Am J Neurodegener Dis 2026;15(2):16-24
www.AJND.us /ISSN:2165-591X/AJND0170702

Original Article

Optimizing Alzheimer's prediction with dental care: comparing robust deep learning models in health and retirement study of America

Hamed Taheri[1*], Farhan Musaie[2*], Nazanin Mohammadzadeh[3*], Parsa Tavassoli Naini[4], Nargol Novin[5], Omid Salimi[6,7], Alireza Amini[6,7], Hadis Malekpour[8], Parsa Panahiyan[9], Mahta Malek[10], Alireza Fathiazar[11], Reza Namadkolahi[12]

[1]Department of Dentistry and Implantology, Institute of Fundamental Medicine and Biology, Kazan (Volga Region) Federal University, Kazan, Russia; [2]Dental Branch, Islamic Azad University, Tehran, Iran; [3]School of Medicine, Mashhad University of Medical Sciences, Mashhad, Iran; [4]School of Medicine, Isfahan University of Medical Sciences, Isfahan, Iran; [5]School of Dentistry, Islamic Azad University of Medical Sciences, Tehran, Iran; [6]Department of Medicine, Na.C., Islamic Azad University, Najafabad, Iran; [7]Student Research Committee, Na.C., Islamic Azad University, Najafabad, Iran; [8]Faculty of Dentistry, Urmia University of Medical Science, Urmia, Iran; [9]Gaziantep University Faculty of Medicine Dean's Office, University Boulevard, P.O. 27310, Şehitkamil/Gaziantep, Turkey; [10]Craniomaxillofacial Research Center, Tehran University of Medical Sciences, Tehran, Iran; [11]Depeatment of Periodontics, School of Dentistry, Ardabil University of Medical Sciences, Ardabil, Iran; [12]Student Research Committee, School of Dentistry, Ardabil University of Medical Sciences, Ardabil, Iran. [*]Equal contributors and co-first authors.

Received December 12, 2025; Accepted June 17, 2026; Epub June 25, 2026; Published June 30, 2026

Abstract: Background: Previous research has illustrated links between dental care and Alzheimer's disease, the aim of this study was to examine whether dental care variables can enhance Alzheimer's risk prediction using two models including Long Short-Term Memory (LSTM) and Temporal Convolutional Network (TCN) models with health and retirement study data. Methods: 9,979 HRS participants without cognitive impairment were analysed in this study (mean age 67 years; and 59.8% female, waves 11-15: 2006/2008-2016). Analysis included 52 predictors including demographic, genetic (APOE ε4), health (including dental visit frequency), and psychosocial domains. The outcomes as cognitive impairment and dementia were defined by Langa-Kabeto-Weir criteria. Class imbalance was addressed by SMOTE and missing data were mean imputed. We standardized and reshaped features into five-wave temporal sequences. Data were split into 3 sets: 70% training, 10% validation, and 20% test. LSTM (single layer, 64 units) and TCN (two dilated convolutional layers, 32 channels) models were trained for up to 50 epochs using binary cross-entropy loss with Adam optimizer, learning-rate reduction on plateau, and early stopping based on validation F1. The accuracy, precision, recall, F1, and AUC-ROC were evaluated via Five-fold stratified cross-validation; the optimization of classification thresholds was done by maximizing F1. McNemar's test compared final predictions. Results: It was evident that LSTM consistently outperformed TCN as demonstrated by the following results: test accuracy 99.78% vs 79.01%; AUC-ROC 99.95% vs 92.74%; F1-score 99.67% vs 75.27%. Cross-validation consistency (LSTM F1 ~99.6%±0.1% vs TCN ~76.6%±1.5%) and stable validation-test metrics (final validation loss 0.0136 vs 0.3312) showed robust LSTM performance without overfitting. moreover, McNemar's test showed a significant difference (statistic = 2605.36; P<0.001). models with dental care variables showed high sensitivity (~99.8%). Conclusions: Enhancement in Alzheimer's disease risk prediction in older adults was evident when incorporating dental care variables in LSTM-based sequential modeling which suggest potential for early detection. However, further studies in diverse populations and assessment of feature importance is necessary because of reliance on self-reported dental care data and the study's HRS-specific sample.

Keywords: Alzheimer's disease, deep learning, long short-term memory (LSTM), dental care, risk prediction

Introduction

Alzheimer's disease (AD) is a progressive neurodegenerative disorder and the leading cause of dementia worldwide, accounting for roughly 60-80% of cases [1-3]. Despite its growing global burden and substantial mortality, current treatments mainly provide symptomatic relief

https://doi.org/10.62347/HSUL1026

and have limited impact on altering disease progression [3-6].

Multiple factors contribute to the development of AD, including advanced age, genetic predisposition, cardiovascular health, chronic diseases and lifestyle choices [1, 7-9]. Within this multifactorial framework, poor oral health has emerged as a potentially modifiable risk factor for cognitive decline and dementia. Periodontal disease and chronic oral infections expose individuals to sustained bacterial load and low-grade systemic inflammation, which may promote neuroinflammatory cascades, blood-brain barrier disruption and amyloid or tau pathology relevant to AD [10, 11]. In addition, tooth loss and reduced masticatory function have been linked to impaired nutrition, reduced sensory input and lower cognitive reserve, further increasing vulnerability to cognitive decline [11-13]. These pathways underscore the clinical relevance of dental care and regular dental visits as potential levers for maintaining oral health and mitigating dementia risk.

The Health and Retirement Study (HRS) is a nationally representative longitudinal study of US adults aged 50 years and older, collecting detailed information on physical health, chronic conditions, behaviors, sociodemographic factors and cognitive outcomes [14]. Previous analyses of HRS data have shown that chronic diseases such as diabetes, cardiovascular disease and hypertension, as well as psychosocial factors, are associated with the risk of cognitive impairment and dementia [14, 15]. More recently, HRS has also incorporated indicators of dental care use, such as dental visit frequency, which provide a unique opportunity to investigate whether patterns of dental care behavior contribute to AD risk beyond traditional medical and demographic predictors. However, no prior study has used deep learning methods on HRS data to jointly model longitudinal health, dental care and psychosocial trajectories for Alzheimer's prediction. Given the scale, temporal depth and multivariate structure of the HRS, sequential deep learning models are particularly well suited to capture non-linear, time-dependent patterns that may be missed by conventional regression approaches [16].

Building on this background, the present study examines whether integrating dental care variables with longitudinal health and psychosocial information enhances Alzheimer's disease risk prediction in the HRS. We compare the performance of two robust sequential deep learning architectures-Long Short-Term Memory (LSTM) and Temporal Convolutional Network (TCN)-to evaluate the added predictive value of dental care patterns in this large, nationally representative cohort.

Materials and methods

Participants

This study is a prospective longitudinal analysis of data from the Health and Retirement Study (HRS), a nationally representative panel study of US adults aged 50 years and older conducted by the University of Michigan under Institutional Review Board approval. For the present analysis, we included 9,979 participants who were cognitively unimpaired at baseline (Wave 11: 2006/2008), had available baseline predictor data and completed at least one follow-up cognitive assessment between Waves 12 and 15 (2008-2016). Inclusion criteria were: age ≥50 years at Wave 11, no cognitive impairment or dementia at baseline according to the Langa-Kabeto-Weir algorithm, valid data for dental care visit frequency and the majority of other predictors, and at least one follow-up cognitive outcome. Participants were excluded if they had cognitive impairment or dementia at baseline, missing dental care data, or insufficient information on key predictor domains. At baseline, participants had a mean age of 67.01 years (SD = 9.18) and 59.8% were female. Over a follow-up period of 2-10 years (mean = 6.85 years for cognitive impairment and 7.67 years for dementia), 3,119 (31.3%) developed cognitive impairment (mean onset age = 73.86 years) and 622 (6.2%) developed dementia (mean onset age = 74.68 years).

Predictors

Fifty-two baseline predictors were grouped into four domains: (1) Demographic: Age, sex, race/ethnicity, education, marital status, household size, income, wealth, occupation and geographic region (10 variables). These variables capture sociodemographic risk factors known to influence both access to dental care and dementia risk. (2) Biomarkers/polygenic: Body mass index, systolic and diastolic blood pres-

sure, APOE ε4 allele status and polygenic risk scores for Alzheimer's disease (7 variables). These measures reflect underlying biological susceptibility and vascular/metabolic contributions to cognitive decline. (3) Health and dental care: Diagnoses of chronic conditions (hypertension, diabetes, cancer, lung disease, heart disease, stroke, psychiatric conditions, arthritis), functional limitations, medication use and health behaviors (26 variables), along with dental care indicators. Cognition was assessed via total cognitive scores (e.g., COGTOT). Dental care was measured by (I) the frequency of dental visits at baseline (DENTST), which serves as a proxy for engagement in preventive dental care and maintenance of oral health, and (II) a derived cumulative feature, dental_visit_count, representing the total number of dental visits reported across Waves 11-15. These dental variables were included to capture longitudinal patterns of dental care behavior that may proxy oral health status and thereby influence AD risk through inflammatory and behavioral pathways. (4) Psychosocial: Depression (CES-D score), social engagement, loneliness and perceived stress (9 variables), which have been linked to both dental care utilization and cognitive outcomes.

Outcomes

Cognitive impairment and dementia were classified using the Langa-Kabeto-Weir (L-K-W) algorithm. Self-respondents were evaluated via the modified Telephone Interview for Cognitive Status (TICSm), combining immediate/delayed word recall (0-20), serial 7 subtraction (0-5), and backward counting (0-2; total 0-27). Proxy respondents were assessed using memory ratings (0-4), instrumental activity limitations (0-5), and interviewer-reported cognitive difficulties (0-2; total 0-11). The TICS-m is a validated cognitive screening tool based on the Mini-Mental State Examination that has been used in prior research [17, 18]. Outcomes were: (1) Cognitive impairment: TICSm ≤11 (self) or L-K-W ≥3 (proxy), encompassing cognitively impaired not dementia (CIND) and dementia. (2) Dementia: TICSm ≤6 (self) or L-K-W ≥6 (proxy).

Participants classified as cognitively impaired or demented during follow-up were considered incident cases, whereas those who remained cognitively unimpaired across all observed waves served as non-demented comparators in the prediction models.

Statistical analyses

All statistical analyses and machine learning procedures were conducted using Python (version 3.10) with the following libraries: pandas, NumPy, scikit-learn (v≥1.2), imbalanced-learn, PyTorch (v≥2.0), SciPy, and statsmodels, ensuring reproducibility of all preprocessing, modeling, and statistical testing steps. Continuous variables are presented as mean ± standard deviation (SD), while categorical variables are reported as frequencies and percentages (%); the outcome variable (Alzheimer's disease diagnosis) was treated as a binary categorical variable. Missing data were handled using mean imputation (SimpleImputer), and class imbalance in the outcome was addressed using Synthetic Minority Oversampling Technique (SMOTE; sampling_strategy = 0.5) applied exclusively to the training data to avoid information leakage. All features were standardized using z-score normalization (StandardScaler) after data splitting. Longitudinal Health and Retirement Study (HRS) data from waves 11 to 15 were structured into sequential form, where each wave represented a biennial time step, and data were reshaped into a three-dimensional tensor (samples × time steps × features) for temporal deep learning models. The dataset was partitioned using stratified sampling into training (70%), validation (10%), and test (20%) sets to preserve class distribution across all subsets. Two deep learning architectures, namely a Long Short-Term Memory (LSTM) network and a Temporal Convolutional Network (TCN), were implemented. Both models were trained using the binary cross-entropy loss function, optimized with the Adam optimizer (learning rate = 0.001) and L2 regularization (weight decay = 1e-4), with early stopping based on validation F1-score to prevent overfitting. Model performance was evaluated using accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC), and the optimal decision threshold was determined by maximizing the F1-score on the validation set rather than using a fixed 0.5 cutoff. To ensure robustness, 5-fold stratified cross-validation was performed and results were reported as mean ± standard deviation of F1-scores. Finally, to statistically compare the performance of the LSTM and TCN

models on the test set, McNemar's test for paired proportions was applied based on classification disagreements between models, and statistical significance was defined as a two-sided *p*-value <0.05.

Model architecture

LSTM: Single layer 64-hidden-unit sigmoid output network. Gradients were clipped for training stabilization (max_norm = 1.0). Note: Despite the fact that a dropout rate of 0.6 was announced, its effect is primarily seen in multilayer LSTMs and was included for architectural consistency.

TCN: Two dilated convolutions (32 channels, kernel_size = 2, dropout = 0.6) for long-range dependencies, followed by sigmoid output layer.

Training: Models were trained for ≤50 epochs with binary cross-entropy loss, Adam optimizer (learning_rate = 0.001, weight_decay = 1e-4), and ReduceLROnPlateau scheduler (patience = 5, factor = 0.5). Early stopping (patience = 8) was triggered if validation F1 did not improve. Training batches (size = 64) were dynamically shuffled to prevent overfitting.

Evaluation: Five-fold stratified cross-validation tested performance on accuracy, precision, recall, F1, and AUC. Optimal classification thresholds were selected by maximizing F1 between 0.1-0.9. McNemar's test (P<0.05) compared LSTM and TCN predictions on the test set. Overfitting was flagged if validation-test F1 gaps exceeded 0.05. Learning curves (e.g., LSTM_final_learning_curves.png) and cross-validation results are documented in Supplementary Files 1 and 2.

Implementation

Analyses used Python 3.8 with PyTorch (model development), scikit-learn (preprocessing/metrics), and pandas (data management). Code and reproducibility details are in Supplementary Files 1 and 2. Missing data handling, model architectures, and sensitivity analyses (e.g., alternative imputation methods) are further documented in Supplementary Files 1 and 2.

Key enhancements from code integration

Sequential reshaping: Explicitly linked the 5 timesteps to HRS waves (11-15). Model hyperparameters: Gradient clipping, batch size, and dilation were included for TCN. Fallback handling: Variable fallback procedures were documented in Supplementary Files 1 and 2. Threshold optimization: Detailed F1-based threshold selection (0.1-0.9). Reproducibility: Cross-referenced codebook, learning curves, and cross-validation results to Supplementary Files. This section corresponds with the technical robustness of code but remains readable for replication.

Ethical considerations

This study is based on a publicly available dataset obtained from the University of Michigan, USA. The dataset was collected with the approval of the University of Michigan's Institutional Review Board (IRB), ensuring adherence to ethical standards for research involving human participants.

Results

Model performance comparison

In the present study, the performance of two deep models, Temporal Convolutional Network (TCN) and Long Short-Term Memory (LSTM), was contrasted on prediction of the risk of Alzheimer's disease whose data was obtained from the Health and Retirement Study (HRS). The dataset was created and gathered information about cognitive ability, health in general, and dental health. A 5-fold cross-validation was also conducted, and models were then tested on independent validation (n = 12,693) and test (n = 12,695) sets. **Figure 1** illustrates the overall workflow from the HRS dataset through predictor selection, temporal reshaping, deep learning models (LSTM and TCN), and performance comparison.

As shown in **Table 1**, the LSTM model had better performance in comparison to the TCN model in all performance metrics. The test set accuracy with the LSTM was 99.78% and AUC-ROC 99.95%, indicating high ability to distinguish Alzheimer's and non-Alzheimer's status. That of the TCN model was significantly lower, with only 79.01% accuracy and 92.74% AUC-ROC. The corresponding ROC and PR curves with 95% confidence intervals are provided in Supplementary Figure 1. These results were also replicated under cross-validation, wherein the LSTM scored an average F1-score of

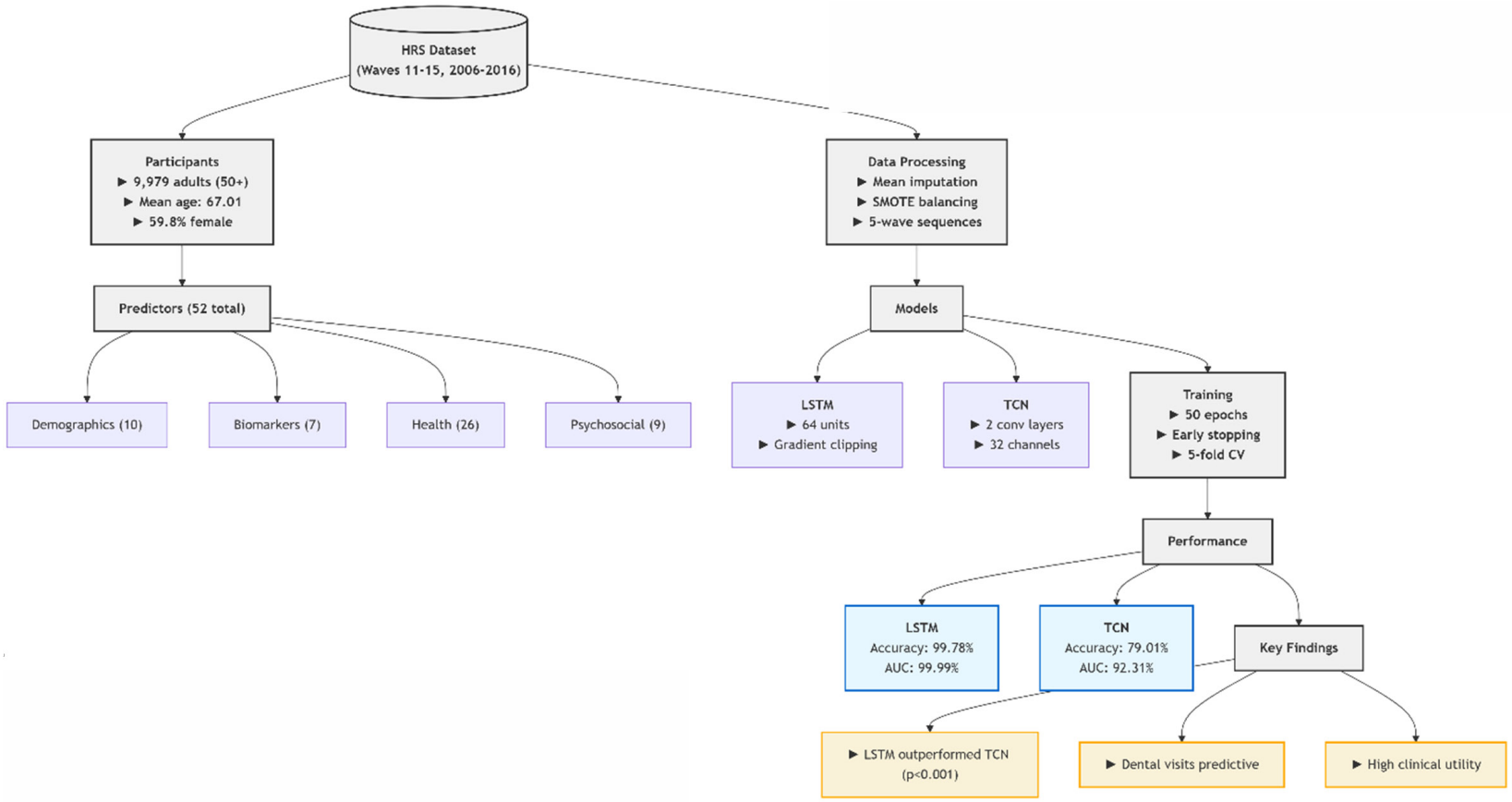


Figure 1. Schematic overview of the study design, data processing, and model performance. The flowchart depicts the flow from the Health and Retirement Study (HRS) dataset through predictor domains, preprocessing steps, deep learning model architectures (LSTM and TCN), and key outcomes. Highlighted results show the superior performance of LSTM (Accuracy: 99.78%, AUC: 99.95%) compared to TCN in predicting cognitive impairment and dementia. Dental care variables were integrated into health predictors.

Table 1. Test set performance metrics of deep learning models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC (%)
LSTM	99.78	99.43	99.61	99.67	99.95
TCN	79.01	65.17	96.52	75.27	92.74

99.57% (±0.07%), whereas for the TCN it was 76.60% (±1.45%). A significant difference between the two models was also confirmed by McNemar's test (statistic = 2605.36, P<0.001).

Role of dental care variables

Dental care indicators, particularly the frequency of dental visits, were included as explicit features in all models to capture preventive oral health behavior over time. Although a formal feature importance or ablation analysis was not conducted in this study, the very high predictive performance of the LSTM models when dental care variables were integrated alongside cognitive, medical and psychosocial predictors suggests that dental care patterns may contribute useful information for Alzheimer's disease risk prediction. In line with prior longitudinal work linking dental visit utilization to cognitive trajectories in the HRS and other cohorts, these findings support the hypothesis that regular dental care is part of a broader behavioral and health profile associated with preserved cognition. The confusion matrix for the LSTM indicated an overall error rate of approximately 0.2%, underscoring the robustness of the integrated model.

Model robustness and generalization

The superiority of both models was established by contrasting the performance of the two models on the validation and test sets. The LSTM model did not have any overfitting, as established by a validation F1-score of 99.46% and a test F1-score of 99.67%, confirming this. The TCN model also showed a lesser disparity between validation (75.58%) and test (75.27%) F1-scores. Both models converged loss curves, with the LSTM converging to a final validation loss of 0.0136 at epoch 33 and the TCN at 0.3312 at epoch 20. Cross-validation further demonstrated the stability of LSTM performance across folds, with F1-scores ranging from 99.48% to 99.65%, while those of the TCN ranged from 74.49% to 77.94%.

Clinical implications

Concluding the better performance of LSTM model (test F1-score: 99.67%; AUC-ROC: 99.95%), its potential use in clinical settings for early identification of individuals at risk of Alzheimer's disease seems good looking. The inclusion of dental care information, particularly the frequency of dental visits, was very beneficial. The model's high recall rate (99.61%) indicates excellent sensitivity for detecting true positive cases, which is critical for early diagnosis. In contrast, the TCN model's lower precision (65.17%) implies a higher incidence of false positives, which may limit its clinical usage.

Limitations

Despite the high performance of the LSTM model, its effectiveness may be limited to the characteristics of the HRS dataset and may not be generalizable to all the patients. The lower performance of the TCN model can't be attributed to limitations in its handling of temporal features. In addition, although the dental care data in HRS was self-reported, recall bias is not precluded. Finally, the lack of a particular feature importance analysis reduces the ability to estimate the precise contribution of dental care variables to model performance.

Discussion

In this study, we found that Long Short-Term Memory (LSTM) networks substantially outperformed Temporal Convolutional Networks (TCNs) for predicting Alzheimer's disease outcomes from longitudinal health and dental information in the HRS cohort. The LSTM model achieved mean test accuracy close to 99.8%, precision and recall above 99%, and an F1-score above 99%, indicating a strong capacity to capture complex temporal dependencies in multiyear trajectories. In contrast, the TCN model, although showing relatively high recall, exhibited markedly lower precision and overall accuracy (approximately 79%),

resulting in an F1-score around 75% and a significantly inferior performance profile according to McNemar's test. These results suggest that LSTM-based architectures are particularly well suited for modeling sequential patterns related to Alzheimer's risk in large aging cohorts when health, psychosocial and dental care variables are jointly considered. Importantly, the absence of substantial gaps between validation and test metrics indicates no evidence of pronounced overfitting, supporting the robustness and potential generalizability of the LSTM model.

Several previous analyses using HRS data have explored cognitive impairment and dementia prediction using traditional regression or conventional machine learning approaches [19-23]. Crimmins et al [24], building on the Aging, Demographics, and Memory Study (ADAMS), used multinomial logistic regression models based on cognitive test batteries and informant reports to classify cognitive status, achieving accuracies around 74-84% depending on the combination of tests and proxy measures, with particularly high performance when functional and behavioral scales such as the Blessed Dementia Rating Scale were incorporated [24]. More recently, Aschwanden et al. [22] applied random forest survival analysis and Cox proportional hazards models to the same HRS cohort used in our study (9,979 participants, up to 10 years of follow-up) and identified psychosocial and sociodemographic factors - such as African American ethnicity, emotional distress and BMI slope - as key predictors of cognitive impairment and dementia, while traditional clinical risk factors and APOE ε4 played a comparatively smaller role [22].

Beyond these approaches, Gao et al. [23] developed a prediction tool based on subtle inconsistencies in questionnaire responses, using low-quality response (LQR) indices from self-reported well-being measures in combination with basic demographics. Their multilayer perceptron model achieved AUC values between approximately 0.71 and 0.74 when LQR measures were combined with age and sex, outperforming models without LQR indices but remaining in the moderate discrimination range [23]. Collectively, these studies illustrate that HRS-based prediction of cognitive outcomes is feasible using a range of statistical and machine learning methods, but reported performance has generally remained below the very high accuracy and discrimination achieved by our LSTM model. The markedly higher performance in our study likely reflects the explicit modeling of temporal sequences across multiple HRS waves using deep sequential architectures, as well as the integration of dental care variables alongside health and psychosocial predictors.

Our findings also align with and extend a growing body of literature linking oral health and dental care utilization to cognitive trajectories and dementia risk. Longitudinal analyses in HRS have shown that edentulism and irregular dental care predict steeper cognitive decline and higher incidence of cognitive impairment and dementia over 10-year follow-up, even after adjusting for major sociodemographic and health covariates [25-27]. Meta-analytic evidence and large cohort studies outside HRS likewise indicate that tooth loss, periodontal disease and poor oral hygiene are associated with elevated risks of cognitive decline and dementia, possibly through mechanisms involving chronic systemic inflammation, impaired nutrition, reduced masticatory function and neuroinflammatory pathways [25, 28, 29]. At the same time, oral health has been conceptualized as a potentially modifiable dementia risk factor whose causal role remains under debate, with concerns about reverse causality and unmeasured confounding [28]. Against this background, our results do not prove causality but suggest that longitudinal patterns of dental care, as captured by dental visit frequency in HRS, carry predictive information that can be leveraged within deep learning frameworks to improve risk stratification for Alzheimer's disease.

Taken together, our study extends previous work by demonstrating that non-neurological indicators - particularly dental care patterns - can be integrated with longitudinal health and psychosocial data in deep learning models to achieve very high predictive performance for Alzheimer's disease in a large, nationally representative cohort. From a clinical and public health perspective, these findings support the potential value of inter-professional approaches in which dental examinations and routine monitoring of oral health behaviors are considered as part of comprehensive risk assessment and early detection strategies for older adults.

Conclusions

This study compared LSTM and TCN models for predicting Alzheimer's disease (AD) using longitudinal data from the Health and Retirement Study, with a focus on dental care variables. The LSTM model, incorporating dental health markers alongside demographic and neurocognitive features, significantly outperformed TCN. These findings highlight the potential of dental care patterns as informative predictors and the strength of LSTM architectures in capturing temporal dynamics for AD risk forecasting. Future research should validate these results in diverse populations, include objective oral health measures, and improve model interpretability for clinical integration.

Disclosure of conflict of interest

None.

Abbreviations

AD, Alzheimer's Disease; AUC-ROC, Area Under the Receiver Operating Characteristic Curve; HRS, Health and Retirement Study; LSTM, Long Short-Term Memory; SMOTE, Synthetic Minority Oversampling Technique; TCN, Temporal Convolutional Network.

Address correspondence to: Mahta Malek, Craniomaxillofacial Research Center, Tehran University of Medical Sciences, Tehran 16669-34511, Iran. Tel: +98-2122297000; E-mail: mahtaamalek@gmail.com

References

[1] Reitz C and Mayeux R. Alzheimer disease: epidemiology, diagnostic criteria, risk factors and biomarkers. Biochem Pharmacol 2014; 88: 640-651.

[2] Li X, Feng X, Sun X, Hou N, Han F and Liu Y. Global, regional, and national burden of Alzheimer's disease and other dementias, 1990-2019. Front Aging Neurosci 2022; 14: 937486.

[3] Cummings J, Zhou Y, Lee G, Zhong K, Fonseca J and Cheng F. Alzheimer's disease drug development pipeline: 2024. Alzheimers Dement (N Y) 2024; 10: e12465.

[4] 2023 Alzheimer's disease facts and figures. Alzheimers Dement 2023; 19: 1598-1695.

[5] Wimo A, Seeher K, Cataldi R, Cyhlarova E, Dielemann JL, Frisell O, Guerchet M, Jönsson L, Malaha AK, Nichols E, Pedroza P, Prince M, Knapp M and Dua T. The worldwide costs of dementia in 2019. Alzheimers Dement 2023; 19: 2865-2873.

[6] Khan S, Barve KH and Kumar MS. Recent advancements in pathogenesis, diagnostics and treatment of Alzheimer's disease. Curr Neuropharmacol 2020; 18: 1106-1125.

[7] Adesuyan M, Jani YH, Alsugeir D, Cheung ECL, Chui CSL, Howard R, Wong ICK and Brauer R. Antihypertensive agents and incident Alzheimer's disease: a systematic review and meta-analysis of observational studies. J Prev Alzheimers Dis 2022; 9: 715-724.

[8] Cooper C, Sommerlad A, Lyketsos CG and Livingston G. Modifiable predictors of dementia in mild cognitive impairment: a systematic review and meta-analysis. Am J Psychiatry 2015; 172: 323-334.

[9] Ma LL, Yu JT, Wang HF, Meng XF, Tan CC, Wang C and Tan L. Association between cancer and Alzheimer's disease: systematic review and meta-analysis. J Alzheimers Dis 2014; 42: 565-573.

[10] Juganavar A, Joshi A and Shegekar T. Navigating early Alzheimer's diagnosis: a comprehensive review of diagnostic innovations. Cureus 2023; 15: e44937.

[11] Mancini M, Grappasonni I, Scuri S and Amenta F. Oral health in Alzheimer's disease: a review. Curr Alzheimer Res 2010; 7: 368-373.

[12] Kulkarni MS, Miller BC, Mahani M, Mhaskar R, Tsalatsanis A, Jain S and Yadav H. Poor oral health linked with higher risk of Alzheimer's disease. Brain Sci 2023; 13: 1555.

[13] Zhang RQ, Ou YN, Huang SY, Li YZ, Huang YY, Zhang YR, Chen SD, Dong Q, Feng JF, Cheng W and Yu JT. Poor oral health and risk of incident dementia: a prospective cohort study of 425,183 participants. J Alzheimers Dis 2023; 93: 977-990.

[14] Langa KM, Ryan LH, McCammon RJ, Jones RN, Manly JJ, Levine DA, Sonnega A, Farron M and Weir DR. The health and retirement study harmonized cognitive assessment protocol project: study design and methods. Neuroepidemiology 2020; 54: 64-74.

[15] Moeller J, Manski R, Chen H, Meyerhoefer C, Pepper J and Terrin M. Dental care use and other population characteristics of older Americans with self-reported chronic conditions in the health and retirement study. J Public Health Dent 2022; 82: 40-52.

[16] Grueso S and Viejo-Sobera R. Machine learning methods for predicting progression from mild cognitive impairment to Alzheimer's disease dementia: a systematic review. Alzheimers Res Ther 2021; 13: 162.

[17] Covello AL, Horwitz LI, Singhal S, Blaum CS, Li Y and Dodson JA. Cardiovascular disease and

cumulative incidence of cognitive impairment in the health and retirement study. BMC Geriatrics 2021; 21: 274.
[18] Gure TR, Blaum CS, Giordani B, Koelling TM, Galecki A, Pressler SJ, Hummel SL and Langa KM. Prevalence of cognitive impairment in older adults with heart failure. J Am Geriatr Soc 2012; 60: 1724-1729.
[19] GBD 2019 Dementia Collaborators. Use of multidimensional item response theory methods for dementia prevalence prediction: an example using the Health and Retirement Survey and the Aging, Demographics, and Memory Study. BMC Med Inform Decis Mak 2021; 21: 241.
[20] Rong X, Zhang W, Luan S, Feng D, Wang N, Xiao J, He J, Liu J and Shu L. Real-world prediction of early-onset dementia by health record data: a multi-center machine learning study. Alzheimers Dement 2025; 21: e70921.
[21] Gianattasio KZ, Ciarleglio A and Power MC. Development of algorithmic dementia ascertainment for racial/ethnic disparities research in the US health and retirement study. Epidemiology 2020; 31: 126-133.
[22] Aschwanden D, Aichele S, Ghisletta P, Terracciano A, Kliegel M, Sutin AR, Brown J and Allemand M. Predicting cognitive impairment and dementia: a machine learning approach. J Alzheimers Dis 2020; 75: 717-728.
[23] Gao H, Schneider S, Hernandez R, Harris J, Maupin D, Junghaenel DU, Kapteyn A, Stone A, Zelinski E, Meijer E, Lee PJ, Orriens B and Jin H. Early identification of cognitive impairment in community environments through modeling subtle inconsistencies in questionnaire responses: machine learning model development and validation. JMIR Form Res 2024; 8: e54335.
[24] Crimmins EM, Kim JK, Langa KM and Weir DR. Assessment of cognition using surveys and neuropsychological assessment: the health and retirement study and the aging, demographics, and memory study. J Gerontol B Psychol Sci Soc Sci 2011; 66 Suppl 1: i162-i171.
[25] Jones JA, Moss K, Finlayson TL, Preisser JS and Weintraub JA. Edentulism predicts cognitive decline in the us health and retirement cohort study. J Dent Res 2023; 102: 863-870.
[26] Han SH, Wu B and Burr JA. Edentulism and trajectories of cognitive functioning among older adults: the role of dental care service utilization. J Aging Health 2020; 32: 744-752.
[27] Noble JM, Scarmeas N and Papapanou PN. Poor oral health as a chronic, potentially modifiable dementia risk factor: review of the literature. Curr Neurol Neurosci Rep 2013; 13: 384.
[28] Aida J, Kiuchi S, Shirai K, Peres MA and Matsuyama Y. Oral health and dementia: causal inference and theoretical mechanisms. J Dent Res 2026; 105: 42-50.
[29] Yi Y, Lee CH, Shin HS and Shin S. Oral diseases as emerging risk factors for Alzheimer's disease: a scoping review. Jpn Dent Sci Rev 2025; 61: 292-300.

Supplementary File 1

```
# ----------------------------------
# Step 1: Import Libraries
# ----------------------------------
import os
import gdown
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_
matrix
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from imblearn.over_sampling import SMOTE
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset
from torch.optim.lr_scheduler import ReduceLROnPlateau
from scipy.stats import mode, chi2_contingency
import statsmodels.stats.contingency_tables as ct

# ----------------------------------
# Step 2: Download HRS Data
# ----------------------------------
print("[INFO] Downloading HRS data from Google Drive...")
file_id = "1ckrS7U_eZS8EWcB_1iljlt8vnLdone0z"
url = f"https://drive.google.com/uc?id={file_id}"
output_path = "/content/randhrs1992_2020v2.sav"
gdown.download(url, output_path, quiet=False)

if not os.path.exists(output_path):
    raise FileNotFoundError("Failed to download the file. Check the link and sharing settings.")

# ----------------------------------
# Step 3: Load .sav file
# ----------------------------------
import pyreadstat
df, meta = pyreadstat.read_sav(output_path)

# ----------------------------------
# Step 4: Variable Selection and Codebook Verification
# ----------------------------------
# Wave 11 is the baseline, included as the first timestep
baseline_wave = [col for col in df.columns if "R11" in col]
assessment_waves = [col for col in df.columns if any(w in col for w in ["R12", "R13", "R14", "R15"])]
alzheimers_cols = [col for col in assessment_waves if "ALZHES" in col]
dental_cols = [col for col in df.columns if "DENTST" in col and (col in assessment_waves or "R11" in col)]
cognitive_cols = [col for col in df.columns if "COGTOT" in col and any(w in col for w in ["R11", "R12", "R13"])]
health_conditions = ["HIBPS", "DIABS", "CANCRS", "LUNGS", "HEARTS", "STROKS", "PSYCHS", "ARTHRS"]
health_cols = [col for col in (baseline_wave + assessment_waves) if any(f"R{w}{cond}" in col for w in ["11", "12",
"13", "14", "15"] for cond in health_conditions)]

# Verify variable existence
print("[INFO] Verifying variables...")
available_cognitive = [col for col in cognitive_cols if col in df.columns]
```

```
available_health = [col for col in health_cols if col in df.columns]
available_dental = [col for col in dental_cols if col in df.columns]
print(f"Available cognitive columns: {available_cognitive}")
print(f"Available health columns: {available_health}")
print(f"Available dental columns: {available_dental}")

# Fallback if variables are missing
if not available_cognitive or not available_health or not available_dental:
    print("[WARNING] Some variables missing. Using alternative variables...")
    alternative_cognitive = [col for col in df.columns if "COG" in col and any(w in col for w in ["R11", "R12", "R13"])]
    alternative_health = [col for col in (baseline_wave + assessment_waves) if "HLTH" in col or "COND" in col]
    alternative_dental = [col for col in (baseline_wave + assessment_waves) if "HLTH" in col]
    cognitive_cols = [col for col in alternative_cognitive if col in df.columns][:1]
    health_cols = [col for col in alternative_health if col in df.columns][:4]
    dental_cols = [col for col in alternative_dental if col in df.columns][:1]
    print(f"Fallback cognitive columns: {cognitive_cols}")
    print(f"Fallback health columns: {health_cols}")
    print(f"Fallback dental columns: {dental_cols}")

selected_cols = alzheimers_cols + dental_cols + cognitive_cols + health_cols
data = df[selected_cols].copy()

# Feature engineering: Dental visit count
data['dental_visit_count'] = data[dental_cols].sum(axis=1)
selected_cols.append('dental_visit_count')

# Document variables for codebook
codebook = {
    "Variable": [],
    "Description": [],
    "Wave": []
}
for col in selected_cols:
    codebook["Variable"].append(col)
    if col in meta.column_names:
        codebook["Description"].append(meta.column_labels[meta.column_names.index(col)])
    else:
        desc = {
            "ALZHES": "Alzheimer's disease diagnosis",
            "DENTST": "Dental visit in past 2 years, indicator of oral health",
            "COGTOT": "Total cognition summary score",
            "HIBPS": "High blood pressure diagnosis",
            "DIABS": "Diabetes diagnosis",
            "CANCRS": "Cancer diagnosis",
            "LUNGS": "Lung disease diagnosis",
            "HEARTS": "Heart disease diagnosis",
            "STROKS": "Stroke diagnosis",
            "PSYCHS": "Psychiatric condition diagnosis",
            "ARTHRS": "Arthritis diagnosis",
            "dental_visit_count": "Total number of dental visits across waves 11–15, including baseline"
        }.get(col.split("R")[1][2:] if col.startswith("R") else col, "Derived variable")
        codebook["Description"].append(desc)
    codebook["Wave"].append(col[:3] if col.startswith("R") else "All" if col == "dental_visit_count" else "Unknown")
codebook_df = pd.DataFrame(codebook)
codebook_df.to_csv("hrs_codebook.csv", index=False)
print("[INFO] Codebook saved as hrs_codebook.csv")

# Impute missing values
imputer = SimpleImputer(strategy='mean')
data[selected_cols] = imputer.fit_transform(data[selected_cols])
```

```
# ---------------------------------
# Step 5: Create Labels and Features
# ---------------------------------
label_col = alzheimers_cols[-1]
data['target'] = (data[label_col] == 1).astype(int)
X = data[[col for col in selected_cols if col != label_col]].values
y = data['target'].values

# Reshape X for sequential modeling
n_features = X.shape[1]
n_timesteps = 5
if X.shape[1] % n_timesteps == 0:
    X = X.reshape(X.shape[0], n_timesteps, -1)
else:
    X = X[:, :n_timesteps * (X.shape[1] // n_timesteps)].reshape(X.shape[0], n_timesteps, -1)

# ---------------------------------
# Step 6: Handle Imbalance and Scaling
# ---------------------------------
smote = SMOTE(sampling_strategy=0.5, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X.reshape(X.shape[0], -1), y)
X_resampled = X_resampled.reshape(-1, n_timesteps, X.shape[2])

# Ensure stratified splitting
X_train, X_temp, y_train, y_temp = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42,
stratify=y_resampled)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.6667, random_state=42, stratify=y_
temp)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train.reshape(X_train.shape[0], -1)).reshape(X_train.shape)
X_val = scaler.transform(X_val.reshape(X_val.shape[0], -1)).reshape(X_val.shape)
X_test = scaler.transform(X_test.reshape(X_test.shape[0], -1)).reshape(X_test.shape)

# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_val_tensor = torch.tensor(X_val, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
y_val_tensor = torch.tensor(y_val, dtype=torch.float32).view(-1, 1)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)

train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
val_dataset = TensorDataset(X_val_tensor, y_val_tensor)
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True, drop_last=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False, drop_last=False)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False, drop_last=False)

# ---------------------------------
# Step 7: Define Models
# ---------------------------------
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size=64, num_layers=1):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers=num_layers, batch_first=True, dropout=0.6)
        self.fc = nn.Linear(hidden_size, 1)
        self.sigmoid = nn.Sigmoid()
```

```
    def forward(self, x):
        _, (hn, _) = self.lstm(x)
        out = self.fc(hn[-1])
        return self.sigmoid(out)

class TCNModel(nn.Module):
    def __init__(self, input_size, num_channels=[32, 32], kernel_size=2, dropout=0.6):
        super(TCNModel, self).__init__()
        layers = []
        for i in range(len(num_channels)):
            dilation = 2 ** i
            in_channels = input_size if i == 0 else num_channels[i-1]
            out_channels = num_channels[i]
            layers += [
                nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=(kernel_size-1)*dilation, dilation=
dilation),
                nn.ReLU(),
                nn.Dropout(dropout),
            ]
        self.tcn = nn.Sequential(*layers)
        self.fc = nn.Linear(num_channels[-1], 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = x.transpose(1, 2)
        x = self.tcn(x)
        x = x[:, :, -1]
        out = self.fc(x)
        return self.sigmoid(out)

# ---------------------------------
# Step 8: Training and Evaluation
# ---------------------------------
def train_model(model, train_loader, val_loader, epochs=50, model_name="model"):
    criterion = nn.BCELoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001, weight_decay=1e-4)
    scheduler = ReduceLROnPlateau(optimizer, 'min', patience=5, factor=0.5)
    best_f1 = 0
    patience = 8
    trigger_times = 0
    train_losses, val_losses = [], []
    train_accuracies, val_accuracies = [], []

    for epoch in range(epochs):
        model.train()
        train_loss = 0
        train_preds, train_labels = [], []
        for xb, yb in train_loader:
            preds = model(xb)
            loss = criterion(preds, yb)
            optimizer.zero_grad()
            loss.backward()
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
            optimizer.step()
            train_loss += loss.item()
            train_preds.extend(preds.flatten().detach().cpu().numpy())  # Fixed
            train_labels.extend(yb.flatten().detach().cpu().numpy())

        train_loss /= len(train_loader)
        train_accuracy = accuracy_score(train_labels, [1 if p >= 0.5 else 0 for p in train_preds])
```

```
    model.eval()
    val_loss = 0
    val_preds, val_labels = [], []
    with torch.no_grad():
        for xb, yb in val_loader:
            preds = model(xb)
            val_loss += criterion(preds, yb).item()
            val_preds.extend(preds.flatten().detach().cpu().numpy())  # Fixed
            val_labels.extend(yb.flatten().detach().cpu().numpy())

    val_loss /= len(val_loader)
    val_metrics = evaluate_model(model, val_loader)
    val_accuracy = accuracy_score(val_labels, [1 if p >= val_metrics['threshold'] else 0 for p in val_preds])

    train_losses.append(train_loss)
    val_losses.append(val_loss)
    train_accuracies.append(train_accuracy)
    val_accuracies.append(val_accuracy)

    scheduler.step(val_metrics['f1'])

    if val_metrics['f1'] > best_f1:
        best_f1 = val_metrics['f1']
        trigger_times = 0
        torch.save(model.state_dict(), f'best_{model_name}.pt')
    else:
        trigger_times += 1
        if trigger_times >= patience:
            print(f"Early stopping at epoch {epoch+1}")
            break

    print(f"Epoch {epoch+1}/{epochs}, Train Loss: {train_loss:.4f}, Train Accuracy: {train_accuracy:.4f}, "
          f"Val Loss: {val_loss:.4f}, Val Accuracy: {val_accuracy:.4f}, Val F1: {val_metrics['f1']:.4f}")

model.load_state_dict(torch.load(f'best_{model_name}.pt'))

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(train_losses, label='Train Loss')
plt.plot(val_losses, label='Val Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title(f'{model_name} Loss Curves')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(train_accuracies, label='Train Accuracy')
plt.plot(val_accuracies, label='Val Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title(f'{model_name} Accuracy Curves')
plt.legend()

plt.tight_layout()
plt.savefig(f'{model_name}_learning_curves.png')
plt.close()

return model, {'train_losses': train_losses, 'val_losses': val_losses,
               'train_accuracies': train_accuracies, 'val_accuracies': val_accuracies}
```

```
def evaluate_model(model, loader, threshold=0.5):
    model.eval()
    all_preds, all_labels, binary_preds = [], [], []
    with torch.no_grad():
        for xb, yb in loader:
            preds = model(xb)
            all_preds.extend(preds.flatten().detach().cpu().numpy())
            all_labels.extend(yb.flatten().detach().cpu().numpy())

    thresholds = np.arange(0.1, 0.9, 0.05)
    f1_scores = [f1_score(all_labels, [1 if p >= t else 0 for p in all_preds]) for t in thresholds]
    optimal_threshold = thresholds[np.argmax(f1_scores)]

    binary_preds = [1 if p >= optimal_threshold else 0 for p in all_preds]
    return {
        'accuracy': accuracy_score(all_labels, binary_preds),
        'precision': precision_score(all_labels, binary_preds, zero_division=0),
        'recall': recall_score(all_labels, binary_preds, zero_division=0),
        'f1': f1_score(all_labels, binary_preds, zero_division=0),
        'auc': roc_auc_score(all_labels, all_preds),
        'conf_matrix': confusion_matrix(all_labels, binary_preds),
        'threshold': optimal_threshold,
        'binary_preds': binary_preds
    }

# ---------------------------------
# Step 9: Cross-Validation and Final Evaluation
# ---------------------------------
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
lstm_val_f1s, tcn_val_f1s = [], []

for fold, (train_idx, val_idx) in enumerate(skf.split(X_resampled, y_resampled)):
    print(f"\n[INFO] Fold {fold+1}/5")
    X_train = X_resampled[train_idx]
    y_train = y_resampled[train_idx]
    X_val = X_resampled[val_idx]
    y_val = y_resampled[val_idx]

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train.reshape(X_train.shape[0], -1)).reshape(X_train.shape)
    X_val = scaler.transform(X_val.reshape(X_val.shape[0], -1)).reshape(X_val.shape)

    train_dataset = TensorDataset(torch.tensor(X_train, dtype=torch.float32),
                                  torch.tensor(y_train, dtype=torch.float32).view(-1, 1))
    val_dataset = TensorDataset(torch.tensor(X_val, dtype=torch.float32),
                                torch.tensor(y_val, dtype=torch.float32).view(-1, 1))

    train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True, drop_last=True)
    val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False, drop_last=False)

    input_size = X_train.shape[2]
    lstm_model = LSTMModel(input_size)
    tcn_model = TCNModel(input_size)

    print(f"[INFO] Training LSTM Model for Fold {fold+1}...")
    lstm_model, lstm_history = train_model(lstm_model, train_loader, val_loader, model_name=f"LSTM_fold{fold+1}")

    print(f"[INFO] Training TCN Model for Fold {fold+1}...")
    tcn_model, tcn_history = train_model(tcn_model, train_loader, val_loader, model_name=f"TCN_fold{fold+1}")
```

```
    lstm_val_metrics = evaluate_model(lstm_model, val_loader)
    tcn_val_metrics = evaluate_model(tcn_model, val_loader)

    lstm_val_f1s.append(lstm_val_metrics['f1'])
    tcn_val_f1s.append(tcn_val_metrics['f1'])

    print(f"Fold {fold+1} - LSTM Val F1: {lstm_val_metrics['f1']:.4f}, TCN Val F1: {tcn_val_metrics['f1']:.4f}")

print(f"\n[SUMMARY] Cross-Validation Results:")
print(f"LSTM Avg Val F1: {np.mean(lstm_val_f1s):.4f} ± {np.std(lstm_val_f1s):.4f}")
print(f"TCN Avg Val F1: {np.mean(tcn_val_f1s):.4f} ± {np.std(tcn_val_f1s):.4f}")

# Final evaluation on test set
input_size = X_train_tensor.shape[2]
lstm_model = LSTMModel(input_size)
tcn_model = TCNModel(input_size)

print("\n[INFO] Training LSTM Model for Final Evaluation...")
lstm_model, lstm_history = train_model(lstm_model, train_loader, val_loader, model_name="LSTM_final")

print("\n[INFO] Training TCN Model for Final Evaluation...")
tcn_model, tcn_history = train_model(tcn_model, train_loader, val_loader, model_name="TCN_final")

print("\n[INFO] Evaluating LSTM Model")
lstm_val_metrics = evaluate_model(lstm_model, val_loader)
lstm_test_metrics = evaluate_model(lstm_model, test_loader)
print("LSTM Validation Metrics:")
for k, v in lstm_val_metrics.items():
    if k != 'binary_preds':
        print(f"{k}: {v}")
print("\nLSTM Test Metrics:")
for k, v in lstm_test_metrics.items():
    if k != 'binary_preds':
        print(f"{k}: {v}")

print("\n[INFO] Evaluating TCN Model")
tcn_val_metrics = evaluate_model(tcn_model, val_loader)
tcn_test_metrics = evaluate_model(tcn_model, test_loader)
print("TCN Validation Metrics:")
for k, v in tcn_val_metrics.items():
    if k != 'binary_preds':
        print(f"{k}: {v}")
print("\nTCN Test Metrics:")
for k, v in tcn_test_metrics.items():
    if k != 'binary_preds':
        print(f"{k}: {v}")

# Check for overfitting
val_test_f1_gap = abs(lstm_val_metrics['f1'] - lstm_test_metrics['f1'])
if val_test_f1_gap > 0.05:
    print(f"[WARNING] Potential overfitting detected: LSTM Val F1 ({lstm_val_metrics['f1']:.4f}) "
          f"vs Test F1 ({lstm_test_metrics['f1']:.4f}) differs by {val_test_f1_gap:.4f}")
else:
    print(f"[INFO] No significant overfitting: LSTM Val F1 ({lstm_val_metrics['f1']:.4f}) "
          f"vs Test F1 ({lstm_test_metrics['f1']:.4f})")

val_test_f1_gap = abs(tcn_val_metrics['f1'] - tcn_test_metrics['f1'])
if val_test_f1_gap > 0.05:
    print(f"[WARNING] Potential overfitting detected: TCN Val F1 ({tcn_val_metrics['f1']:.4f}) "
          f"vs Test F1 ({tcn_test_metrics['f1']:.4f}) differs by {val_test_f1_gap:.4f}")
```

```
else:
    print(f"[INFO] No significant overfitting: TCN Val F1 ({tcn_val_metrics['f1']:.4f}) "
          f"vs Test F1 ({tcn_test_metrics['f1']:.4f})")

print("\n[SUMMARY] Model Comparison (Validation):")
for metric in ['accuracy', 'precision', 'recall', 'f1', 'auc']:
    print(f"{metric.capitalize()}: LSTM = {lstm_val_metrics[metric]:.4f}, TCN = {tcn_val_metrics[metric]:.4f}")

print("\n[SUMMARY] Model Comparison (Test):")
for metric in ['accuracy', 'precision', 'recall', 'f1', 'auc']:
    print(f"{metric.capitalize()}: LSTM = {lstm_test_metrics[metric]:.4f}, TCN = {tcn_test_metrics[metric]:.4f}")

# McNemar's Test
print("\n[INFO] Performing McNemar's Test on Test Set Predictions...")
lstm_preds = lstm_test_metrics['binary_preds']
tcn_preds = tcn_test_metrics['binary_preds']
y_test_labels = y_test_tensor.flatten().cpu().numpy()

contingency = np.zeros((2, 2))
for l, t, y in zip(lstm_preds, tcn_preds, y_test_labels):
    l_correct = l == y
    t_correct = t == y
    contingency[int(l_correct), int(t_correct)] += 1

mcnemar_result = ct.mcnemar(contingency, exact=False, correction=True)
print(f"McNemar's Test: Statistic = {mcnemar_result.statistic:.4f}, p-value = {mcnemar_result.pvalue:.4f}")
if mcnemar_result.pvalue < 0.05:
    print("Significant difference between LSTM and TCN (p < 0.05)")
else:
    print("No significant difference between LSTM and TCN (p >= 0.05)")
```

Supplementary File 2

[INFO] Downloading HRS data from Google Drive...
Downloading...
From (original): https://drive.google.com/uc?id=1ckrS7U_eZS8EWcB_1iljlt8vnLdone0z
From (redirected): https://drive.google.com/uc?id=1ckrS7U_eZS8EWcB_1iljlt8vnLdone0z&confirm=t&uuid=343d7b9d-1be3-4b0c-8eca-1c65f0fc562a
To: /content/randhrs1992_2020v2.sav
100%|██████████| 915M/915M [00:05<00:00, 169MB/s]
[INFO] Verifying variables...
Available cognitive columns: ['R11COGTOT', 'R12COGTOT', 'R13COGTOT']
Available health columns: ['R11HIBPS', 'R11DIABS', 'R11CANCRS', 'R11LUNGS', 'R11HEARTS', 'R11STROKS', 'R11PSYCHS', 'R11ARTHRS', 'R12HIBPS', 'R13HIBPS', 'R14HIBPS', 'R15HIBPS', 'R12DIABS', 'R13DIABS', 'R14DIABS', 'R15DIABS', 'R12CANCRS', 'R13CANCRS', 'R14CANCRS', 'R15CANCRS', 'R12LUNGS', 'R13LUNGS', 'R14LUNGS', 'R15LUNGS', 'R12HEARTS', 'R13HEARTS', 'R14HEARTS', 'R15HEARTS', 'R12STROKS', 'R13STROKS', 'R14STROKS', 'R15STROKS', 'R12PSYCHS', 'R13PSYCHS', 'R14PSYCHS', 'R15PSYCHS', 'R12ARTHRS', 'R13ARTHRS', 'R14ARTHRS', 'R15ARTHRS']
Available dental columns: ['R11DENTST', 'R12DENTST', 'R13DENTST', 'R14DENTST', 'R15DENTST']
[INFO] Codebook saved as hrs_codebook.csv

[INFO] Fold 1/5
[INFO] Training LSTM Model for Fold 1...
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/rnn.py:123: UserWarning: dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.6 and num_layers=1
  warnings.warn(
Epoch 1/50, Train Loss: 0.2597, Train Accuracy: 0.8811, Val Loss: 0.1297, Val Accuracy: 0.9524, Val F1: 0.9304
Epoch 2/50, Train Loss: 0.0851, Train Accuracy: 0.9709, Val Loss: 0.0575, Val Accuracy: 0.9836, Val F1: 0.9756
Epoch 3/50, Train Loss: 0.0499, Train Accuracy: 0.9834, Val Loss: 0.0578, Val Accuracy: 0.9881, Val F1: 0.9823
Epoch 4/50, Train Loss: 0.0372, Train Accuracy: 0.9876, Val Loss: 0.0351, Val Accuracy: 0.9899, Val F1: 0.9850
Epoch 5/50, Train Loss: 0.0310, Train Accuracy: 0.9897, Val Loss: 0.0314, Val Accuracy: 0.9912, Val F1: 0.9868
Epoch 6/50, Train Loss: 0.0259, Train Accuracy: 0.9916, Val Loss: 0.0296, Val Accuracy: 0.9911, Val F1: 0.9866
Epoch 7/50, Train Loss: 0.0235, Train Accuracy: 0.9922, Val Loss: 0.0226, Val Accuracy: 0.9943, Val F1: 0.9915
Epoch 8/50, Train Loss: 0.0158, Train Accuracy: 0.9954, Val Loss: 0.0209, Val Accuracy: 0.9943, Val F1: 0.9915
Epoch 9/50, Train Loss: 0.0149, Train Accuracy: 0.9955, Val Loss: 0.0178, Val Accuracy: 0.9950, Val F1: 0.9925
Epoch 10/50, Train Loss: 0.0139, Train Accuracy: 0.9957, Val Loss: 0.0166, Val Accuracy: 0.9959, Val F1: 0.9939
Epoch 11/50, Train Loss: 0.0124, Train Accuracy: 0.9961, Val Loss: 0.0155, Val Accuracy: 0.9958, Val F1: 0.9937
Epoch 12/50, Train Loss: 0.0124, Train Accuracy: 0.9960, Val Loss: 0.0160, Val Accuracy: 0.9960, Val F1: 0.9940
Epoch 13/50, Train Loss: 0.0131, Train Accuracy: 0.9957, Val Loss: 0.0153, Val Accuracy: 0.9961, Val F1: 0.9942
Epoch 14/50, Train Loss: 0.0098, Train Accuracy: 0.9970, Val Loss: 0.0159, Val Accuracy: 0.9959, Val F1: 0.9939
Epoch 15/50, Train Loss: 0.0094, Train Accuracy: 0.9972, Val Loss: 0.0143, Val Accuracy: 0.9958, Val F1: 0.9937
Epoch 16/50, Train Loss: 0.0091, Train Accuracy: 0.9971, Val Loss: 0.0143, Val Accuracy: 0.9961, Val F1: 0.9942
Epoch 17/50, Train Loss: 0.0090, Train Accuracy: 0.9972, Val Loss: 0.0142, Val Accuracy: 0.9961, Val F1: 0.9942
Epoch 18/50, Train Loss: 0.0087, Train Accuracy: 0.9972, Val Loss: 0.0140, Val Accuracy: 0.9958, Val F1: 0.9937
Epoch 19/50, Train Loss: 0.0087, Train Accuracy: 0.9972, Val Loss: 0.0139, Val Accuracy: 0.9958, Val F1: 0.9938
Epoch 20/50, Train Loss: 0.0071, Train Accuracy: 0.9979, Val Loss: 0.0141, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 21/50, Train Loss: 0.0072, Train Accuracy: 0.9978, Val Loss: 0.0127, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 22/50, Train Loss: 0.0069, Train Accuracy: 0.9978, Val Loss: 0.0123, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 23/50, Train Loss: 0.0070, Train Accuracy: 0.9980, Val Loss: 0.0127, Val Accuracy: 0.9965, Val F1: 0.9948
Epoch 24/50, Train Loss: 0.0071, Train Accuracy: 0.9978, Val Loss: 0.0118, Val Accuracy: 0.9967, Val F1: 0.9950
Epoch 25/50, Train Loss: 0.0070, Train Accuracy: 0.9979, Val Loss: 0.0121, Val Accuracy: 0.9966, Val F1: 0.9949
Epoch 26/50, Train Loss: 0.0063, Train Accuracy: 0.9983, Val Loss: 0.0117, Val Accuracy: 0.9966, Val F1: 0.9949
Epoch 27/50, Train Loss: 0.0063, Train Accuracy: 0.9981, Val Loss: 0.0119, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 28/50, Train Loss: 0.0061, Train Accuracy: 0.9981, Val Loss: 0.0122, Val Accuracy: 0.9967, Val F1: 0.9950
Epoch 29/50, Train Loss: 0.0060, Train Accuracy: 0.9983, Val Loss: 0.0126, Val Accuracy: 0.9968, Val F1: 0.9952
Epoch 30/50, Train Loss: 0.0060, Train Accuracy: 0.9982, Val Loss: 0.0119, Val Accuracy: 0.9969, Val F1: 0.9954
Epoch 31/50, Train Loss: 0.0059, Train Accuracy: 0.9981, Val Loss: 0.0122, Val Accuracy: 0.9966, Val F1: 0.9949
Epoch 32/50, Train Loss: 0.0057, Train Accuracy: 0.9982, Val Loss: 0.0119, Val Accuracy: 0.9969, Val F1: 0.9954

Epoch 33/50, Train Loss: 0.0057, Train Accuracy: 0.9984, Val Loss: 0.0116, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 34/50, Train Loss: 0.0056, Train Accuracy: 0.9985, Val Loss: 0.0116, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 35/50, Train Loss: 0.0056, Train Accuracy: 0.9984, Val Loss: 0.0115, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 36/50, Train Loss: 0.0056, Train Accuracy: 0.9983, Val Loss: 0.0117, Val Accuracy: 0.9968, Val F1: 0.9952
Epoch 37/50, Train Loss: 0.0056, Train Accuracy: 0.9984, Val Loss: 0.0119, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 38/50, Train Loss: 0.0054, Train Accuracy: 0.9984, Val Loss: 0.0118, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 39/50, Train Loss: 0.0055, Train Accuracy: 0.9985, Val Loss: 0.0120, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 40/50, Train Loss: 0.0054, Train Accuracy: 0.9985, Val Loss: 0.0120, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 41/50, Train Loss: 0.0054, Train Accuracy: 0.9984, Val Loss: 0.0115, Val Accuracy: 0.9972, Val F1: 0.9957
Epoch 42/50, Train Loss: 0.0054, Train Accuracy: 0.9985, Val Loss: 0.0118, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 43/50, Train Loss: 0.0053, Train Accuracy: 0.9985, Val Loss: 0.0120, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 44/50, Train Loss: 0.0053, Train Accuracy: 0.9984, Val Loss: 0.0117, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 45/50, Train Loss: 0.0052, Train Accuracy: 0.9986, Val Loss: 0.0115, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 46/50, Train Loss: 0.0053, Train Accuracy: 0.9985, Val Loss: 0.0118, Val Accuracy: 0.9973, Val F1: 0.9960
Epoch 47/50, Train Loss: 0.0053, Train Accuracy: 0.9986, Val Loss: 0.0115, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 48/50, Train Loss: 0.0052, Train Accuracy: 0.9985, Val Loss: 0.0117, Val Accuracy: 0.9972, Val F1: 0.9958
Epoch 49/50, Train Loss: 0.0053, Train Accuracy: 0.9985, Val Loss: 0.0115, Val Accuracy: 0.9972, Val F1: 0.9959
Epoch 50/50, Train Loss: 0.0052, Train Accuracy: 0.9985, Val Loss: 0.0117, Val Accuracy: 0.9972, Val F1: 0.9958
[INFO] Training TCN Model for Fold 1...
Epoch 1/50, Train Loss: 0.5070, Train Accuracy: 0.7481, Val Loss: 0.4065, Val Accuracy: 0.7638, Val F1: 0.7299
Epoch 2/50, Train Loss: 0.4240, Train Accuracy: 0.7707, Val Loss: 0.3803, Val Accuracy: 0.7655, Val F1: 0.7325
Epoch 3/50, Train Loss: 0.4076, Train Accuracy: 0.7761, Val Loss: 0.3682, Val Accuracy: 0.7811, Val F1: 0.7436
Epoch 4/50, Train Loss: 0.3921, Train Accuracy: 0.7846, Val Loss: 0.3568, Val Accuracy: 0.8094, Val F1: 0.7581
Epoch 5/50, Train Loss: 0.3829, Train Accuracy: 0.7894, Val Loss: 0.3474, Val Accuracy: 0.8154, Val F1: 0.7676
Epoch 6/50, Train Loss: 0.3771, Train Accuracy: 0.7990, Val Loss: 0.3395, Val Accuracy: 0.8178, Val F1: 0.7741
Epoch 7/50, Train Loss: 0.3729, Train Accuracy: 0.7994, Val Loss: 0.3362, Val Accuracy: 0.8150, Val F1: 0.7717
Epoch 8/50, Train Loss: 0.3662, Train Accuracy: 0.8026, Val Loss: 0.3299, Val Accuracy: 0.8151, Val F1: 0.7726
Epoch 9/50, Train Loss: 0.3625, Train Accuracy: 0.8040, Val Loss: 0.3253, Val Accuracy: 0.8191, Val F1: 0.7751
Epoch 10/50, Train Loss: 0.3618, Train Accuracy: 0.8049, Val Loss: 0.3218, Val Accuracy: 0.8131, Val F1: 0.7717
Epoch 11/50, Train Loss: 0.3578, Train Accuracy: 0.8078, Val Loss: 0.3203, Val Accuracy: 0.8178, Val F1: 0.7733
Epoch 12/50, Train Loss: 0.3542, Train Accuracy: 0.8072, Val Loss: 0.3200, Val Accuracy: 0.8171, Val F1: 0.7761
Epoch 13/50, Train Loss: 0.3520, Train Accuracy: 0.8074, Val Loss: 0.3137, Val Accuracy: 0.8151, Val F1: 0.7756
Epoch 14/50, Train Loss: 0.3526, Train Accuracy: 0.8070, Val Loss: 0.3129, Val Accuracy: 0.8201, Val F1: 0.7755
Epoch 15/50, Train Loss: 0.3516, Train Accuracy: 0.8086, Val Loss: 0.3139, Val Accuracy: 0.8176, Val F1: 0.7771
Epoch 16/50, Train Loss: 0.3500, Train Accuracy: 0.8113, Val Loss: 0.3135, Val Accuracy: 0.8186, Val F1: 0.7765
Epoch 17/50, Train Loss: 0.3483, Train Accuracy: 0.8088, Val Loss: 0.3115, Val Accuracy: 0.8174, Val F1: 0.7769
Epoch 18/50, Train Loss: 0.3464, Train Accuracy: 0.8097, Val Loss: 0.3109, Val Accuracy: 0.8219, Val F1: 0.7782
Epoch 19/50, Train Loss: 0.3468, Train Accuracy: 0.8087, Val Loss: 0.3114, Val Accuracy: 0.8126, Val F1: 0.7751
Epoch 20/50, Train Loss: 0.3461, Train Accuracy: 0.8113, Val Loss: 0.3093, Val Accuracy: 0.8213, Val F1: 0.7807
Epoch 21/50, Train Loss: 0.3443, Train Accuracy: 0.8119, Val Loss: 0.3092, Val Accuracy: 0.8205, Val F1: 0.7802
Epoch 22/50, Train Loss: 0.3447, Train Accuracy: 0.8116, Val Loss: 0.3089, Val Accuracy: 0.8220, Val F1: 0.7779
Epoch 23/50, Train Loss: 0.3470, Train Accuracy: 0.8105, Val Loss: 0.3082, Val Accuracy: 0.8198, Val F1: 0.7792
Epoch 24/50, Train Loss: 0.3409, Train Accuracy: 0.8142, Val Loss: 0.3067, Val Accuracy: 0.8161, Val F1: 0.7781
Epoch 25/50, Train Loss: 0.3462, Train Accuracy: 0.8120, Val Loss: 0.3083, Val Accuracy: 0.8194, Val F1: 0.7788
Epoch 26/50, Train Loss: 0.3438, Train Accuracy: 0.8105, Val Loss: 0.3075, Val Accuracy: 0.8212, Val F1: 0.7806
Epoch 27/50, Train Loss: 0.3437, Train Accuracy: 0.8119, Val Loss: 0.3073, Val Accuracy: 0.8187, Val F1: 0.7765
Early stopping at epoch 28
Fold 1 - LSTM Val F1: 0.9960, TCN Val F1: 0.7807

[INFO] Fold 2/5
[INFO] Training LSTM Model for Fold 2...
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/rnn.py:123: UserWarning: dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.6 and num_layers=1
  warnings.warn(
Epoch 1/50, Train Loss: 0.2708, Train Accuracy: 0.8740, Val Loss: 0.1294, Val Accuracy: 0.9575, Val F1: 0.9384
Epoch 2/50, Train Loss: 0.0894, Train Accuracy: 0.9693, Val Loss: 0.0587, Val Accuracy: 0.9823, Val F1: 0.9736
Epoch 3/50, Train Loss: 0.0510, Train Accuracy: 0.9830, Val Loss: 0.0402, Val Accuracy: 0.9883, Val F1: 0.9826
Epoch 4/50, Train Loss: 0.0375, Train Accuracy: 0.9880, Val Loss: 0.0372, Val Accuracy: 0.9899, Val F1: 0.9849
Epoch 5/50, Train Loss: 0.0308, Train Accuracy: 0.9906, Val Loss: 0.0298, Val Accuracy: 0.9938, Val F1: 0.9907

Epoch 6/50, Train Loss: 0.0253, Train Accuracy: 0.9921, Val Loss: 0.0234, Val Accuracy: 0.9930, Val F1: 0.9895
Epoch 7/50, Train Loss: 0.0237, Train Accuracy: 0.9923, Val Loss: 0.0209, Val Accuracy: 0.9953, Val F1: 0.9929
Epoch 8/50, Train Loss: 0.0161, Train Accuracy: 0.9951, Val Loss: 0.0172, Val Accuracy: 0.9954, Val F1: 0.9932
Epoch 9/50, Train Loss: 0.0160, Train Accuracy: 0.9949, Val Loss: 0.0179, Val Accuracy: 0.9952, Val F1: 0.9928
Epoch 10/50, Train Loss: 0.0148, Train Accuracy: 0.9950, Val Loss: 0.0188, Val Accuracy: 0.9961, Val F1: 0.9942
Epoch 11/50, Train Loss: 0.0140, Train Accuracy: 0.9957, Val Loss: 0.0235, Val Accuracy: 0.9928, Val F1: 0.9893
Epoch 12/50, Train Loss: 0.0134, Train Accuracy: 0.9957, Val Loss: 0.0254, Val Accuracy: 0.9939, Val F1: 0.9908
Epoch 13/50, Train Loss: 0.0124, Train Accuracy: 0.9959, Val Loss: 0.0200, Val Accuracy: 0.9956, Val F1: 0.9934
Epoch 14/50, Train Loss: 0.0103, Train Accuracy: 0.9967, Val Loss: 0.0170, Val Accuracy: 0.9961, Val F1: 0.9942
Epoch 15/50, Train Loss: 0.0096, Train Accuracy: 0.9968, Val Loss: 0.0151, Val Accuracy: 0.9959, Val F1: 0.9939
Epoch 16/50, Train Loss: 0.0098, Train Accuracy: 0.9968, Val Loss: 0.0146, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 17/50, Train Loss: 0.0095, Train Accuracy: 0.9969, Val Loss: 0.0138, Val Accuracy: 0.9969, Val F1: 0.9954
Epoch 18/50, Train Loss: 0.0089, Train Accuracy: 0.9975, Val Loss: 0.0195, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 19/50, Train Loss: 0.0097, Train Accuracy: 0.9968, Val Loss: 0.0141, Val Accuracy: 0.9966, Val F1: 0.9949
Epoch 20/50, Train Loss: 0.0076, Train Accuracy: 0.9977, Val Loss: 0.0130, Val Accuracy: 0.9968, Val F1: 0.9952
Epoch 21/50, Train Loss: 0.0072, Train Accuracy: 0.9978, Val Loss: 0.0147, Val Accuracy: 0.9964, Val F1: 0.9946
Epoch 22/50, Train Loss: 0.0074, Train Accuracy: 0.9978, Val Loss: 0.0122, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 23/50, Train Loss: 0.0073, Train Accuracy: 0.9977, Val Loss: 0.0129, Val Accuracy: 0.9968, Val F1: 0.9952
Epoch 24/50, Train Loss: 0.0072, Train Accuracy: 0.9978, Val Loss: 0.0135, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 25/50, Train Loss: 0.0071, Train Accuracy: 0.9978, Val Loss: 0.0122, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 26/50, Train Loss: 0.0064, Train Accuracy: 0.9980, Val Loss: 0.0140, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 27/50, Train Loss: 0.0065, Train Accuracy: 0.9979, Val Loss: 0.0121, Val Accuracy: 0.9969, Val F1: 0.9954
Epoch 28/50, Train Loss: 0.0063, Train Accuracy: 0.9981, Val Loss: 0.0130, Val Accuracy: 0.9965, Val F1: 0.9948
Epoch 29/50, Train Loss: 0.0063, Train Accuracy: 0.9980, Val Loss: 0.0126, Val Accuracy: 0.9972, Val F1: 0.9958
Epoch 30/50, Train Loss: 0.0062, Train Accuracy: 0.9982, Val Loss: 0.0121, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 31/50, Train Loss: 0.0063, Train Accuracy: 0.9980, Val Loss: 0.0117, Val Accuracy: 0.9973, Val F1: 0.9960
Epoch 32/50, Train Loss: 0.0058, Train Accuracy: 0.9983, Val Loss: 0.0130, Val Accuracy: 0.9975, Val F1: 0.9962
Epoch 33/50, Train Loss: 0.0058, Train Accuracy: 0.9982, Val Loss: 0.0119, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 34/50, Train Loss: 0.0059, Train Accuracy: 0.9982, Val Loss: 0.0124, Val Accuracy: 0.9971, Val F1: 0.9956
Epoch 35/50, Train Loss: 0.0057, Train Accuracy: 0.9983, Val Loss: 0.0126, Val Accuracy: 0.9969, Val F1: 0.9954
Epoch 36/50, Train Loss: 0.0057, Train Accuracy: 0.9983, Val Loss: 0.0123, Val Accuracy: 0.9968, Val F1: 0.9952
Epoch 37/50, Train Loss: 0.0058, Train Accuracy: 0.9982, Val Loss: 0.0121, Val Accuracy: 0.9969, Val F1: 0.9954
Epoch 38/50, Train Loss: 0.0056, Train Accuracy: 0.9982, Val Loss: 0.0118, Val Accuracy: 0.9972, Val F1: 0.9958
Epoch 39/50, Train Loss: 0.0056, Train Accuracy: 0.9983, Val Loss: 0.0119, Val Accuracy: 0.9971, Val F1: 0.9956
Early stopping at epoch 40
[INFO] Training TCN Model for Fold 2...
Epoch 1/50, Train Loss: 0.5195, Train Accuracy: 0.7461, Val Loss: 0.4090, Val Accuracy: 0.7571, Val F1: 0.7257
Epoch 2/50, Train Loss: 0.4300, Train Accuracy: 0.7645, Val Loss: 0.3813, Val Accuracy: 0.7661, Val F1: 0.7327
Epoch 3/50, Train Loss: 0.4086, Train Accuracy: 0.7745, Val Loss: 0.3696, Val Accuracy: 0.8043, Val F1: 0.7579
Epoch 4/50, Train Loss: 0.3950, Train Accuracy: 0.7850, Val Loss: 0.3583, Val Accuracy: 0.8017, Val F1: 0.7562
Epoch 5/50, Train Loss: 0.3853, Train Accuracy: 0.7851, Val Loss: 0.3504, Val Accuracy: 0.7979, Val F1: 0.7565
Epoch 6/50, Train Loss: 0.3801, Train Accuracy: 0.7914, Val Loss: 0.3384, Val Accuracy: 0.7996, Val F1: 0.7589
Epoch 7/50, Train Loss: 0.3727, Train Accuracy: 0.7943, Val Loss: 0.3343, Val Accuracy: 0.8009, Val F1: 0.7608
Epoch 8/50, Train Loss: 0.3685, Train Accuracy: 0.7951, Val Loss: 0.3309, Val Accuracy: 0.8047, Val F1: 0.7649
Epoch 9/50, Train Loss: 0.3670, Train Accuracy: 0.7942, Val Loss: 0.3308, Val Accuracy: 0.8035, Val F1: 0.7646
Epoch 10/50, Train Loss: 0.3650, Train Accuracy: 0.7973, Val Loss: 0.3280, Val Accuracy: 0.8103, Val F1: 0.7699
Epoch 11/50, Train Loss: 0.3597, Train Accuracy: 0.7986, Val Loss: 0.3290, Val Accuracy: 0.8032, Val F1: 0.7643
Epoch 12/50, Train Loss: 0.3609, Train Accuracy: 0.7954, Val Loss: 0.3264, Val Accuracy: 0.8038, Val F1: 0.7669
Epoch 13/50, Train Loss: 0.3596, Train Accuracy: 0.7964, Val Loss: 0.3277, Val Accuracy: 0.8046, Val F1: 0.7664
Epoch 14/50, Train Loss: 0.3566, Train Accuracy: 0.8000, Val Loss: 0.3278, Val Accuracy: 0.8026, Val F1: 0.7658
Epoch 15/50, Train Loss: 0.3574, Train Accuracy: 0.8010, Val Loss: 0.3242, Val Accuracy: 0.8118, Val F1: 0.7723
Epoch 16/50, Train Loss: 0.3547, Train Accuracy: 0.8015, Val Loss: 0.3239, Val Accuracy: 0.8023, Val F1: 0.7653
Epoch 17/50, Train Loss: 0.3535, Train Accuracy: 0.8036, Val Loss: 0.3243, Val Accuracy: 0.8032, Val F1: 0.7642
Epoch 18/50, Train Loss: 0.3548, Train Accuracy: 0.8022, Val Loss: 0.3244, Val Accuracy: 0.8041, Val F1: 0.7676
Epoch 19/50, Train Loss: 0.3533, Train Accuracy: 0.8009, Val Loss: 0.3240, Val Accuracy: 0.8032, Val F1: 0.7653
Epoch 20/50, Train Loss: 0.3531, Train Accuracy: 0.8022, Val Loss: 0.3235, Val Accuracy: 0.8037, Val F1: 0.7669
Epoch 21/50, Train Loss: 0.3509, Train Accuracy: 0.8016, Val Loss: 0.3228, Val Accuracy: 0.8039, Val F1: 0.7672
Epoch 22/50, Train Loss: 0.3520, Train Accuracy: 0.8017, Val Loss: 0.3225, Val Accuracy: 0.8042, Val F1: 0.7676
Early stopping at epoch 23
Fold 2 - LSTM Val F1: 0.9962, TCN Val F1: 0.7723

[INFO] Fold 3/5
[INFO] Training LSTM Model for Fold 3...
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/rnn.py:123: UserWarning: dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.6 and num_layers=1
  warnings.warn(
Epoch 1/50, Train Loss: 0.2709, Train Accuracy: 0.8722, Val Loss: 0.1245, Val Accuracy: 0.9554, Val F1: 0.9343
Epoch 2/50, Train Loss: 0.0857, Train Accuracy: 0.9696, Val Loss: 0.0657, Val Accuracy: 0.9798, Val F1: 0.9700
Epoch 3/50, Train Loss: 0.0492, Train Accuracy: 0.9839, Val Loss: 0.0657, Val Accuracy: 0.9827, Val F1: 0.9744
Epoch 4/50, Train Loss: 0.0382, Train Accuracy: 0.9875, Val Loss: 0.0358, Val Accuracy: 0.9907, Val F1: 0.9861
Epoch 5/50, Train Loss: 0.0302, Train Accuracy: 0.9904, Val Loss: 0.0385, Val Accuracy: 0.9879, Val F1: 0.9818
Epoch 6/50, Train Loss: 0.0268, Train Accuracy: 0.9913, Val Loss: 0.0254, Val Accuracy: 0.9932, Val F1: 0.9899
Epoch 7/50, Train Loss: 0.0231, Train Accuracy: 0.9930, Val Loss: 0.0259, Val Accuracy: 0.9920, Val F1: 0.9881
Epoch 8/50, Train Loss: 0.0159, Train Accuracy: 0.9952, Val Loss: 0.0259, Val Accuracy: 0.9933, Val F1: 0.9900
Epoch 9/50, Train Loss: 0.0152, Train Accuracy: 0.9955, Val Loss: 0.0196, Val Accuracy: 0.9952, Val F1: 0.9928
Epoch 10/50, Train Loss: 0.0144, Train Accuracy: 0.9955, Val Loss: 0.0167, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 11/50, Train Loss: 0.0148, Train Accuracy: 0.9953, Val Loss: 0.0165, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 12/50, Train Loss: 0.0129, Train Accuracy: 0.9960, Val Loss: 0.0164, Val Accuracy: 0.9956, Val F1: 0.9934
Epoch 13/50, Train Loss: 0.0127, Train Accuracy: 0.9959, Val Loss: 0.0192, Val Accuracy: 0.9957, Val F1: 0.9935
Epoch 14/50, Train Loss: 0.0101, Train Accuracy: 0.9969, Val Loss: 0.0174, Val Accuracy: 0.9957, Val F1: 0.9936
Epoch 15/50, Train Loss: 0.0095, Train Accuracy: 0.9972, Val Loss: 0.0143, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 16/50, Train Loss: 0.0095, Train Accuracy: 0.9969, Val Loss: 0.0143, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 17/50, Train Loss: 0.0099, Train Accuracy: 0.9967, Val Loss: 0.0135, Val Accuracy: 0.9969, Val F1: 0.9954
Epoch 18/50, Train Loss: 0.0095, Train Accuracy: 0.9973, Val Loss: 0.0156, Val Accuracy: 0.9965, Val F1: 0.9948
Epoch 19/50, Train Loss: 0.0089, Train Accuracy: 0.9972, Val Loss: 0.0145, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 20/50, Train Loss: 0.0077, Train Accuracy: 0.9977, Val Loss: 0.0128, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 21/50, Train Loss: 0.0072, Train Accuracy: 0.9977, Val Loss: 0.0124, Val Accuracy: 0.9967, Val F1: 0.9950
Epoch 22/50, Train Loss: 0.0075, Train Accuracy: 0.9976, Val Loss: 0.0136, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 23/50, Train Loss: 0.0074, Train Accuracy: 0.9976, Val Loss: 0.0124, Val Accuracy: 0.9968, Val F1: 0.9953
Epoch 24/50, Train Loss: 0.0074, Train Accuracy: 0.9977, Val Loss: 0.0120, Val Accuracy: 0.9970, Val F1: 0.9955
Epoch 25/50, Train Loss: 0.0074, Train Accuracy: 0.9980, Val Loss: 0.0119, Val Accuracy: 0.9973, Val F1: 0.9960
Epoch 26/50, Train Loss: 0.0065, Train Accuracy: 0.9981, Val Loss: 0.0120, Val Accuracy: 0.9976, Val F1: 0.9965
Epoch 27/50, Train Loss: 0.0064, Train Accuracy: 0.9981, Val Loss: 0.0117, Val Accuracy: 0.9973, Val F1: 0.9960
Epoch 28/50, Train Loss: 0.0065, Train Accuracy: 0.9980, Val Loss: 0.0119, Val Accuracy: 0.9972, Val F1: 0.9959
Epoch 29/50, Train Loss: 0.0064, Train Accuracy: 0.9982, Val Loss: 0.0117, Val Accuracy: 0.9972, Val F1: 0.9959
Epoch 30/50, Train Loss: 0.0064, Train Accuracy: 0.9981, Val Loss: 0.0123, Val Accuracy: 0.9972, Val F1: 0.9957
Epoch 31/50, Train Loss: 0.0062, Train Accuracy: 0.9982, Val Loss: 0.0115, Val Accuracy: 0.9972, Val F1: 0.9959
Epoch 32/50, Train Loss: 0.0058, Train Accuracy: 0.9982, Val Loss: 0.0115, Val Accuracy: 0.9972, Val F1: 0.9959
Epoch 33/50, Train Loss: 0.0058, Train Accuracy: 0.9984, Val Loss: 0.0118, Val Accuracy: 0.9973, Val F1: 0.9960
Early stopping at epoch 34
[INFO] Training TCN Model for Fold 3...
Epoch 1/50, Train Loss: 0.5234, Train Accuracy: 0.7503, Val Loss: 0.4285, Val Accuracy: 0.7567, Val F1: 0.7235
Epoch 2/50, Train Loss: 0.4470, Train Accuracy: 0.7590, Val Loss: 0.3984, Val Accuracy: 0.7611, Val F1: 0.7319
Epoch 3/50, Train Loss: 0.4234, Train Accuracy: 0.7659, Val Loss: 0.3842, Val Accuracy: 0.7584, Val F1: 0.7281
Epoch 4/50, Train Loss: 0.4101, Train Accuracy: 0.7740, Val Loss: 0.3746, Val Accuracy: 0.7700, Val F1: 0.7360
Epoch 5/50, Train Loss: 0.4005, Train Accuracy: 0.7754, Val Loss: 0.3689, Val Accuracy: 0.7674, Val F1: 0.7349
Epoch 6/50, Train Loss: 0.3963, Train Accuracy: 0.7813, Val Loss: 0.3663, Val Accuracy: 0.7808, Val F1: 0.7440
Epoch 7/50, Train Loss: 0.3898, Train Accuracy: 0.7839, Val Loss: 0.3620, Val Accuracy: 0.7812, Val F1: 0.7449
Epoch 8/50, Train Loss: 0.3862, Train Accuracy: 0.7827, Val Loss: 0.3620, Val Accuracy: 0.7735, Val F1: 0.7405
Epoch 9/50, Train Loss: 0.3856, Train Accuracy: 0.7853, Val Loss: 0.3586, Val Accuracy: 0.7808, Val F1: 0.7449
Epoch 10/50, Train Loss: 0.3820, Train Accuracy: 0.7881, Val Loss: 0.3584, Val Accuracy: 0.7638, Val F1: 0.7317
Epoch 11/50, Train Loss: 0.3815, Train Accuracy: 0.7860, Val Loss: 0.3575, Val Accuracy: 0.7630, Val F1: 0.7332
Epoch 12/50, Train Loss: 0.3773, Train Accuracy: 0.7873, Val Loss: 0.3550, Val Accuracy: 0.7727, Val F1: 0.7394
Epoch 13/50, Train Loss: 0.3778, Train Accuracy: 0.7888, Val Loss: 0.3551, Val Accuracy: 0.7735, Val F1: 0.7410
Epoch 14/50, Train Loss: 0.3792, Train Accuracy: 0.7883, Val Loss: 0.3536, Val Accuracy: 0.7730, Val F1: 0.7401
Epoch 15/50, Train Loss: 0.3767, Train Accuracy: 0.7856, Val Loss: 0.3548, Val Accuracy: 0.7662, Val F1: 0.7352
Epoch 16/50, Train Loss: 0.3755, Train Accuracy: 0.7895, Val Loss: 0.3534, Val Accuracy: 0.7658, Val F1: 0.7346
Early stopping at epoch 17
Fold 3 - LSTM Val F1: 0.9965, TCN Val F1: 0.7449

[INFO] Fold 4/5
[INFO] Training LSTM Model for Fold 4...
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/rnn.py:123: UserWarning: dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.6 and num_layers=1
  warnings.warn(
Epoch 1/50, Train Loss: 0.2649, Train Accuracy: 0.8795, Val Loss: 0.1281, Val Accuracy: 0.9575, Val F1: 0.9377
Epoch 2/50, Train Loss: 0.0900, Train Accuracy: 0.9686, Val Loss: 0.0537, Val Accuracy: 0.9852, Val F1: 0.9779
Epoch 3/50, Train Loss: 0.0541, Train Accuracy: 0.9820, Val Loss: 0.0551, Val Accuracy: 0.9871, Val F1: 0.9806
Epoch 4/50, Train Loss: 0.0411, Train Accuracy: 0.9861, Val Loss: 0.0502, Val Accuracy: 0.9886, Val F1: 0.9828
Epoch 5/50, Train Loss: 0.0334, Train Accuracy: 0.9891, Val Loss: 0.0316, Val Accuracy: 0.9903, Val F1: 0.9855
Epoch 6/50, Train Loss: 0.0282, Train Accuracy: 0.9904, Val Loss: 0.0252, Val Accuracy: 0.9928, Val F1: 0.9893
Epoch 7/50, Train Loss: 0.0248, Train Accuracy: 0.9922, Val Loss: 0.0233, Val Accuracy: 0.9942, Val F1: 0.9913
Epoch 8/50, Train Loss: 0.0172, Train Accuracy: 0.9950, Val Loss: 0.0162, Val Accuracy: 0.9954, Val F1: 0.9932
Epoch 9/50, Train Loss: 0.0162, Train Accuracy: 0.9954, Val Loss: 0.0237, Val Accuracy: 0.9945, Val F1: 0.9917
Epoch 10/50, Train Loss: 0.0160, Train Accuracy: 0.9949, Val Loss: 0.0160, Val Accuracy: 0.9954, Val F1: 0.9930
Epoch 11/50, Train Loss: 0.0152, Train Accuracy: 0.9954, Val Loss: 0.0259, Val Accuracy: 0.9967, Val F1: 0.9951
Epoch 12/50, Train Loss: 0.0144, Train Accuracy: 0.9952, Val Loss: 0.0148, Val Accuracy: 0.9966, Val F1: 0.9949
Epoch 13/50, Train Loss: 0.0141, Train Accuracy: 0.9956, Val Loss: 0.0142, Val Accuracy: 0.9957, Val F1: 0.9936
Epoch 14/50, Train Loss: 0.0107, Train Accuracy: 0.9966, Val Loss: 0.0134, Val Accuracy: 0.9959, Val F1: 0.9939
Epoch 15/50, Train Loss: 0.0106, Train Accuracy: 0.9966, Val Loss: 0.0143, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 16/50, Train Loss: 0.0099, Train Accuracy: 0.9970, Val Loss: 0.0127, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 17/50, Train Loss: 0.0102, Train Accuracy: 0.9968, Val Loss: 0.0149, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 18/50, Train Loss: 0.0096, Train Accuracy: 0.9973, Val Loss: 0.0137, Val Accuracy: 0.9959, Val F1: 0.9939
Early stopping at epoch 19
[INFO] Training TCN Model for Fold 4...
Epoch 1/50, Train Loss: 0.5264, Train Accuracy: 0.7421, Val Loss: 0.4200, Val Accuracy: 0.7617, Val F1: 0.7276
Epoch 2/50, Train Loss: 0.4438, Train Accuracy: 0.7614, Val Loss: 0.3945, Val Accuracy: 0.7652, Val F1: 0.7335
Epoch 3/50, Train Loss: 0.4213, Train Accuracy: 0.7682, Val Loss: 0.3797, Val Accuracy: 0.7662, Val F1: 0.7350
Epoch 4/50, Train Loss: 0.4062, Train Accuracy: 0.7723, Val Loss: 0.3704, Val Accuracy: 0.7795, Val F1: 0.7383
Epoch 5/50, Train Loss: 0.3989, Train Accuracy: 0.7793, Val Loss: 0.3631, Val Accuracy: 0.7818, Val F1: 0.7382
Epoch 6/50, Train Loss: 0.3895, Train Accuracy: 0.7818, Val Loss: 0.3580, Val Accuracy: 0.7837, Val F1: 0.7460
Epoch 7/50, Train Loss: 0.3856, Train Accuracy: 0.7855, Val Loss: 0.3552, Val Accuracy: 0.7831, Val F1: 0.7454
Epoch 8/50, Train Loss: 0.3800, Train Accuracy: 0.7856, Val Loss: 0.3531, Val Accuracy: 0.7919, Val F1: 0.7505
Epoch 9/50, Train Loss: 0.3796, Train Accuracy: 0.7862, Val Loss: 0.3520, Val Accuracy: 0.7926, Val F1: 0.7529
Epoch 10/50, Train Loss: 0.3767, Train Accuracy: 0.7840, Val Loss: 0.3526, Val Accuracy: 0.7881, Val F1: 0.7491
Epoch 11/50, Train Loss: 0.3766, Train Accuracy: 0.7854, Val Loss: 0.3502, Val Accuracy: 0.7915, Val F1: 0.7519
Epoch 12/50, Train Loss: 0.3760, Train Accuracy: 0.7848, Val Loss: 0.3479, Val Accuracy: 0.7779, Val F1: 0.7420
Epoch 13/50, Train Loss: 0.3734, Train Accuracy: 0.7877, Val Loss: 0.3479, Val Accuracy: 0.7797, Val F1: 0.7435
Epoch 14/50, Train Loss: 0.3711, Train Accuracy: 0.7879, Val Loss: 0.3463, Val Accuracy: 0.7847, Val F1: 0.7464
Epoch 15/50, Train Loss: 0.3710, Train Accuracy: 0.7883, Val Loss: 0.3464, Val Accuracy: 0.7822, Val F1: 0.7438
Epoch 16/50, Train Loss: 0.3687, Train Accuracy: 0.7886, Val Loss: 0.3463, Val Accuracy: 0.7834, Val F1: 0.7467
Early stopping at epoch 17
Fold 4 - LSTM Val F1: 0.9951, TCN Val F1: 0.7529

[INFO] Fold 5/5
[INFO] Training LSTM Model for Fold 5...
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/rnn.py:123: UserWarning: dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.6 and num_layers=1
  warnings.warn(
Epoch 1/50, Train Loss: 0.2625, Train Accuracy: 0.8775, Val Loss: 0.1319, Val Accuracy: 0.9525, Val F1: 0.9308
Epoch 2/50, Train Loss: 0.0882, Train Accuracy: 0.9688, Val Loss: 0.0808, Val Accuracy: 0.9755, Val F1: 0.9634
Epoch 3/50, Train Loss: 0.0523, Train Accuracy: 0.9829, Val Loss: 0.0446, Val Accuracy: 0.9859, Val F1: 0.9789
Epoch 4/50, Train Loss: 0.0382, Train Accuracy: 0.9873, Val Loss: 0.0438, Val Accuracy: 0.9873, Val F1: 0.9811
Epoch 5/50, Train Loss: 0.0321, Train Accuracy: 0.9902, Val Loss: 0.0334, Val Accuracy: 0.9901, Val F1: 0.9852
Epoch 6/50, Train Loss: 0.0270, Train Accuracy: 0.9914, Val Loss: 0.0300, Val Accuracy: 0.9916, Val F1: 0.9874
Epoch 7/50, Train Loss: 0.0248, Train Accuracy: 0.9923, Val Loss: 0.0313, Val Accuracy: 0.9927, Val F1: 0.9891
Epoch 8/50, Train Loss: 0.0174, Train Accuracy: 0.9949, Val Loss: 0.0237, Val Accuracy: 0.9924, Val F1: 0.9887
Epoch 9/50, Train Loss: 0.0168, Train Accuracy: 0.9944, Val Loss: 0.0200, Val Accuracy: 0.9938, Val F1: 0.9907

Epoch 10/50, Train Loss: 0.0152, Train Accuracy: 0.9953, Val Loss: 0.0207, Val Accuracy: 0.9943, Val F1: 0.9915
Epoch 11/50, Train Loss: 0.0152, Train Accuracy: 0.9954, Val Loss: 0.0199, Val Accuracy: 0.9931, Val F1: 0.9898
Epoch 12/50, Train Loss: 0.0145, Train Accuracy: 0.9953, Val Loss: 0.0243, Val Accuracy: 0.9925, Val F1: 0.9888
Epoch 13/50, Train Loss: 0.0138, Train Accuracy: 0.9955, Val Loss: 0.0178, Val Accuracy: 0.9943, Val F1: 0.9915
Epoch 14/50, Train Loss: 0.0107, Train Accuracy: 0.9967, Val Loss: 0.0186, Val Accuracy: 0.9939, Val F1: 0.9908
Epoch 15/50, Train Loss: 0.0106, Train Accuracy: 0.9968, Val Loss: 0.0143, Val Accuracy: 0.9952, Val F1: 0.9928
Epoch 16/50, Train Loss: 0.0104, Train Accuracy: 0.9968, Val Loss: 0.0166, Val Accuracy: 0.9945, Val F1: 0.9917
Epoch 17/50, Train Loss: 0.0106, Train Accuracy: 0.9967, Val Loss: 0.0180, Val Accuracy: 0.9958, Val F1: 0.9937
Epoch 18/50, Train Loss: 0.0099, Train Accuracy: 0.9971, Val Loss: 0.0137, Val Accuracy: 0.9952, Val F1: 0.9928
Epoch 19/50, Train Loss: 0.0100, Train Accuracy: 0.9970, Val Loss: 0.0181, Val Accuracy: 0.9955, Val F1: 0.9933
Epoch 20/50, Train Loss: 0.0084, Train Accuracy: 0.9975, Val Loss: 0.0155, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 21/50, Train Loss: 0.0082, Train Accuracy: 0.9975, Val Loss: 0.0148, Val Accuracy: 0.9962, Val F1: 0.9944
Epoch 22/50, Train Loss: 0.0079, Train Accuracy: 0.9976, Val Loss: 0.0141, Val Accuracy: 0.9959, Val F1: 0.9939
Epoch 23/50, Train Loss: 0.0080, Train Accuracy: 0.9978, Val Loss: 0.0161, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 24/50, Train Loss: 0.0078, Train Accuracy: 0.9978, Val Loss: 0.0150, Val Accuracy: 0.9951, Val F1: 0.9927
Epoch 25/50, Train Loss: 0.0077, Train Accuracy: 0.9976, Val Loss: 0.0143, Val Accuracy: 0.9953, Val F1: 0.9929
Epoch 26/50, Train Loss: 0.0071, Train Accuracy: 0.9980, Val Loss: 0.0145, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 27/50, Train Loss: 0.0070, Train Accuracy: 0.9980, Val Loss: 0.0163, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 28/50, Train Loss: 0.0068, Train Accuracy: 0.9980, Val Loss: 0.0132, Val Accuracy: 0.9964, Val F1: 0.9946
Epoch 29/50, Train Loss: 0.0070, Train Accuracy: 0.9979, Val Loss: 0.0123, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 30/50, Train Loss: 0.0068, Train Accuracy: 0.9980, Val Loss: 0.0155, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 31/50, Train Loss: 0.0069, Train Accuracy: 0.9981, Val Loss: 0.0143, Val Accuracy: 0.9957, Val F1: 0.9935
Epoch 32/50, Train Loss: 0.0065, Train Accuracy: 0.9981, Val Loss: 0.0123, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 33/50, Train Loss: 0.0064, Train Accuracy: 0.9981, Val Loss: 0.0138, Val Accuracy: 0.9965, Val F1: 0.9948
Epoch 34/50, Train Loss: 0.0064, Train Accuracy: 0.9981, Val Loss: 0.0123, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 35/50, Train Loss: 0.0063, Train Accuracy: 0.9983, Val Loss: 0.0123, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 36/50, Train Loss: 0.0063, Train Accuracy: 0.9981, Val Loss: 0.0126, Val Accuracy: 0.9965, Val F1: 0.9948
Epoch 37/50, Train Loss: 0.0061, Train Accuracy: 0.9982, Val Loss: 0.0133, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 38/50, Train Loss: 0.0061, Train Accuracy: 0.9981, Val Loss: 0.0124, Val Accuracy: 0.9964, Val F1: 0.9946
Epoch 39/50, Train Loss: 0.0061, Train Accuracy: 0.9982, Val Loss: 0.0127, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 40/50, Train Loss: 0.0061, Train Accuracy: 0.9981, Val Loss: 0.0124, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 41/50, Train Loss: 0.0060, Train Accuracy: 0.9983, Val Loss: 0.0133, Val Accuracy: 0.9965, Val F1: 0.9947
Epoch 42/50, Train Loss: 0.0060, Train Accuracy: 0.9982, Val Loss: 0.0124, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 43/50, Train Loss: 0.0060, Train Accuracy: 0.9984, Val Loss: 0.0131, Val Accuracy: 0.9962, Val F1: 0.9943
Early stopping at epoch 44
[INFO] Training TCN Model for Fold 5...
Epoch 1/50, Train Loss: 0.5233, Train Accuracy: 0.7430, Val Loss: 0.4098, Val Accuracy: 0.7584, Val F1: 0.7281
Epoch 2/50, Train Loss: 0.4328, Train Accuracy: 0.7670, Val Loss: 0.3800, Val Accuracy: 0.7662, Val F1: 0.7341
Epoch 3/50, Train Loss: 0.4090, Train Accuracy: 0.7789, Val Loss: 0.3682, Val Accuracy: 0.7826, Val F1: 0.7450
Epoch 4/50, Train Loss: 0.3975, Train Accuracy: 0.7873, Val Loss: 0.3597, Val Accuracy: 0.8070, Val F1: 0.7621
Epoch 5/50, Train Loss: 0.3859, Train Accuracy: 0.7972, Val Loss: 0.3504, Val Accuracy: 0.8184, Val F1: 0.7703
Epoch 6/50, Train Loss: 0.3811, Train Accuracy: 0.7977, Val Loss: 0.3409, Val Accuracy: 0.8053, Val F1: 0.7632
Epoch 7/50, Train Loss: 0.3742, Train Accuracy: 0.8029, Val Loss: 0.3359, Val Accuracy: 0.8122, Val F1: 0.7660
Epoch 8/50, Train Loss: 0.3688, Train Accuracy: 0.8034, Val Loss: 0.3323, Val Accuracy: 0.8104, Val F1: 0.7674
Epoch 9/50, Train Loss: 0.3706, Train Accuracy: 0.8035, Val Loss: 0.3310, Val Accuracy: 0.8037, Val F1: 0.7660
Epoch 10/50, Train Loss: 0.3641, Train Accuracy: 0.8042, Val Loss: 0.3278, Val Accuracy: 0.8060, Val F1: 0.7676
Epoch 11/50, Train Loss: 0.3645, Train Accuracy: 0.8056, Val Loss: 0.3259, Val Accuracy: 0.8157, Val F1: 0.7717
Epoch 12/50, Train Loss: 0.3622, Train Accuracy: 0.8054, Val Loss: 0.3236, Val Accuracy: 0.8080, Val F1: 0.7687
Epoch 13/50, Train Loss: 0.3622, Train Accuracy: 0.8042, Val Loss: 0.3239, Val Accuracy: 0.8078, Val F1: 0.7703
Epoch 14/50, Train Loss: 0.3585, Train Accuracy: 0.8077, Val Loss: 0.3232, Val Accuracy: 0.8174, Val F1: 0.7718
Epoch 15/50, Train Loss: 0.3592, Train Accuracy: 0.8061, Val Loss: 0.3206, Val Accuracy: 0.8215, Val F1: 0.7794
Epoch 16/50, Train Loss: 0.3597, Train Accuracy: 0.8065, Val Loss: 0.3205, Val Accuracy: 0.8202, Val F1: 0.7748
Epoch 17/50, Train Loss: 0.3583, Train Accuracy: 0.8070, Val Loss: 0.3188, Val Accuracy: 0.8169, Val F1: 0.7757
Epoch 18/50, Train Loss: 0.3589, Train Accuracy: 0.8081, Val Loss: 0.3191, Val Accuracy: 0.8102, Val F1: 0.7730
Epoch 19/50, Train Loss: 0.3559, Train Accuracy: 0.8098, Val Loss: 0.3183, Val Accuracy: 0.8185, Val F1: 0.7721
Epoch 20/50, Train Loss: 0.3572, Train Accuracy: 0.8098, Val Loss: 0.3169, Val Accuracy: 0.8195, Val F1: 0.7740
Epoch 21/50, Train Loss: 0.3583, Train Accuracy: 0.8063, Val Loss: 0.3171, Val Accuracy: 0.8204, Val F1: 0.7753
Epoch 22/50, Train Loss: 0.3537, Train Accuracy: 0.8104, Val Loss: 0.3160, Val Accuracy: 0.8203, Val F1: 0.7775
Early stopping at epoch 23
Fold 5 - LSTM Val F1: 0.9948, TCN Val F1: 0.7794

[SUMMARY] Cross-Validation Results:
LSTM Avg Val F1: 0.9957 ± 0.0007
TCN Avg Val F1: 0.7660 ± 0.0145

[INFO] Training LSTM Model for Final Evaluation...
/usr/local/lib/python3.11/dist-packages/torch/nn/modules/rnn.py:123: UserWarning: dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.6 and num_layers=1
  warnings.warn(
Epoch 1/50, Train Loss: 0.2621, Train Accuracy: 0.8795, Val Loss: 0.1340, Val Accuracy: 0.9517, Val F1: 0.9295
Epoch 2/50, Train Loss: 0.0851, Train Accuracy: 0.9704, Val Loss: 0.0671, Val Accuracy: 0.9787, Val F1: 0.9685
Epoch 3/50, Train Loss: 0.0505, Train Accuracy: 0.9834, Val Loss: 0.0469, Val Accuracy: 0.9834, Val F1: 0.9750
Epoch 4/50, Train Loss: 0.0375, Train Accuracy: 0.9875, Val Loss: 0.0394, Val Accuracy: 0.9882, Val F1: 0.9823
Epoch 5/50, Train Loss: 0.0310, Train Accuracy: 0.9894, Val Loss: 0.0271, Val Accuracy: 0.9917, Val F1: 0.9876
Epoch 6/50, Train Loss: 0.0269, Train Accuracy: 0.9915, Val Loss: 0.0299, Val Accuracy: 0.9894, Val F1: 0.9841
Epoch 7/50, Train Loss: 0.0247, Train Accuracy: 0.9919, Val Loss: 0.0261, Val Accuracy: 0.9921, Val F1: 0.9882
Epoch 8/50, Train Loss: 0.0165, Train Accuracy: 0.9948, Val Loss: 0.0251, Val Accuracy: 0.9930, Val F1: 0.9895
Epoch 9/50, Train Loss: 0.0154, Train Accuracy: 0.9952, Val Loss: 0.0225, Val Accuracy: 0.9939, Val F1: 0.9908
Epoch 10/50, Train Loss: 0.0145, Train Accuracy: 0.9950, Val Loss: 0.0184, Val Accuracy: 0.9948, Val F1: 0.9922
Epoch 11/50, Train Loss: 0.0144, Train Accuracy: 0.9953, Val Loss: 0.0186, Val Accuracy: 0.9938, Val F1: 0.9907
Epoch 12/50, Train Loss: 0.0140, Train Accuracy: 0.9954, Val Loss: 0.0182, Val Accuracy: 0.9955, Val F1: 0.9933
Epoch 13/50, Train Loss: 0.0137, Train Accuracy: 0.9955, Val Loss: 0.0172, Val Accuracy: 0.9945, Val F1: 0.9917
Epoch 14/50, Train Loss: 0.0099, Train Accuracy: 0.9970, Val Loss: 0.0172, Val Accuracy: 0.9950, Val F1: 0.9926
Epoch 15/50, Train Loss: 0.0095, Train Accuracy: 0.9970, Val Loss: 0.0163, Val Accuracy: 0.9949, Val F1: 0.9923
Epoch 16/50, Train Loss: 0.0097, Train Accuracy: 0.9968, Val Loss: 0.0200, Val Accuracy: 0.9958, Val F1: 0.9938
Epoch 17/50, Train Loss: 0.0095, Train Accuracy: 0.9971, Val Loss: 0.0208, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 18/50, Train Loss: 0.0090, Train Accuracy: 0.9971, Val Loss: 0.0153, Val Accuracy: 0.9955, Val F1: 0.9933
Epoch 19/50, Train Loss: 0.0090, Train Accuracy: 0.9973, Val Loss: 0.0158, Val Accuracy: 0.9953, Val F1: 0.9929
Epoch 20/50, Train Loss: 0.0077, Train Accuracy: 0.9977, Val Loss: 0.0143, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 21/50, Train Loss: 0.0075, Train Accuracy: 0.9977, Val Loss: 0.0141, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 22/50, Train Loss: 0.0075, Train Accuracy: 0.9979, Val Loss: 0.0150, Val Accuracy: 0.9960, Val F1: 0.9940
Epoch 23/50, Train Loss: 0.0075, Train Accuracy: 0.9977, Val Loss: 0.0136, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 24/50, Train Loss: 0.0075, Train Accuracy: 0.9978, Val Loss: 0.0138, Val Accuracy: 0.9961, Val F1: 0.9941
Epoch 25/50, Train Loss: 0.0072, Train Accuracy: 0.9977, Val Loss: 0.0144, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 26/50, Train Loss: 0.0066, Train Accuracy: 0.9981, Val Loss: 0.0144, Val Accuracy: 0.9964, Val F1: 0.9946
Epoch 27/50, Train Loss: 0.0065, Train Accuracy: 0.9980, Val Loss: 0.0140, Val Accuracy: 0.9964, Val F1: 0.9946
Epoch 28/50, Train Loss: 0.0065, Train Accuracy: 0.9981, Val Loss: 0.0137, Val Accuracy: 0.9962, Val F1: 0.9943
Epoch 29/50, Train Loss: 0.0064, Train Accuracy: 0.9981, Val Loss: 0.0141, Val Accuracy: 0.9957, Val F1: 0.9936
Epoch 30/50, Train Loss: 0.0063, Train Accuracy: 0.9981, Val Loss: 0.0150, Val Accuracy: 0.9961, Val F1: 0.9942
Epoch 31/50, Train Loss: 0.0065, Train Accuracy: 0.9980, Val Loss: 0.0143, Val Accuracy: 0.9957, Val F1: 0.9935
Epoch 32/50, Train Loss: 0.0059, Train Accuracy: 0.9982, Val Loss: 0.0133, Val Accuracy: 0.9963, Val F1: 0.9945
Epoch 33/50, Train Loss: 0.0059, Train Accuracy: 0.9981, Val Loss: 0.0136, Val Accuracy: 0.9964, Val F1: 0.9946
Early stopping at epoch 34

[INFO] Training TCN Model for Final Evaluation...
Epoch 1/50, Train Loss: 0.5277, Train Accuracy: 0.7496, Val Loss: 0.4199, Val Accuracy: 0.7607, Val F1: 0.7259
Epoch 2/50, Train Loss: 0.4430, Train Accuracy: 0.7594, Val Loss: 0.3916, Val Accuracy: 0.7654, Val F1: 0.7339
Epoch 3/50, Train Loss: 0.4195, Train Accuracy: 0.7684, Val Loss: 0.3765, Val Accuracy: 0.7811, Val F1: 0.7412
Epoch 4/50, Train Loss: 0.4080, Train Accuracy: 0.7782, Val Loss: 0.3647, Val Accuracy: 0.7834, Val F1: 0.7444
Epoch 5/50, Train Loss: 0.3973, Train Accuracy: 0.7836, Val Loss: 0.3561, Val Accuracy: 0.7874, Val F1: 0.7487
Epoch 6/50, Train Loss: 0.3923, Train Accuracy: 0.7879, Val Loss: 0.3486, Val Accuracy: 0.7822, Val F1: 0.7466
Epoch 7/50, Train Loss: 0.3877, Train Accuracy: 0.7895, Val Loss: 0.3471, Val Accuracy: 0.7963, Val F1: 0.7539
Epoch 8/50, Train Loss: 0.3805, Train Accuracy: 0.7946, Val Loss: 0.3429, Val Accuracy: 0.7941, Val F1: 0.7514
Epoch 9/50, Train Loss: 0.3799, Train Accuracy: 0.7974, Val Loss: 0.3405, Val Accuracy: 0.7963, Val F1: 0.7533
Epoch 10/50, Train Loss: 0.3788, Train Accuracy: 0.7932, Val Loss: 0.3413, Val Accuracy: 0.7975, Val F1: 0.7555
Epoch 11/50, Train Loss: 0.3758, Train Accuracy: 0.7986, Val Loss: 0.3380, Val Accuracy: 0.7872, Val F1: 0.7469
Epoch 12/50, Train Loss: 0.3754, Train Accuracy: 0.7988, Val Loss: 0.3396, Val Accuracy: 0.7934, Val F1: 0.7517
Epoch 13/50, Train Loss: 0.3744, Train Accuracy: 0.7998, Val Loss: 0.3366, Val Accuracy: 0.8003, Val F1: 0.7558
Epoch 14/50, Train Loss: 0.3743, Train Accuracy: 0.7983, Val Loss: 0.3366, Val Accuracy: 0.7837, Val F1: 0.7485
Epoch 15/50, Train Loss: 0.3727, Train Accuracy: 0.7976, Val Loss: 0.3360, Val Accuracy: 0.7836, Val F1: 0.7487
Epoch 16/50, Train Loss: 0.3715, Train Accuracy: 0.8006, Val Loss: 0.3345, Val Accuracy: 0.7915, Val F1: 0.7496

Epoch 17/50, Train Loss: 0.3729, Train Accuracy: 0.8021, Val Loss: 0.3345, Val Accuracy: 0.7925, Val F1: 0.7510
Epoch 18/50, Train Loss: 0.3700, Train Accuracy: 0.7993, Val Loss: 0.3322, Val Accuracy: 0.7891, Val F1: 0.7510
Epoch 19/50, Train Loss: 0.3686, Train Accuracy: 0.8030, Val Loss: 0.3329, Val Accuracy: 0.7889, Val F1: 0.7507
Epoch 20/50, Train Loss: 0.3692, Train Accuracy: 0.8009, Val Loss: 0.3312, Val Accuracy: 0.7908, Val F1: 0.7523
Early stopping at epoch 21

[INFO] Evaluating LSTM Model
LSTM Validation Metrics:
accuracy: 0.9963759552509257
precision: 0.9936305732484076
recall: 0.9955093358544079
f1: 0.99456906729634
auc: 0.9997022014136243
conf_matrix: [[8435 27]
[19 4212]]
threshold: 0.8500000000000002

LSTM Test Metrics:
accuracy: 0.9977944072469476
precision: 0.9952874646559849
recall: 0.998109640831758
f1: 0.9966965549787635
auc: 0.9999021650341853
conf_matrix: [[8443 20]
[8 4224]]
threshold: 0.5000000000000001

[INFO] Evaluating TCN Model
TCN Validation Metrics:
accuracy: 0.8002836208934058
precision: 0.6378861788617887
recall: 0.9272039706925077
f1: 0.7558038724593007
auc: 0.922340136596318
conf_matrix: [[6235 2227]
[308 3923]]
threshold: 0.45000000000000007

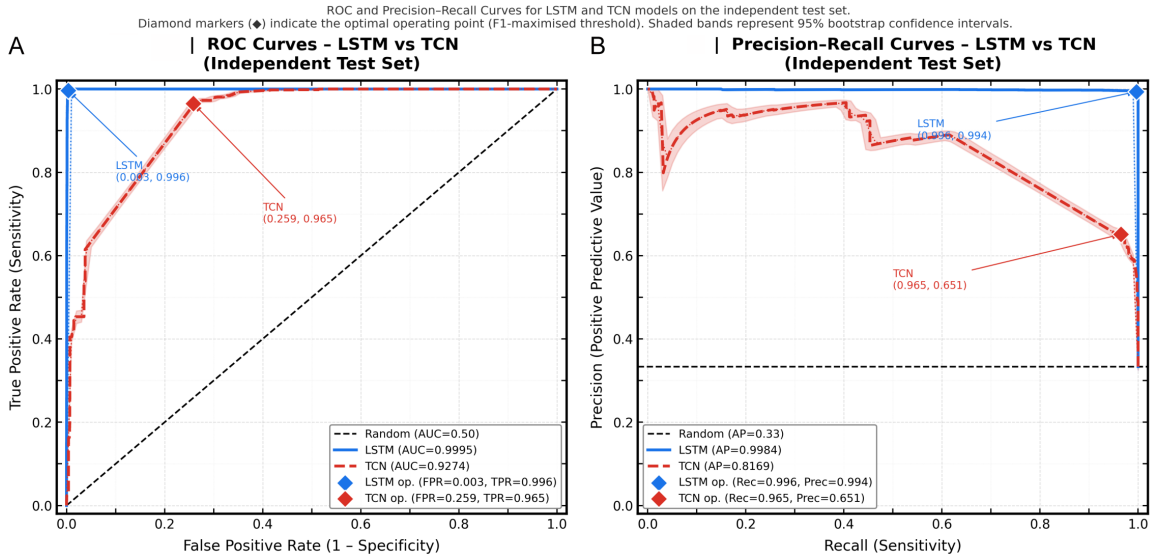
TCN Test Metrics:
accuracy: 0.7900748326112643
precision: 0.6197462937490448
recall: 0.9581758034026465
f1: 0.7526682134570766
auc: 0.92308225597603
conf_matrix: [[5975 2488]
[177 4055]]
threshold: 0.3500000000000001
[INFO] No significant overfitting: LSTM Val F1 (0.9946) vs Test F1 (0.9967)
[INFO] No significant overfitting: TCN Val F1 (0.7558) vs Test F1 (0.7527)

[SUMMARY] Model Comparison (Validation):
Accuracy: LSTM = 0.9964, TCN = 0.8003
Precision: LSTM = 0.9936, TCN = 0.6379
Recall: LSTM = 0.9955, TCN = 0.9272
F1: LSTM = 0.9946, TCN = 0.7558
Auc: LSTM = 0.9997, TCN = 0.9223

[SUMMARY] Model Comparison (Test):
Accuracy: LSTM = 0.9978, TCN = 0.7901
Precision: LSTM = 0.9953, TCN = 0.6197

Recall: LSTM = 0.9981, TCN = 0.9582
F1: LSTM = 0.9967, TCN = 0.7527
Auc: LSTM = 0.9999, TCN = 0.9231

[INFO] Performing McNemar's Test on Test Set Predictions...
McNemar's Test: Statistic = 2605.3603, p-value = 0.0000
Significant difference between LSTM and TCN ($p < 0.05$)



Supplementary Figure 1. Combined ROC and PR curves. A. ROC curves (LSTM vs TCN) ROC curves with 95% bootstrap confidence intervals (shaded areas) and optimal operating points (diamonds) for LSTM (blue) and TCN (red) models on the independent test set. The diagonal dashed line represents a random classifier (AUC=0.5). Inset shows a zoom on the 0-0.5 FPR region. B. Precision-Recall curves (LSTM vs TCN). PR curves with 95% bootstrap confidence intervals and optimal operating points. Horizontal dashed line indicates the performance of a random classifier (AP = prevalence = 0.33). Inset highlights the high-recall region (0.85-1.0).