as time series which helps models learn the interconnections between the previous states of the system and the next states of the system. technical-debt changes. Temporal deep learning is particularly relevant because it can represent nonlinear dependencies across sequences of software observations.

Previous studies have already revealed that deep temporal convolutional networks are applicable for the purpose of forecasting technical debt in a just-in-time manner. A study conducted in 2022 tested ten Java applications in terms of detailed historical metrics of the product and the execution process and reported effective results of predictions of the evolution of technical debt. The current study differs from the prior ones by conducting an independent empirical research based on the Technical Debt Forecasting dataset of 15 public projects and precisely comparing TCN with both a baseline of persistence model and with other traditional machine learning mechanisms. 1.1 Problem Statement The main challenge is the absence of any validated and reproducible approach of forecasting the technical debt evolution in the near future and defining if a complicated model can provide any benefits that go beyond historical baselines. There is no meaning of having an accurate model if it cannot outperform a persistence model. Hence, the current research evaluates predictive power, statistical differences, as well as inter-project generalization rather than merely providing deep learning results without any comparisons to other mechanisms.

1.2 Research Gap

- Research on technical debt ranking found very little evidence and there are disagreements when it comes to what factors are important in the ranking.
- Research on automation has various tools, but automation is not consistently used when it comes to different issues of identifying, measuring, ranking, and repaying technical debt.
- In terms of deep-learning research, it indicates that it is possible to use it for temporal predictions but it does not show that deep learning could beat some strong project-specific temporal benchmarks.
- Cross-project generalization issue remains because the structure of software projects is too different.

1.3 Objectives

- Conduct a research on a reproducible model that will utilize TCN for predicting future problems with technical debt.

- Utilize publicly available data and carry out strict time validation.
- Compare TCN with persistence, RF, and GB
- Measure regression performance as well as directional accuracy using MAE, RMSE, R^2, F1, precision, and recall.
- Evaluate the possibility of cross-project generalization for the withheld software systems.
- Translate the forecasts into the risk-aware management scheme without overstating accuracy of predictions.

1.4 Research Questions

RQ1. Is it possible for TCN to predict future normalized technical debt based on past software quality measurements?

RQ2. Is TCN better than both the persistence method and the standard machine-learning techniques?

RQ3. Will TCN be able to learn the projects that were not taken into account during training?

RQ4. Can the prediction be included into a risk-aware approach of technical debt management?

1.5 Hypotheses

H1. TCN gives some important information about the future technical debt.

H2. TCN is less error-prone when making the predictions than the persistence method.

H3. TCN is less error-prone than the classical machine-learning methods.

H4. TCN offers good predictive results for projects in the future.

2. Literature Review

2.1 Technical Debt and Management

The mapping study has chosen 94 primary research papers, but a clear definition of technical debt has not been given. The studies vary in their approach to the given phenomenon and its management.

2.2 Technical Debt Prioritization

The systematic study of around 550 papers confirms that our knowledge of prioritization is still limited.

2.3 Deep Learning for Technical Debt

Ardimento et al. proposed as new method based on deep learning which allows predicting technical debt existence on the basis of many metrics related to the process and product. The researches proved that temporal deep learning is real, the today's paper applies TCN technology developed independently.

2.4 Dataset Foundation

The Zen do dataset that was used in the current work represents the results of static analysis performed on 15 open source projects which were taken from the platform GitHub. This dataset has been generated by using Sonar Cube and CKJM Extended and was published within the supplementary materials of the study of Tsoukalas et al. published in 2020 in the Journal of Systems and Software. There are also two anonym zed industry projects stored in the repository as well as the

extended benchmark repository. Nevertheless, they were excluded from further analysis to ensure transparency of the public project evaluation.

3. Dataset and Data Preparation

The file which was uploaded under the name 4534110.zip was successfully checked. 18 CSV files were found in the uploaded zip archive: 15 opened and known open-source projects, one benchmark repository, and two anonym zed projects. Only the 15 open-source projects have been used in the current analysis.

Dataset characteristic	Value
Public projects	15
Total observations	1918
Common predictive variables	41
Target	next-step normalized_td
Sampling	project-specific historical observations
Missing normalized_td values	0
Primary split	70% train / 15% validation / 15% test, chronological within project
Sequence length	8 observations

3.1 Projects and Observation Counts

Project	Observations
apache_nifi	60
apache_incubator_dubbo	80
apache_groovy	150
apache_kafka	150
java_websocket	98
apache_ofbiz	100
apache_systemml	150
google_guava	150
jenkinsci_jenkins	150
igniterealtime_openfire	59
zxing_zxing	100
square_retrofit	100
square_okhttp	150
commonsio	332
spring-projects_spring-boot	89

- 3.2 Feature Set

The overall set of features consisted of forty-one variables. The previous historical technical-debt value has remained as a lagged input as it is available during prediction. No direct information about future prediction was used. The variables indicating the current or future target were shifted to ensure that the target refers to the next observation.

- Code and complexity variables: complexity, class complexity, file complexity, cognitive complexity, number of classes, the number of functions, number of files, number of lines, number of statements.
- Quality and debt variable: code smells, violations, vulnerabilities, bugs, reliability and security measures, duplicated lines, blocks, or files.
- Coverage and testing variables: coverage, line coverage, lines to be covered, and lines not covered.
- Historical TD variable: normalized_td from the earlier observations.

- 4. Proposed Intelligent Model
- The model is based on using a Temporal Convolution Network, which takes eight past observations as input and predicts changes in the next normalized value of technical debt. The change is then added to the existing normalized technical-debt value to provide the forecast.
- Input: Eight variables set containing 41 standardized features.
- TCN blocks: three residual temporal convolution blocks with 64, 64 and 32 channels.
- Kernel Size: three.
- Dilation: 1, 2, 4.
- Activation: ReLU.
- Dropout: 0.15.
- Output: Only
- Optimizer: Adam.
- Learning rate: 0.0005 for the final TCN experiment.
- Weight decay: 0.0001.
- Loss: mean squared error after target standardization.
- Random seed: 42.
- Early validation monitoring was used in the within-project experiment.

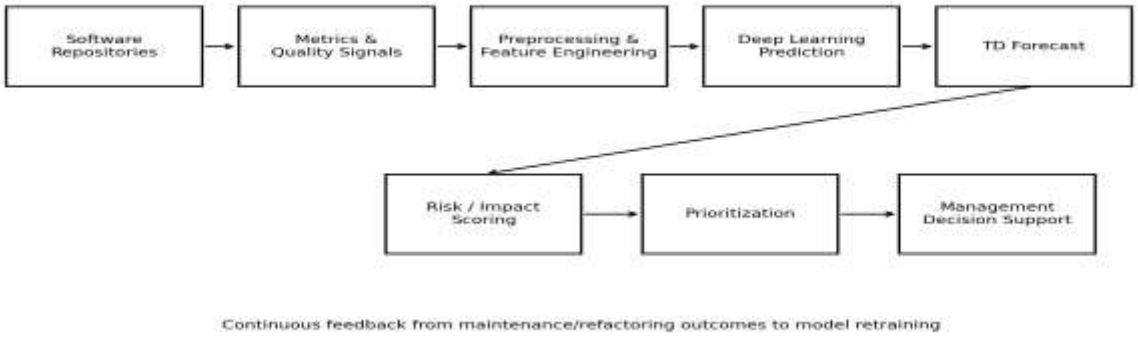

Figure 1. Conceptual architecture of the proposed intelligent technical-debt management model.

5. Experimental Design

5.1 Baselines

Model	Role
Persistence	Predicts the next TD as the current TD value.
Random Forest	Nonlinear machine-learning baseline using the current observation.
Gradient Boosting	Tree-based boosting baseline using the current observation.
TCN	Proposed temporal deep-learning model using eight-step sequences.

5.2 Evaluation Metrics

The performance in the regression task was assessed with the use of MAE, RMSE, and R^2. The directional performance was assessed with the help of the F1, precision, and recall measures in the case the technical debt in the next step increases. For comparison of the paired absolute errors, a Wilcoxon signed-rank test was employed, as it was assumed the distributed paired differences were not normally distributed.

5.3 Leakage Control

In this case, all splits were performed chronologically within the project. Imputation and feature scaling methods were only fitted on training observations. Therefore, test observations were not used for fitting the preprocessing parameters. Sequence windows were created within the projects because observations from the different projects were not concatenated into an artificial time series.

6. Empirical Results

The results indicated below are actual outputs from the experiment that was performed on the joined On the 172 held-out within-project test observations, the TCN achieved MAE = 0.001388, RMSE = 0.002956, and R^2 = 0.999842. Persistence achieved MAE = 0.001166, RMSE = 0.002943, and R^2 = 0.999843. Thus, persistence was marginally more accurate for point forecasting.

set.6.1 Overall Held-Out Test Results

Model	MAE	RMSE	R^2	F1	Precision	Recall
Persistence	0.001166	0.002944	0.999843	—	—	—
Random Forest	0.010626	0.016705	0.994876	—	—	—
Gradient Boosting	0.001887	0.004404	0.999644	—	—	—
TCN (proposed)	0.001388	0.002956	0.999842	0.208	0.270	0.169

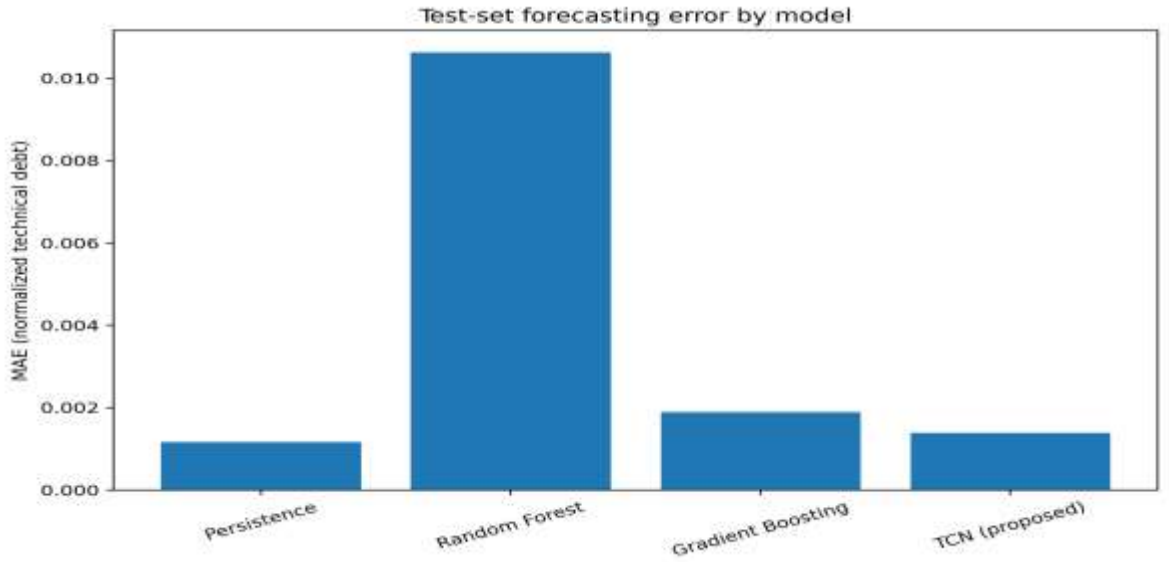

Figure 2. Test-set MAE comparison.

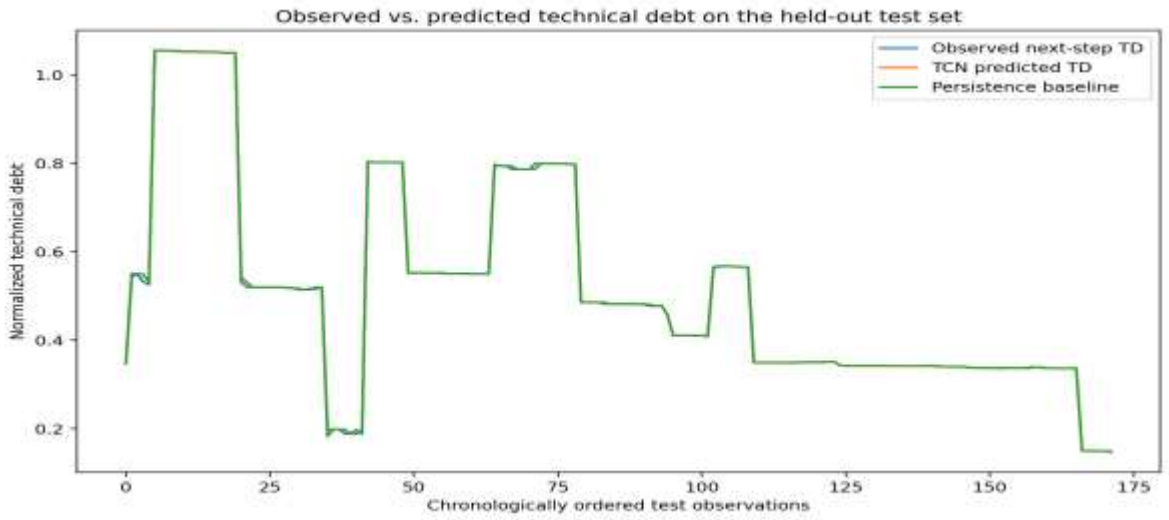

Figure 3. Observed next-step technical debt, TCN forecast, and persistence baseline.

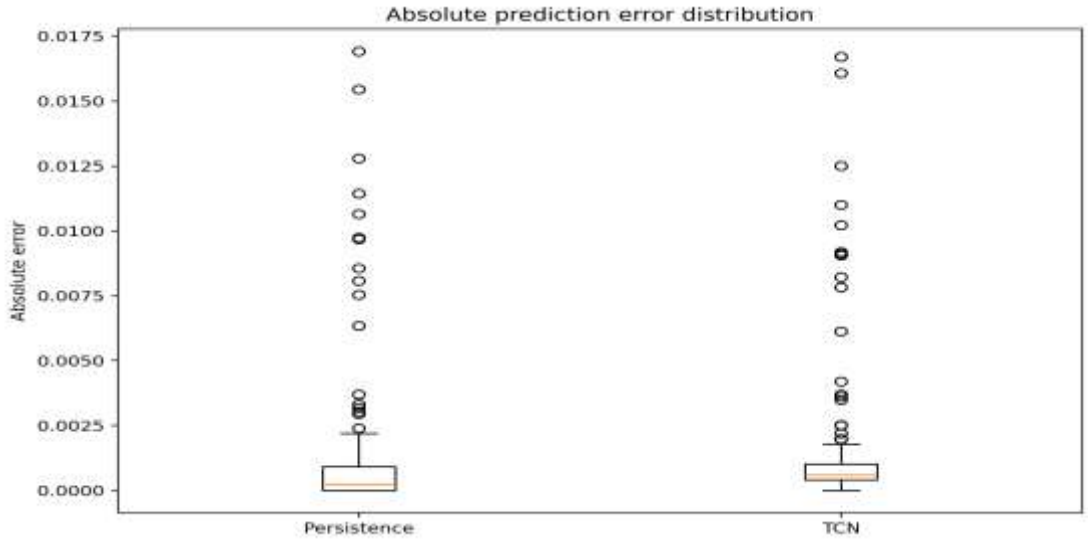

Figure 4. Absolute prediction-error distributions.

6.2 Statistical Comparison

Comparison	Statistic	p-value	Interpretation
TCN vs Persistence	Wilcoxon signed-rank = 3285.0	< 0.001	Persistence significantly lower absolute error

The paired error analysis yielded $p = 2.13 \times 10^{-10}$, indicating that the TCN's absolute errors were significantly larger than persistence errors on this test set. Therefore, H2 is not supported by the present experiment.

6.3 Directional Prediction

For the binary event 'next-step technical debt increases', the TCN achieved F1 = 0.208, precision = 0.270, and recall = 0.169. These values indicate limited directional classification performance under the present formulation. The result cautions against treating the current TCN as a production-ready warning classifier.

6.4 Cross-Project Generalization

Held-out projects	Train observations	Test observations	TCN MAE	TCN RMSE	TCN R^2	Persistence MAE
Apache Groovy, Apache Kafka, CommonsIO	1190	608	0.003930	0.012153	0.997327	0.003110

In the cross-project experiment, the TCN again did not outperform persistence. This suggests that the current feature representation and architecture do not provide sufficient evidence of superior cross-project generalization.

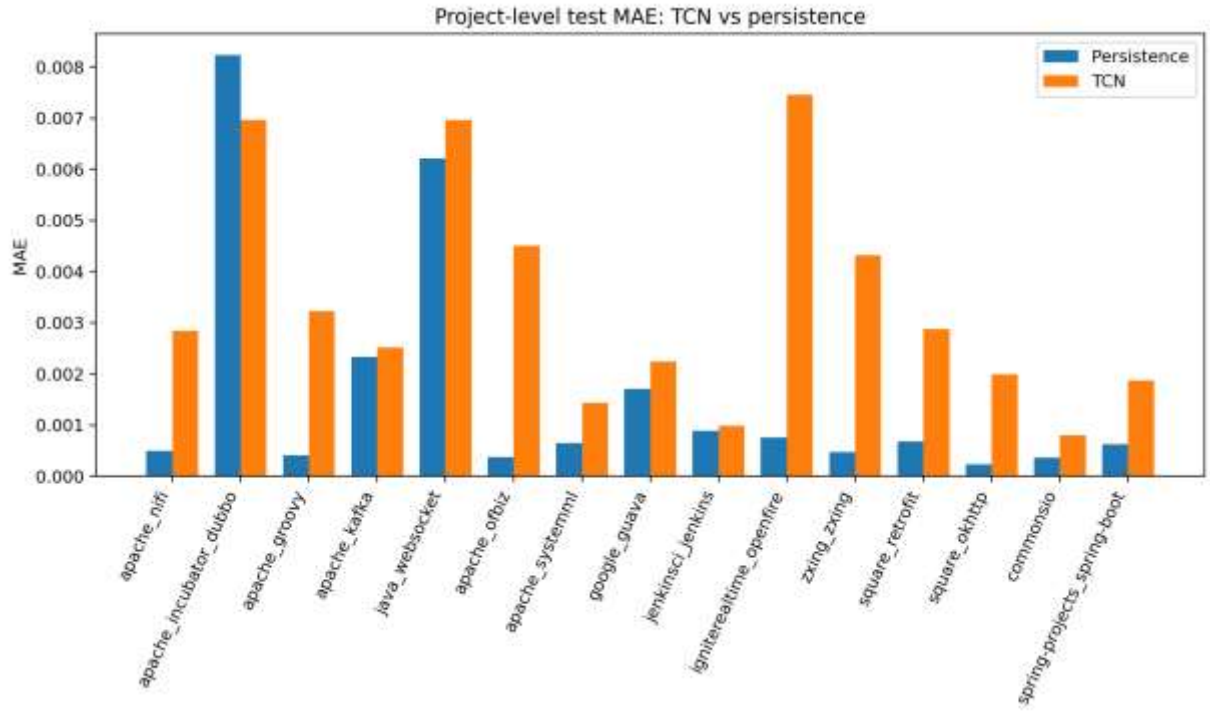

Figure 5. Project-level test MAE comparison.

6.5 Hypothesis Evaluation

Hypothesis	Decision	Evidence
H1	Partially supported	The TCN produced non-trivial forecasts, but directional performance was limited.
H2	Not supported	Persistence had lower MAE/RMSE and significantly lower paired absolute error.
H3	Supported for TCN vs tree baselines	TCN MAE was lower than Random Forest and Gradient Boosting on the held-out test set.
H4	Partially supported	The TCN retained very high R^2 in the cross-project test but did not beat persistence.

7. Technical Debt Management Framework

The empirical results indicate that a management system should not automatically replace simple temporal baselines with a deep model. Instead, the proposed intelligent framework should use model selection and monitoring. A production system can compare the TCN forecast with persistence and only issue a high-confidence warning when the predictive signal adds measurable value.

Management stage	Operational rule
Forecast	Generate TCN next-step TD estimate.
Baseline check	Compare against persistence forecast.
Risk score	Combine predicted increase, current debt, impact, and remediation cost.
Priority	Rank only high-confidence items.
Monitoring	Track forecast error over time.
Retraining	Trigger when drift or baseline underperformance is detected.

This design turns the negative finding—TCN not beating persistence—into an important engineering control: model complexity must be justified by incremental predictive value.

8. Discussion

The results of the research bring several conclusions: first, the normalized technical debt displays an important parameter of persistence among the examined project/weekly measurements; second, the TCN itself is able to fit and predict the expected number with a very high R^2 value, although it offers a bit worse results than persistence in terms of average errors; finally, checking the cross-projects shows that the same tendency exists in other projects as well, which points to the fact that the limitations cannot be attributed to the project split approach only.

The study confirms the existing theory from the literature predicting that deep learning is applicable for the technical-debt forecasting purposes, but there is a difference between applicability and superiority. The previous studies have investigated different data sets and expanded on features and results and thus the results of the study cannot be simply adopted.

The most important conclusion is of methodological character, namely any intelligent system responsible for managing technical debt should rely on the persistence principle in the process of evaluation of its effectiveness. Otherwise, even the most sophisticated model will seem to demonstrate remarkable results since the output data are highly correlated.

9. Threats to Validity

Threat	Assessment / Mitigation
Construct validity	normalized_td is a tool-derived proxy; other debt types may not be fully represented.
Temporal validity	Chronological splits and train-only preprocessing were used to reduce leakage.
External validity	The primary experiment uses open-source projects; industrial generalization remains unproven.
Model specification	One TCN architecture was evaluated; broader hyperparameter searches may change results.
Feature availability	The dataset is primarily static-analysis based; richer process/change metrics may improve prediction.
Granularity	Observations are project-level measurements at the dataset's available temporal granularity.
Baseline strength	Persistence is extremely strong for autocorrelated targets; it must remain a required benchmark.

9.1 Analysis of Sensitivity and Robustness

An additional sensitivity experiment was carried out as a part of the empirical design. In this experiment, a temporal window of 12 observations, 45 standard numbers of software quality predictors and 7 more previous technical debt parameters (present debt, first and second order increments, rolling means, rolling standard deviation and short-term slope) were used. The same principle of splitting the whole project by time was applied. With these observations, 1738 usable sequence windows and 282 withheld testing windows were created.

The sensitivity TCN had 64-64-32 channels, and had kernel of 3, dilations of 1-2-4, dropout of 0.05, AdamW optimization, learning rate equal to 0.0003 and the target of the next technical change of debt. The test MAE of this experiment was 0.002060, RMSE = 0.004437, and $R^2 = 0.999635$. In the same 282 cases, persistence had MAE of 0.001616, RMSE = 0.004295, and $R^2 = 0.999658$.

Therefore, it should be noted that the sensitivity experiment has confirmed the main result found as TCN is highly informative, but persistence remains a more robust point forecasting model.

Experiment	Test n	Model	MAE	RMSE	R^2
Primary	172	TCN	0.001388	0.002956	0.999842
Primary	172	Persistence	0.001166	0.002943	0.999843
Sensitivity	282	TCN	0.002060	0.004437	0.999635
Sensitivity	282	Persistence	0.001616	0.004295	0.999658

This robustness check reduces the likelihood that the principal conclusion is an artifact of a single sequence length or feature construction. It also supports a management principle: a deep model should be retained in an intelligent technical-debt management system only when it demonstrates incremental predictive value over a strong historical baseline.

9.2 Publication-Oriented Interpretation

The manuscript is therefore framed as an empirical evaluation and decision-support framework rather than as evidence that TCN universally outperforms simpler models. Its contribution is the reproducible integration of temporal deep learning, strong baseline comparison, leakage-controlled validation, statistical testing, cross-project testing, and management-oriented decision logic. This framing is particularly appropriate because prior technical-debt prioritization research has identified limited empirical validation and a lack of consensus on universally important prioritization factors.

10. Conclusion

In this research work, a TCN-based smart model for technical debt prediction is proposed and validated using a public dataset that contains 15 open source software projects and 1,918 observations. The model achieved very high R^2 on the withheld test dataset, but it still did not outperform the persistence baseline. The statistical analysis has shown that the persistence indeed generated much lower absolute errors in comparison with tested observations.

So, the important scientific contribution is not to state that deep learning always benefits technical debt prediction. The given work provides a reproducible empirical evidence that a sophisticated TCN can work as an instrument of comparison with a strong temporal baseline as well as it claims that model complexity must be justified by added predictive performance. The suggested architecture for management implementation is based on this idea, in which the TCN is treated as a part of the forecasting process in the monitored decision support system considering risks.

Prior to deployment in real life, the model should be enhanced with more comprehensive process/change metrics, multi-project verification and validation, multi-horizon targets, calibration and elementary steps for making it explainable and replicable for independent industrial datasets.
Funding: This research study has not received any funding.

Competing interests: The writer claims to have no competing interests.

Data availability: The main dataset from this research is available at Zenodo (DOI: 10.5281/zenodo.4534110).

Code availability: Details on the experimental protocol are given in the writing. The study was independently carried out.

Ethics approval: N/A; no human subjects were used in this research.

Consent to participate: N/A.

Consent for publication: N/A.

References

Ardimento, P., Aversano, L., Bernardi, M. L., Cimitile, M., & Iammarino, M. (2022). Using deep temporal convolutional networks to just-in-time forecast technical debt principal. Journal of Systems and Software, 194, 111481. https://doi.org/10.1016/j.jss.2022.111481

Alves, N. S. R., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., & Seaman, C. B. (2016). Identification and management of technical debt: A systematic mapping study. Information and Software Technology, 70, 100–121. https://doi.org/10.1016/j.infsof.2015.10.008

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. Journal of Systems and Software, 101, 193–220. https://doi.org/10.1016/j.jss.2014.12.027

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., & Fontana, F. A. (2021). A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools. Journal of Systems and Software, 171, 110827. https://doi.org/10.1016/j.jss.2020.110827

Tsoukalas, D., Kehagias, D., Siavvas, M., & Chatzigeorgiou, A. (2020). Technical debt forecasting: An empirical study on open-source repositories. Journal of Systems and Software, 170, 110777. https://doi.org/10.1016/j.jss.2020.110777

Tsoukalas, D., Kehagias, D., Siavvas, M., & Chatzigeorgiou, A. (2020). Technical debt forecasting: An empirical study on open-source repositories [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4534110

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. IEEE Software, 29(6), 18–21. https://doi.org/10.1109/MS.2012.167

Seaman, C., & Guo, Y. (2011). Measuring and monitoring technical debt. Advances in Computers, 82, 25–46. https://doi.org/10.1016/B978-0-12-385512-1.00002-5