Cureus Journal of Computer Science
https://doi.org/10.7759/s44389-026-00158-z

Original Article

# Structural Analysis of Business Data Using Convolutional Neural Network

Daniel A. Folorunso [1,]

1. *Computer Science, Federal University of Agriculture Abeokuta, Abeokuta, NGA*

Received: May 12, 2026 | Review began: June 18, 2026 | Review ended: July 06, 2026 | Published: July 14, 2026

© **Copyright** 2026
This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

## Abstract

### Background

Structural analysis in business and econometric systems traditionally relied on numerical proportionality-based modeling techniques. However, the increasing dimensionality and heterogeneity of modern business datasets have exposed limitations in conventional dimensionality reduction approaches such as Principal Component Analysis (PCA), particularly in preserving relationships between input variables and target outcomes. Deep learning approaches such as Convolutional Neural Networks (CNNs) have recently emerged as alternatives capable of automated feature extraction and adaptive structural learning.

### Methods

This study evaluated the effectiveness of a one-dimensional CNN (1D-CNN) for structural analysis of business data, using customer churn prediction as a case study, and comparatively assessed its performance against PCA-supported Artificial Neural Networks (PCA-ANNs). A customer churn dataset containing 440,882 customer records was obtained from Kaggle. The dataset consisted of numerical and categorical business variables associated with customer attrition. A preprocessing pipeline involving custom StringLookup implementations, one-hot encoding, categorical indexing, standardization, and PCA transformations was developed for the respective models. The dataset was divided into training, validation, and testing sets using a 64:16:20 ratio. A custom weighted binary cross-entropy loss function was introduced to address class imbalance during training. Multiple PCA-ANN models corresponding to varying principal component sizes were evaluated alongside the proposed 1D-CNN architecture. Model evaluation was conducted using accuracy, precision, recall, F1-score, and area under the curve.

### Results

The proposed 1D-CNN model achieved a training accuracy of 98.50%, a validation accuracy of 98.29%, and a test accuracy of 98.34%, outperforming PCA-ANN models utilizing up to eight principal components. The highest overall performance was achieved by the PCA-ANN-9 model, which attained a training accuracy of 99.25%, a validation accuracy of 99.12%, and a test accuracy of 99.13%. Despite its slightly lower predictive accuracy, the CNN architecture demonstrated advantages in automated supervised feature extraction, adaptability to heterogeneous business data, and reduced dependence on manually selected dimensionality reduction parameters.

### Conclusion

The findings demonstrate that 1D-CNN architectures provide an effective and scalable alternative for the structural analysis of business data. While the final PCA-supported ANN model achieved marginally higher predictive accuracy in this study, the CNN model offered greater flexibility, automated feature learning, and enhanced adaptability to evolving

business environments. The study further highlights the potential of deep learning-based structural modeling approaches for customer behavior analysis and predictive business analytics.

**Categories:** Business Intelligence (BI), Machine Learning (ML), Deep Learning

**Keywords:** customer churn, structural analysis, business data analytics, convolutional neural network, principal component analysis, feature extraction, econometric modelling, customer attrition prediction, dimensionality reduction

## Introduction

Structural analysis, or structural modeling, in econometrics is a branch of econometrics concerned not only with the cause-and-effect relationships among variables but also with how these variables interact collectively to produce deterministic business outcomes. Traditionally, such analyses have relied heavily on strictly numerical data with well-established direct or inverse proportional relationships. However, with the increasing dimensionality and heterogeneity of modern business data, these traditional approaches have become increasingly difficult to sustain effectively [1].

The increase in data dimensionality has led to the adoption of dimensionality reduction techniques such as Principal Component Analysis (PCA), in which variables are prioritized based on their cumulative contribution [2]. Although PCA helps address the curse of dimensionality, it presents several limitations. First, the optimal number of principal components required to achieve the best predictive performance is not readily known [3]. Second, selecting components based on a predefined variance threshold remains largely experimental because the variance structure of the data may change as additional observations become available [4]. Consequently, the relative importance of variables may shift over time, potentially affecting model performance. Furthermore, PCA reduces dimensionality solely on the basis of variance distribution rather than the direct relationships between the input variables and the target output [5]. As a result, additional experimentation is often required to determine the variance threshold that yields the optimal predictive performance.

To address these limitations and bridge the relationship gap observed in PCA-based modeling, recent studies have explored deep learning approaches such as Graph Neural Networks (GNNs), Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs), and Convolutional Neural Networks (CNNs). GNNs are designed to model relationships among multiple entities simultaneously through graph structures rather than traditional flat input-output representations [6]. LSTM-RNNs are commonly applied to sequential and time-series data, particularly in forecasting applications such as price prediction over time [7]. CNNs, originally developed for image recognition tasks involving two-dimensional data, are designed to identify and extract meaningful local features from data [8].

More recently, CNNs have been adapted for one-dimensional data applications, particularly in audio processing and the analysis of high-dimensional structured datasets. While GNNs focus on relationships among multiple interconnected entities and LSTM-RNNs model temporal dependencies, one-dimensional CNNs (1D-CNNs) primarily perform localized feature extraction within individual records. This process resembles filtering or sieving relevant patterns from raw data. Among these deep learning approaches, CNNs share the closest conceptual similarity with PCA because both aim to reduce feature complexity by extracting meaningful representations [8].

Business data encompass multiple domains and applications. Fraud detection datasets map customer actions to the likelihood of fraudulent behavior [9]. Pricing datasets analyze changes in product prices over time to estimate future trends [7]. Another important application is customer attrition analysis, in which customer experiences and interactions are mapped to the probability of reduced patronage or customer churn [10].

Customer behavior and decision-making are highly individualized, implying that the variables used in business structural analysis are often entity-specific [11]. Given this context, this study proposes the use of 1D-CNN for the structural analysis of business data, particularly customer attrition analysis. The proposed architecture performs dimensionality reduction through feature extraction while simultaneously learning the relationship between the extracted features and the target

variable. Owing to the conceptual similarities between CNNs and PCA in dimensionality reduction, this study also compares the performance of a 1D-CNN model with that of PCA-supported Artificial Neural Networks (PCA-ANNs) to evaluate their respective strengths and limitations in customer churn prediction.

## Materials And Methods

### Data collection and preparation

A customer churn dataset containing 440,882 customer records was obtained from Kaggle. Each record represented an individual customer profile, along with behavioral and transactional variables associated with the likelihood of customer churn. The dataset consisted of 11 input columns, including eight numerical variables and three categorical variables represented as object data types within the DataFrame, as shown in Table *1*. The final column, Churn, indicated whether a customer would churn (1) or not (0) and served as the target variable. The dataset contained 56.71% positive churn cases and 43.29% negative churn cases.

| CustomerID | Age | Gender | Tenure | Usage Frequency | Support Calls | Payment Delay | Subscription Type | Contract Length | Total Spend | Last Interaction | Churn |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 30 | Female | 39 | 14 | 5 | 18 | Standard | Annual | 932 | 17 | 1 |
| 3 | 65 | Female | 49 | 1 | 10 | 8 | Basic | Monthly | 557 | 6 | 1 |
| 4 | 55 | Female | 14 | 4 | 6 | 18 | Basic | Quarterly | 185 | 3 | 1 |
| 5 | 58 | Male | 38 | 21 | 7 | 7 | Standard | Monthly | 396 | 29 | 1 |
| 6 | 23 | Male | 32 | 20 | 5 | 8 | Basic | Monthly | 617 | 20 | 1 |
| 8 | 51 | Male | 33 | 25 | 9 | 26 | Premium | Annual | 129 | 8 | 1 |
| 9 | 58 | Female | 49 | 12 | 3 | 16 | Standard | Quarterly | 821 | 24 | 1 |
| 10 | 55 | Female | 37 | 8 | 4 | 15 | Premium | Annual | 445 | 30 | 1 |

**TABLE 1: Customer Churn Dataset Sample**

**How to cite this article:**
Folorunso D A (July 14, 2026) Structural Analysis of Business Data Using Convolutional Neural Network. Cureus J Comput Sci 3 : es44389-026-00158-z. DOI https://doi.org/10.7759/s44389-026-00158-z

Before preprocessing, the dataset was examined for missing values and duplicated rows. No missing values or duplicate records were identified. The CustomerID column was removed using the Pandas DataFrame *drop* function along the first axis because it did not provide generalized behavioral insights relevant to predictive modeling. This reduced the total number of inputs and numerical columns to 10 and 7, respectively [12].

Similarly, prior to the preprocessing stage, the dataset was shuffled and divided using an 80:20 ratio with a random state of 42 to ensure reproducibility through the DataFrame *sample* function. This was achieved by setting the *frac* parameter to 0.8 to select 80% of the data, while reserving the remaining 20% as unseen test data by dropping the index of the mapped 80% from the entire DataFrame. The separated 80% was further divided into training and validation sets using an 80:20 ratio. Consequently, the final data distribution from the original data corresponded to 64% training, 16% validation, and 20% testing.

**Data preprocessing**

A utility function was first created to identify categorical columns and separate them from numerical columns by checking whether their data type (dtype) was an object. Numerical columns were isolated into a dedicated DataFrame, while categorical columns were extracted independently.

To ensure data type uniformity through categorical-to-numerical transformation, a custom StringLookup layer was developed to provide TensorFlow-like functionality within the PyTorch backend used by Keras in this study. Separate StringLookup instances were created for each categorical column using their unique values as vocabularies.

Noteworthy is the difference between the output modes of the StringLookup instances employed in the PCA-ANN and CNN architectures. In the PCA-ANN pipeline, categorical variables were encoded using integer (int) representations, where categorical values were mapped directly to their corresponding indices within the list of unique variables. In contrast, the CNN pipeline employed one-hot encoding, where the first index represented the out-of-vocabulary (OOV) token. This approach provides a richer feature representation, allowing the model to learn more expressive local patterns [13] compared with the relatively compressed integer representations used in the PCA-ANN pipeline. Beyond ensuring data type uniformity, this transformation also converts all input data into numerical representations, which are typically required by machine learning models during training and inference.

After the transformation process, the data type remained numerical across the board. However, the data remained disjointed, lacking the cohesive structure required for the model to capture the complex interrelationships among features [14]. This motivated the next step in the preprocessing pipeline, which involved preparing the data as input for the PCA-ANN and CNN models. Similar to the output mode differences, the resulting inputs for the respective models retain their differences.

To ensure unified input compatibility, the freshly transformed categorical-to-numerical variables in the CNN preprocessing pipeline were merged using the Keras Concatenate layer along the final axis. The final CNN input consisted of a dictionary structure with keys corresponding to numerical_data and object_data, delegating the final unification of the distinct data to the GPU during batch-wise operations to ensure dimensional consistency and minimize CPU-to-GPU overhead, as detailed by [15].

Conversely, for the PCA-ANN pipeline, the resulting numerical outputs from the categorical encodings were first expanded using NumPy's expand_dims function before being merged horizontally using NumPy's hstack. The pipeline merged existing numerical and newly transformed numerical data into a single feature block. This approach introduces resource-intensive preprocessing, unlike in the CNN pipeline, by performing operations in the CPU memory. However, the design choice was necessary to accommodate certain pre-model operations.

The first operation involved data normalization using scikit-learn's StandardScaler. This step was important to the PCA as it ensures features are uniformly scaled, preventing outliers from disproportionately affecting the resulting variance used to select principal components [16]. Because StandardScaler operates on CPU-bound NumPy arrays rather than GPU-accelerated symbolic tensors, this preprocessing step was executed externally before initializing the Keras model graph.

Another reason the bulk of the PCA preprocessing was maintained on the CPU was the PCA itself. This was due to the similar constraint in the normalization process, where the PCA is CPU-bound and cannot execute natively on a GPU. Multiple PCA instances were generated using sklearn.decomposition. PCA with component sizes ranging from 1 to ( $\boldsymbol{n-1}$), where *n* represents the number of dataset columns excluding the target variable. Each PCA instance was fitted independently, and transformed outputs were used as separate inputs for the PCA-ANN experiments. While the n_components parameter in sklearn's PCA used strict integers to depict the number of components to preserve, it also accepted floats from 0 to 1, indicating the percentage of variance to aim for when selecting components. Ideally, this approach appears to be a more straightforward design to avoid the need for multiple instances of PCA per component. However, it is observed that using a strict variance in this case inherently rules out components that could have better informed models [17]. Furthermore, a fixed variance threshold risks dynamically altering the number of selected components as the dataset grows over time [18], making it unsustainable given the structure of existing models. This creates an impasse between resourcefulness and structure.

While the CNN pipeline locks down its preprocessing by presenting dictionary data with the heavy integration task delegated to the GPU during training, it still shares structural similarities with the PCA-ANN pipeline on certain foundational preprocessing configurations. This entailed the serialization of preprocessing artifacts from prior processes across the respective pipelines, including PCA instances, standardization parameters, and categorical lookup configurations in Pickle and JavaScript Object Notation (JSON) formats for reproducibility. This approach freezes the exact mathematical states, such as mean, variance, and projection vectors, to ensure identical transformations on future data batches. Re-fitting these parameters to new data would alter the feature-space boundaries, leading to data leakage and distorting the numerical scale and semantic meaning of the inputs expected by the downstream neural network architecture [19,20].

**PCA-ANN model architecture**

Because distinct PCA instances were generated for each component, process automation became necessary. An iterative loop was established to encompass the maximum range of components within the source data. Subsequently, a modular framework ensured the reusability of the Keras Sequential model. By mirroring the split PCA instances, separate ANN architectures were dynamically initialized and trained on their respective PCA-transformed datasets.

The architecture consisted of two fully connected Dense layers with 256 and 32 units, integrating non-linearity through the ReLU activation function per layer, followed by a final Dense layer with a unit of 1 and a sigmoid activation function for binary classification, as revealed in Figure *1*, with *n* corresponding to the number of components, depicting that each process is executed exactly once for each instance of *n*.

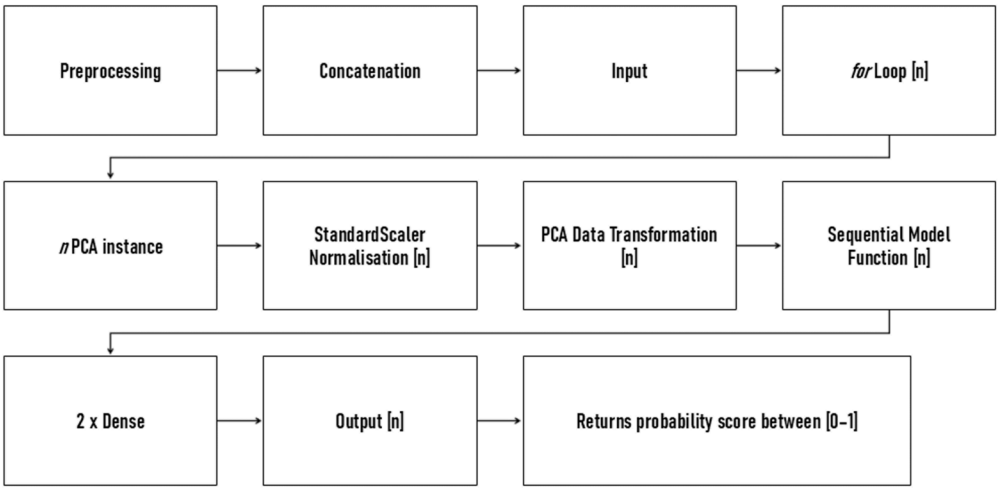


**FIGURE 1: PCA-ANN Model Development Workflow**

ANN, Artificial Neural Network; PCA, Principal Component Analysis

### CNN model architecture

The CNN model was implemented using the Keras Functional API due to its multi-input design. Two input layers corresponding to numerical_data and object_data were defined explicitly using their respective shapes 7 and 3.

The two inputs were merged using a Concatenate layer within the model architecture. Following concatenation, a Reshape layer transformed the resulting tensor from a two-dimensional structure $(\text{batch\_size}, n_{\text{numerical\_data}} + n_{\text{object\_data}})$ into a three-dimensional structure $(\text{batch\_size}, n_{\text{numerical\_data}} + n_{\text{object\_data}}, 1)$ compatible with one-dimensional convolutional operations [21].

A one-dimensional convolutional layer (Conv1D) with nine filters and a kernel size of 3 was subsequently applied using ReLU activation. The number of filters was selected through hyperparameter tuning and based on the expanded feature dimension size following preprocessing.

The convolutional layer was followed by a max-pooling (MaxPooling1D) layer with a pool size of 3 to retain the three most dominant extracted features. A BatchNormalization layer was then applied to stabilize feature distributions within each batch and improve generalization.

A Keras Flatten layer was introduced to flatten the resulting tensor for forward propagation down a similar fully connected structure as in the PCA-ANN architecture. Figure *2* shows a concise flow of the CNN's architecture up to the output layer.

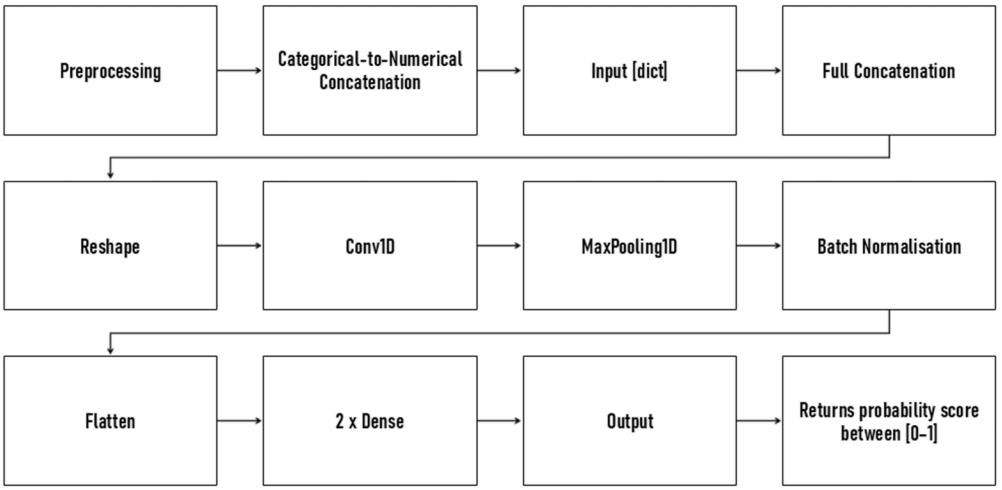


**FIGURE 2: CNN Model Development Workflow**

CNN, Convolutional Neural Network

### Weighted binary cross-entropy loss

A custom weighted binary cross-entropy loss function was developed to address class imbalance within the dataset [22], which dynamically computes class weights per batch using Keras ops. The number of positive samples within each batch was obtained by casting comparisons of the true labels ($y_{\text{true}} = 1$) to integer values ($True = 1; False = 0$) and summing the results. Negative samples were computed by subtracting the number of positives from the batch size.

The class weight was calculated as the ratio of negative samples to positive samples:

$$w = \frac{N_0}{N_1} \tag{1}$$

where $(N_0)$ represents the number of negative samples and $(N_1)$ represents the number of positive samples within the batch.

This dynamically generated weight was multiplied by the binary cross-entropy loss of the positive class before averaging across the batch. The resulting loss gradients were subsequently utilized by the Adam optimizer to update the model parameters.

Finally, the target variable (y) was extracted from the Churn column and expanded using NumPy's expand_dims function to ensure compatibility with batch-based Keras operations within the custom loss function.

### Model and log saving

Training histories for all models were saved using the Keras CSVLogger callback, allowing epoch-wise tracking of training and validation performance metrics. This comprehensive logging was critical for diagnosing model convergence, identifying potential overfitting or underfitting, and auditing performance trajectories across different principal component sizes.

Furthermore, trained models were serialized and saved as .keras files using a Keras ModelCheckpoint callback configured to capture only the best-performing epoch. Preserving these model states eliminates the need to re-run the computationally expensive and time-consuming training pipeline for subsequent evaluations. This serialization ensures that downstream analyses can be executed instantly, providing a permanent and reproducible archive for future inference.

## Results And Discussion

### Experimental setup

Experiments were conducted using Python 3.14 with libraries including NumPy, pandas, scikit-learn, pickle, and Keras.

The experimental environment consisted of a Windows 11 Home system equipped with 16 GB of RAM, 512 GB of SSD storage, and an NVIDIA RTX 3050 GPU utilizing CUDA version 13.1 with NVIDIA driver version 591.74.

Despite the presence of class imbalance within the dataset, no resampling techniques were applied in order to preserve realistic business data conditions and evaluate the effectiveness of the proposed weighted loss function.

The models were trained for 32 epochs using a batch size of 256. Evaluation metrics included accuracy, precision, recall, F1-score, and Area Under the Curve (AUC).

The F1-score threshold was set to 0.3 to improve sensitivity toward churn detection within the imbalanced dataset [23]. Additionally, weighted averaging was used instead of macro averaging to ensure that metric calculations reflected class representation proportions [24].

### Results

The proposed 1D-CNN model achieved strong predictive performance, outperforming PCA-ANN models utilizing up to eight principal components. The CNN achieved a training accuracy of 98.50% and a validation accuracy of 98.29%, compared with PCA-ANN-8, which achieved training and validation accuracies of 95.53% and 95.48%, respectively.

The only model that marginally outperformed the CNN was PCA-ANN-9, which utilized all transformed components and achieved a training accuracy of 99.25% and a validation accuracy of 99.12%, as shown in Table *2*.

**How to cite this article:**
Folorunso D A (July 14, 2026) Structural Analysis of Business Data Using Convolutional Neural Network. Cureus J Comput Sci 3 : es44389-026-00158-z. DOI https://doi.org/10.7759/s44389-026-00158-z

| Model | Accuracy | AUC | F1_Score | Loss | Precision | Recall | Val_Accuracy | Val_AUC | Val_F1_Score | Val_Loss | Val_Precision | Val_Recall |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pca ann log 1 | 83.98 % | 91.87 % | 84.69 % | 0.2683 | 88.99 % | 81.88 % | 84.03 % | 91.86 % | 84.82 % | 0.2704 | 89.07 % | 81.89 % |
| pca ann log 2 | 85.47 % | 93.02 % | 86.02 % | 0.2478 | 90.09 % | 83.57 % | 85.46 % | 93.03 % | 85.60 % | 0.2502 | 88.81 % | 85.08 % |
| pca ann log 3 | 85.50 % | 93.03 % | 86.04 % | 0.2474 | 90.31 % | 83.38 % | 85.42 % | 92.99 % | 85.87 % | 0.2506 | 89.43 % | 84.25 % |
| pca ann log 4 | 85.79 % | 93.38 % | 86.36 % | 0.2424 | 90.26 % | 84.00 % | 85.76 % | 93.39 % | 85.82 % | 0.2444 | 88.94 % | 85.53 % |
| pca ann log 5 | 89.62 % | 96.06 % | 89.83 % | 0.1846 | 93.60 % | 87.70 % | 89.36 % | 95.88 % | 89.62 % | 0.189 | 93.20 % | 87.62 % |
| pca ann log 6 | 89.96 % | 96.27 % | 90.10 % | 0.1794 | 93.84 % | 88.08 % | 89.05 % | 95.93 % | 90.27 % | 0.1936 | 95.55 % | 84.63 % |
| pca ann log 7 | 92.55 % | 97.63 % | 92.53 % | 0.1377 | 96.17 % | 90.47 % | 92.39 % | 97.57 % | 92.87 % | 0.1417 | 97.24 % | 89.11 % |
| pca ann log 8 | 95.53 % | 98.68 % | 95.52 % | 0.0924 | 98.59 % | 93.45 % | 95.48 % | 98.70 % | 95.13 % | 0.0926 | 97.82 % | 94.12 % |
| pca ann log 9 | 99.25 % | 99.95 % | 99.28 % | 0.0169 | 99.77 % | 98.91 % | 99.12 % | 99.86 % | 99.27 % | 0.0227 | 99.89 % | 98.56 % |
| cnn log | 98.50 % | 99.83 % | 98.52 % | 0.0327 | 99.31 % | 98.04 % | 98.29 % | 99.81 % | 98.70 % | 0.0391 | 99.94 % | 97.04 % |

**TABLE 2: Result Summary**

ANN, Artificial Neural Network; AUC, Area Under the Curve; CNN, Convolutional Neural Network; PCA, Principal Component Analysis

Both the 1D-CNN and PCA-ANN-9 models were further evaluated using the reserved test dataset. PCA-ANN-9 achieved a test accuracy of 99.13%, while the 1D-CNN model achieved a test accuracy of 98.34%.

The confusion matrices presented in Figure *3* further demonstrated the effectiveness of both models, with strong diagonal representations indicating accurate correspondence between predicted and actual labels.

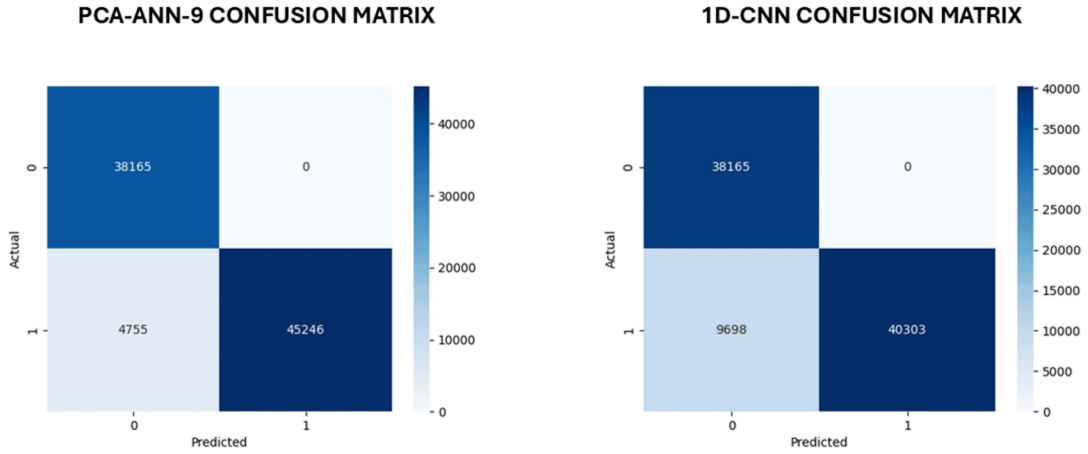


**FIGURE 3: Confusion Matrix Comparison**

ANN, Artificial Neural Network; CNN, Convolutional Neural Network; PCA, Principal Component Analysis

## Discussion

The findings demonstrate that both PCA-ANN and 1D-CNN architectures are highly effective for customer churn prediction despite the presence of moderate dataset imbalance. Although PCA-ANN-9 achieved the highest overall numerical performance, the performance gap between the two architectures remained relatively small.

This observation is significant because the 1D-CNN architecture provides several structural advantages beyond predictive accuracy alone. Unlike PCA, which reduces dimensionality solely based on variance distribution, CNNs perform supervised feature extraction by learning local patterns that are directly optimized with respect to the target variable. Consequently, the CNN architecture preserves relationships between extracted features and business outcomes more effectively.

The CNN architecture also demonstrated greater flexibility for heterogeneous business datasets. The PCA-ANN pipeline required repeated PCA fitting, component experimentation, and multiple model training cycles to determine the optimal number of components. In contrast, the CNN architecture learned feature representations directly from the expanded feature space without requiring repeated dimensionality tuning.

Additionally, the CNN model used one-hot-encoded categorical representations, preserving richer categorical information than the integer encoding required for PCA compatibility. This richer representation contributed to the CNN's strong performance.

From a deployment perspective, CNNs may also offer greater adaptability to evolving business environments where customer behavior and feature importance shift over time. Since PCA transformations are tightly coupled to variance structures within training data, changes in incoming data distributions may require the recalculation of principal components. CNNs, however, learn adaptive convolutional filters that generalize to evolving feature interactions.

The custom weighted binary cross-entropy loss function also contributed significantly to performance stability. By dynamically adjusting penalties based on class representation within each batch, the proposed loss function mitigated class imbalance without relying on oversampling or undersampling strategies.

The results of this study contribute to an evolving discourse on the application of 1D-CNNs to structured, heterogeneous business data. While contemporary literature has increasingly pivoted toward high-complexity architectures, such as GNNs [25], these models primarily operate via global, "top-to-bottom" attention mechanisms. While such frameworks offer significant insights into cross-feature dependencies, they often necessitate extensive data transformation, tokenization, or graph topology construction. In contrast, the present study demonstrates the efficacy of a "bottom-to-top" approach, focusing on the localized extraction capabilities of 1D-CNNs to maintain a seamless, integrated processing pipeline.

Recent efforts to capture behavioral trajectories in churn prediction, such as the spatio-temporal pipelines presented in [26], have demonstrated success by integrating 1D-CNNs with recurrent modules. However, these models frequently rely on data-level manipulation, such as the Synthetic Minority Oversampling Technique, to mitigate class imbalance. The findings of this study demonstrate that such preprocessing is not necessary because, through the implementation of a custom weighted binary cross-entropy loss function, the network dynamically adjusts gradient penalties during backpropagation to account for class distribution. This approach achieves stable performance on imbalanced churn data while preserving the integrity of the original feature space and avoiding the computational cost and potential bias associated with synthetic oversampling.

Within the specific domain of 1D-CNN applications for tabular data, our approaches address identified limitations in recent benchmarks. Although [27] provides a robust demonstration of feature filtering, its reliance on an external LightGBM wrapper introduces a multi-phase dependency that effectively mirrors traditional PCA-based workflows. The similarity lies in the requirement for a mandatory dimensionality reduction or ranking process before reaching the neural architecture, which introduces two limitations. First, it assumes that an external model can pre-determine which features are salient without accounting for the non-linear feature interactions the CNN is intended to learn. Second, it introduces a rigid bottleneck where the neural network is deprived of the full raw feature distribution. This study departs from this by shifting the responsibility of feature stabilization from external algorithms to the model architecture itself, allowing the 1D-CNN to learn latent representations directly from the raw, untruncated input.

Regarding the work of Anh et al. [28], their research establishes a valuable foundation for 1D-CNN application but focuses primarily on homogeneous input domains. This study expands upon their methodology by addressing the distinct challenges posed by high-variance, heterogeneous enterprise datasets. The architectural integration of an internal Batch Normalization layer addresses the primary scalability constraints identified in their work, effectively relaxing the requirement for homogeneous input collections. Consequently, this model serves as a generalized evolution of their foundational 1D-CNN architecture, facilitating its transition from controlled environments to the complex, multi-scale schemas characteristic of dynamic business analytics.

While a direct quantitative comparison with these frameworks remains constrained by their distinct data-level dependencies, such as reliance on sequential logs or specific graph topologies, this analysis underscores the viability of the proposed single-stage framework as an efficient, low-overhead alternative for enterprise churn prediction.

Despite these promising findings, the study has limitations. Only a single customer churn dataset was evaluated, limiting generalizability across broader business domains such as fraud detection, transactional forecasting, or recommendation systems.

While this study establishes the efficacy of a lightweight, single-stage 1D-CNN for heterogeneous business data, future research may further refine these foundations. Rather than adopting the high-compute bloat associated with multi-stage hybrid frameworks, future iterations could focus on optimizing the architectural sparsity of the model to automate the discovery of more efficient convolutional filters. Additionally, extending this single-stage paradigm to support multi-modal input streams remains a compelling frontier for developing next-generation, low-latency churn prediction systems.

Furthermore, Explainable Artificial Intelligence techniques such as SHapley Additive exPlanations or Gradient-weighted Class Activation Mapping may also improve the interpretability of extracted CNN features for practical business decision-making environments [29].

## Conclusions

This study investigated the structural analysis of business data using a 1D-CNN and compared its performance against PCA-supported ANNs for customer churn prediction. The findings revealed that although PCA-ANN-9 achieved the highest overall predictive performance, the proposed 1D-CNN achieved closely comparable results while offering several methodological advantages, including automatic supervised feature extraction, improved adaptability to heterogeneous datasets, and reduced dependence on manually selected dimensionality reduction parameters. The study also demonstrated the effectiveness of a custom weighted binary cross-entropy loss function for handling moderately imbalanced business datasets without explicit resampling techniques.

Overall, the results suggest that 1D-CNN architectures provide a viable and scalable alternative for structural modeling of business data, particularly in environments characterized by mixed data types, evolving feature relationships, and high-dimensional representations. The study further contributes to the growing application of deep learning techniques within econometric and business structural analysis by demonstrating the practicality of convolutional architectures for customer behavioral modeling and predictive analytics.

## Additional Information

### Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

**Concept and design:** Daniel A. Folorunso

**Acquisition, analysis, or interpretation of data:** Daniel A. Folorunso

**Drafting of the manuscript:** Daniel A. Folorunso

**Critical review of the manuscript for important intellectual content:** Daniel A. Folorunso

### Disclosures

**Human subjects:** All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

## Data Availability Statements

The data supporting this study are openly available in the Kaggle repository at https://www.kaggle.com/datasets/muhammadshahidazeem/customer-churn-dataset.

## Acknowledgements

The dataset used in this study is publicly available from the Kaggle repository: https://www.kaggle.com/datasets/muhammadshahidazeem/customer-churn-dataset. The data are anonymized and were accessed in accordance with the platform's data usage policies.

## References

1. Hünermund P, Bareinboim E: Causal inference and data fusion in econometrics. The Econometrics Journal. 2025, 28:41-82. 10.1093/ectj/utad008
2. Zhang L: Principal components analysis in data mining of e-commerce user behavior. BDMIP '23: Proceedings of the 2023 International Conference on Big Data Mining and Information Processing. 2023, 161-165. 10.1145/3645279.3645307
3. Yao Y, Ochoa A: Limitations of principal components in quantitative genetic association models for human studies. eLife. 2023, 12:e79238. 10.7554/elife.79238
4. Li H: Advancing principal component analysis: Challenges and innovations in big data analysis. Applied and Computational Engineering. 2025, 118:48-54. 10.54254/2755-2721/2025.20807
5. Fang P, Gao Z, Tsay RS: Supervised kernel principal component analysis for forecasting. Finance Research Letters. 2023, 58:104292. 10.1016/j.frl.2023.104292
6. Gao F: Consumer psychological modeling and behavior prediction system based on graph neural network. International Journal of Computational Intelligence Systems. 2026, 19:24. 10.1007/s44196-025-01093-y
7. Agarwal H, Mahajan G, Shrotriya A, Shekhawat D: Predictive data analysis: leveraging RNN and LSTM techniques for time series dataset. Procedia Computer Science. 2024, 235:979-989. 10.1016/j.procs.2024.04.093
8. Herlawati H, Handayanto R: Comparative study of PCA, t-SNE, and UMAP for CNN feature representation of image classification. PIKSEL: Penelitian Ilmu Komputer Sistem Embedded and Logic. 2025, 13:293-302. 10.33558/piksel.v13i2.11634
9. Ali A, Abd Razak S, Othman SH, et al.: Financial fraud detection based on machine learning: a systematic literature review. Applied Sciences. 2022, 12:9637. 10.3390/app12199637
10. Singh PP, Anik FI, Senapati R, Sinha A, Sakib N, Hossain E: Investigating customer churn in banking: a machine learning approach and visualization app for data science and management. Data Science and Management. 2024, 7:7-16. 10.1016/j.dsm.2023.09.002
11. Jin G, Matsukawa H: A study on the estimation of customer lifetime value using hierarchical Bayesian model. Innovation and Supply Chain Management. 2025, 19:11-16. 10.14327/iscm.19.11
12. Rana MS, Chouksey A, Das BC, Reza SA, Chowdhury MSR, Sizan MMH, Shawon RER: Evaluating the effectiveness of different machine learning models in predicting customer churn in the USA. Journal of Business and Management Studies. 2023, 5:267-281. 10.32996/jbms.2023.5.5.23
13. Alzubaidi L, Zhang J, Humaidi AJ, et al.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data. 2021, 8:53. 10.1186/s40537-021-00444-8
14. Durán-López A, Bolaños-Martinez D, Bermudez-Edo M: GCVA: a multiview fusion mechanism for heterogeneous data representations. IEEE Internet of Things Journal. 2026, 13:29754-29764. 10.1109/jiot.2026.3685702
15. Jiang Y, Nicholson H, Sanca V, Ailamaki A: Data movement-aware GPU sharing for data-intensive systems. Proceedings of the 16th Conference on Innovative Data Systems Research (CIDR 2026). 2026, 1-12.
16. Kammoun A, Ravier P, Buttelli O: Impact of PCA pre-normalization methods on ground reaction force estimation accuracy. Sensors. 2024, 24:1137. 10.3390/s24041137
17. Zheng J, Rakovski C: On the application of principal component analysis to classification problems. Data Science Journal. 2021, 20:26. 10.5334/dsj-2021-026

18. Bispo R, Marques F: Stability of principal components under normal and non-normal parent populations and different covariance structures scenarios. Journal of Statistical Computation and Simulation. 2023, 93:1060-1076. 10.1080/00949655.2022.2125971

19. Kapoor S, Narayanan A: Leakage and the reproducibility crisis in machine-learning-based science. Patterns. 2023, 4:100804. 10.1016/j.patter.2023.100804

20. Koukaras P, Tjortjis C: Data preprocessing and feature engineering for data mining: techniques, tools, and best practices. AI. 2025, 6:257. 10.3390/ai6100257

21. Priyanka, Nagpal S, Sabharwal S: MH-XAI: hybrid deep learning and XGBoost explainable AI model for mental health prediction. Concurrency and Computation: Practice and Experience. 2026, 38:e70634. 10.1002/cpe.70634

22. Atif D: VAE-INN: variational autoencoder with integrated neural network classifier for imbalanced credit scoring, utilizing weighted loss for improved accuracy. Computational Economics. 2025, 68:1213-1243. 10.1007/s10614-025-11094-w

23. Maharrani RH, Sari L, Somantri O: Comparative performance of class imbalance treatments in random forest for social engineering classification. International Journal of Safety and Security Engineering. 2026, 16:393-402. 10.18280/ijsse.160214

24. Gupta SC, Goel N: A weighted-average approach evaluation of multi-class emotion analysis model using text mining and machine learning. Cureus Journal of Computer Science. 2025, 2:1-11. 10.7759/s44389-025-04827-3

25. Bhadange S, Jha CK, Bajaj A, Sane A, Joshi M, Jha P, Gupta PP: Graph neural networks for customer churn analysis in subscription-based business models. 2025 3rd World Conference on Communication & Computing (WCONF), Raipur, India. 2025, 1-5. 10.1109/WCONF64849.2025.11233666

26. Tahsin MS, Khan RI, Hossain MY: Strategic financial management and customer segmentation: a data-driven approach to business performance optimization. Journal of Emerging Engineering Technologies. 2025, 1:19-37.

27. Mishra T, Misra S: Adaptive 1-D CNN using LSelect feature selection for predicting software faults. International Journal of Software Science and Computational Intelligence. 2026, 18:1-23. 10.4018/ijssci.399476

28. Anh VT, Ha ID, Burke K: CNN-based approaches for various types of tabular data. IEEE Access. 2025, 13:200537-200554. 10.1109/access.2025.3635724

29. Minh D, Wang HX, Li YF, Nguyen TN: Explainable artificial intelligence: a comprehensive review. Artificial Intelligence Review. 2022, 55:3503-3568. 10.1007/s10462-021-10088-y

**How to cite this article:**
Folorunso D A (July 14, 2026) Structural Analysis of Business Data Using Convolutional Neural Network. Cureus J Comput Sci 3 : es44389-026-00158-z. DOI https://doi.org/10.7759/s44389-026-00158-z 