A Factor Retention Method for Exploratory Factor Analysis Based on Long Short-Term Memory Networks

Authors: Guo Lei, Qin Haijiang, Guo Lei

Date: 2025-10-17T21:39:56+00:00

Abstract

In psychological research, determining the dimensions and characteristics of psychological traits is of paramount importance. Exploratory Factor Analysis (EFA) represents a crucial statistical method for identifying latent dimensions. Accurately identifying the number of factors constitutes a key technical challenge in EFA, as both underestimation and overestimation of factor count can lead to deleterious consequences. To accurately identify the number of factors, this study conceptualizes eigenvalues as sequential data and employs a deep neural network constructed with Long Short-Term Memory (LSTM) networks; all evaluation metrics (accuracy, precision, recall, ,) exceed 83%. Through large-scale simulation experiments and empirical studies, the performance of LSTM across various data conditions was validated. Results demonstrate that LSTM achieves superior accuracy compared to CDF, EKC, and PA methods, with an average improvement rate of 48.50% and a maximum improvement rate reaching 171.09%. Furthermore, LSTM exhibits smaller bias and greater robustness relative to CDF, EKC, and PA methods. Researchers may utilize the R package LSTMfactors to employ the LSTM model trained in this study for analyzing empirical data.

Full Text

Factor Retention in Exploratory Factor Analysis Using Long Short-Term Memory Networks

GUO Lei[1,2], **QIN Haijiang**[1]
([1] Faculty of Psychology, Southwest University; [2] Southwest University Branch, Collaborative Innovation Center of Assessment toward Basic Education Quality, Chongqing 400715, China)

Abstract
In psychological research, identifying the dimensions and characteristics of psychological traits is of paramount importance. Exploratory Factor Analysis (EFA) serves as a crucial statistical method for identifying these latent dimensions, and accurately determining the number of factors represents one of its key technical challenges. Both underestimation and overestimation of factor count can lead to adverse consequences. To address this, the present study treats eigenvalues as sequential data and employs Long Short-Term Memory (LSTM) networks to construct a deep neural network. All evaluation metrics—including accuracy, precision, recall, F1-score, and Kappa—exceed 83%. Through large-scale simulation experiments and empirical studies, the performance of LSTM across various data conditions was validated. Results demonstrate that LSTM achieves higher accuracy than CDF, EKC, and PA methods, with an average improvement rate of 48.50% and a maximum improvement of 171.09%. Moreover, LSTM exhibits smaller bias and greater robustness compared to these traditional approaches. Researchers can utilize the R package **LSTMfactors** to apply the trained LSTM model to empirical data analysis.

Keywords: exploratory factor analysis, long short-term memory, factor retention, deep learning
Classification Code: B841

1 Introduction

Psychological research focuses on latent variables such as intelligence and personality, which can only be inferred through observable manifest indicators due to their abstract and unobservable nature. Exploratory Factor Analysis (EFA) is one of the most commonly used techniques in psychological scale development, particularly when strong prior theoretical models are unavailable, as it models the relationships between one or more latent variables and a set of manifest indicators.

A critical issue in EFA is determining the correct number of factors (Zwick & Velicer, 1986). Underestimating factor count can lead to the omission of theoretically important psychological structures or sub-dimensions, resulting in loss of critical information, increased estimation errors in all factor loadings (Wood et al., 1996), and diminished accuracy of factor scores (Fava & Velicer, 1996). Conversely, overestimation may cause factor splitting, where primary loadings of manifest variables become dispersed across multiple factors after rotation, thereby weakening the association between manifest variables and their intended factors (Wood et al., 1996). Additionally, this may yield an overly complex model containing structures with minimal explanatory value (de Winter & Dodou, 2012). Therefore, clarifying the meaning of latent variables and accurately retaining the appropriate number of factors constitute essential prerequisites for reliable psychological assessment and effective clinical use of measurement tools.

To achieve accurate factor retention, researchers have proposed numerous methods. Goretzko (2025) provides a comprehensive review of current factor retention approaches, which can be broadly categorized as follows: (1) **Simple or graphical methods** that employ straightforward rules or examine graphical patterns across different factor counts, such as the eigenvalue-greater-than-one rule (Kaiser criterion; Kaiser, 1960), scree plot inspection, and the Hull method (Lorenzo-Seva et al., 2011). (2) **Sequential testing methods** that select the optimal solution from multiple possible factor counts using chi-square tests or relative fit indices (e.g., AIC, BIC). (3) **Formula-based methods** that calculate reference eigenvalues or specific metrics through specialized formulas to determine factor count (often with the Kaiser criterion as an additional constraint), such as the Empirical Kaiser Criterion (EKC; Braeken & van Assen, 2017) and the Minimum Average Partial Test (Velicer, 1976). (4) **Simulation methods** that compare actual data with simulated datasets having random or specific factor structures, including Comparison Data (CD; Ruscio & Roche, 2012) and Parallel Analysis (PA; Horn, 1965). (5) **Machine learning methods** that utilize trained models for factor retention, such as Factor Forest (FF; Goretzko & Bühner, 2020, 2022) and Comparison Data Forest (CDF; Goretzko & Ruscio, 2024). (6) **Regularization methods** that adjust minor (unnecessary) factors to zero through penalization, including regularized EFA (which adds penalties to the loading matrix and factor correlation matrix in the likelihood function) and exploratory graph analysis (which penalizes edges representing partial correlations between observed variables).

Most of these methods operate based on eigenvalues, including PA, EKC, and CDF. In fact, eigenvalues can be conceptualized as sequential data, offering a novel perspective for factor retention. When using EFA for dimension exploration, the process begins with eigenvalue decomposition of the correlation matrix derived from response data, yielding the eigenvalue vector λ and eigenvector matrix $_{J\times J}$. These three components relate as follows:

$$= \lambda^{T} \quad (1)$$

$$\mathrm{tr}(\ ) = J \quad (2)$$

where tr( ) denotes the trace of . Since the correlation matrix (a standardized covariance matrix) has diagonal elements of 1, its trace equals the number of observed variables J, and consequently, the sum of all eigenvalues is fixed at J. Typically, elements in λ are arranged in descending order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_j \geq \cdots \geq \lambda_J$. Clearly, if earlier eigenvalues are large, later eigenvalues must be smaller, reflecting the interdependence among eigenvalues and endowing them with sequential properties (Braeken & van Assen, 2017; Li et al., 2020).

Given the demonstrated accuracy advantages of artificial intelligence techniques in EFA factor retention (e.g., FF and CDF), and considering that Long Short-

Term Memory (LSTM) networks are a classic AI technology particularly effective for processing sequential data (see Section 3.1 and Online Appendix 4 for detailed introductions), this study aims to validate the feasibility of a sequential perspective on factor retention. Specifically, we: (1) treat eigenvalues extracted for EFA as time-series data and employ LSTM to process sequential features, thereby learning interdependencies among eigenvalues for factor retention; (2) conduct hyperparameter tuning to identify optimal configurations; and (3) comprehensively compare LSTM with traditional methods to validate its performance.

Among traditional methods, PA has shown strong performance across numerous simulation studies, demonstrating robustness across different data analyses and good performance under various conditions (e.g., sample sizes between 30 and 360, observed indicators between 9 and 72; Dinno, 2009; Humphreys & Montanelli, 1975; Peres-Neto et al., 2005; Zwick & Velicer, 1986), thus being regarded as a gold standard. Meanwhile, EKC emphasizes sequential changes in eigenvalues (where a reference eigenvalue' s magnitude depends on preceding eigenvalues), aligning with our sequential perspective. Additionally, CDF represents the latest method combining FF and CD, belonging to the same machine learning family as LSTM, and outperforming EKC when factor counts are high. Therefore, this study includes PA, EKC, and CDF for comparison.

This paper is organized as follows: Section 2 briefly introduces the traditional methods (PA, EKC, and CDF). Section 3 describes LSTM and its application to EFA, including training dataset generation, feature extraction, model training and validation, and hyperparameter tuning. Section 4 presents simulation experiments under typical EFA data conditions to validate LSTM' s advantages through comparison with traditional methods. Section 5 demonstrates practical application through an empirical study. Finally, we conclude with discussion and future directions.

2 Overview of PA, EKC, and CDF

Horn' s (1965) Parallel Analysis method is considered by some researchers as the "gold standard" (Auerswald & Moshagen, 2019; Goretzko, 2025). PA determines factor retention by comparing eigenvalues from actual data with those from multiple (e.g., 100; Auerswald & Moshagen, 2019) randomly generated datasets of the same dimensions. The number of eigenvalues exceeding their corresponding reference eigenvalues represents the number of factors retained. Detailed steps for PA are provided in Online Appendix 1. Auerswald and Moshagen (2019) recommend using principal component analysis for eigenvalue computation and the 95th percentile as the reference eigenvalue for optimal accuracy, a configuration adopted in this study.

The Empirical Kaiser Criterion (Braeken & van Assen, 2017) improves upon the traditional Kaiser criterion. Like its predecessor, EKC uses the critical value of 1 for factor retention but additionally considers variability in eigenval-

ues from random samples. Under the null model, eigenvalue distributions from random correlation matrices asymptotically follow the Marčenko-Pastur distribution (Marčenko & Pastur, 1967). Braeken et al. (2017) calculate reference eigenvalues from this distribution (detailed in Online Appendix 2) and compare them with empirical eigenvalues. Factors are retained when eigenvalues exceed both 1 and their corresponding reference eigenvalue, with the count determined by the same formula as PA.

Goretzko et al. (2024) proposed CDF as a novel method integrating CD and Factor Forest. Similar to PA, CDF requires simulated datasets, but its simulation approach differs. While PA uses entirely random data, CDF simulates data based on the empirical correlation matrix . The method iteratively generates correlation matrices approximating while specifying a particular factor structure, then creates simulated response data from these matrices (see Online Appendix 3 or Ruscio & Kaczetow, 2008, for details). Unlike PA' s direct eigenvalue comparison, CDF determines factor count by assessing the match between simulated and empirical data across different factor numbers.

3.1 Introduction to LSTM

LSTM is a specialized recurrent neural network designed to address gradient vanishing and explosion problems encountered by traditional RNNs when processing sequential data. Through its unique gating mechanism, LSTM effectively captures dependencies in sequential data. A complete LSTM unit comprises a memory cell and three control gates: input gate, forget gate, and output gate (Hochreiter & Schmidhuber, 1997). The memory cell serves long-term information storage, while the gates—incorporating learnable weight parameters—control selective information updates, thereby mitigating gradient issues in long-term dependency learning. Specifically, the forget gate determines which information to discard from the memory cell, the input gate controls the extent to which current input updates the memory cell state, and the output gate decides which information to extract as the current output. This gating mechanism enables LSTM to capture long-term dependencies in sequence modeling tasks, with widespread applications in language modeling, time series forecasting, and sequence tagging.

To further enhance model performance, LSTM is typically followed by one or more fully connected layers (also called dense layers). These layers map the high-dimensional feature vectors extracted by LSTM to the required output dimension (Hochreiter & Schmidhuber, 1997). This mapping reduces model complexity and enables better adaptation to different task requirements (Goodfellow et al., 2016; Brownlee, 2018). Additionally, batch normalization layers (Ioffe & Szegedy, 2015; Lange et al., 2022) can standardize features during training to accelerate network stabilization and mitigate overfitting, making them suitable for insertion after fully connected layers. This architectural combination leverages LSTM' s powerful sequential modeling capabilities while using fully connected layers for efficient feature mapping and final prediction. Read-

ers interested in the mathematical formulation and specific network architecture may consult Online Appendix 4.

3.2 LSTM Model Construction

The procedure for constructing our machine learning model follows the steps outlined in Figure 1 of Goretzko and Bühner (2020)

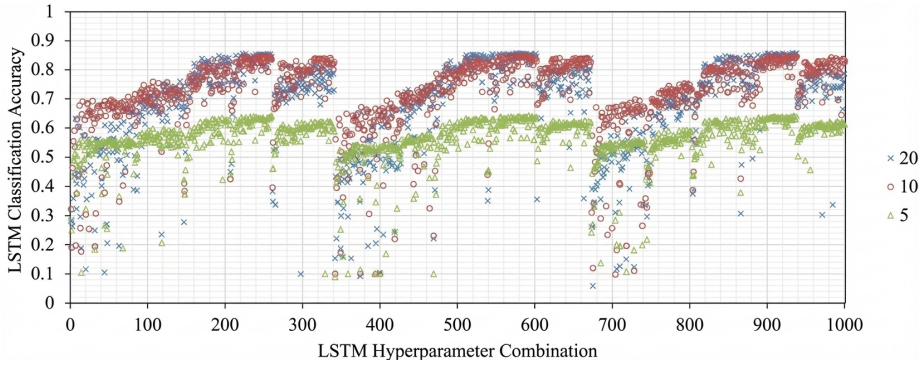

Figure 1: Figure 1

, with details available in that reference. Following Chen et al. (2017), who demonstrated that larger training datasets improve model performance, and building upon Goretzko and Bühner (2020), we generated a training dataset of 1,000,000 samples. Each dataset was generated following Auerswald and Moshagen (2019) and Goretzko and Bühner (2020): factor counts were drawn from a uniform distribution $U(1, 10)$; items per factor from $U(3, 10)$; primary loadings from $U(0.35, 0.80)$; cross-loadings from $U(-0.20, 0.20)$; interfactor correlations from $U(0.00, 0.50)$; and sample sizes from $U(100, 1000)$.

Specifically, to generate each dataset, we first compute the correlation matrix:

$$= \quad ^{T} + \ ^{2} \quad (3)$$

where represents the factor loading matrix (including both primary and cross-loadings), denotes the interfactor correlation matrix, and ^{2} is a diagonal matrix with elements $1 - \text{diag}(\ ^{T})$ to ensure diagonal elements of equal 1. To better approximate real testing scenarios, we employed the multivariate normal distribution used by Auerswald and Moshagen (2019) and Goretzko and Bühner (2020) to simulate response data:

$$X_j = L_j + \varepsilon_j, \quad 1 \le j \le J \quad (4)$$

where L_j follows a multivariate normal distribution $N(\mathbf{0}, \ )$ representing latent factor contributions, and ε_j is a residual term following a standard normal dis-

tribution. Notably, following Auerswald and Moshagen (2019) and Goretzko and Bühner (2020), we constrained L_j and ε_j to be uncorrelated, and ε_j to be uncorrelated across items, to ensure the simulated response data $\mathbf{X}$ has a correlation matrix closely approximating .

3.3 Training Feature Extraction

This study trains LSTM using sequential eigenvalue data λ to more accurately estimate factor count. Based on λ, we derived two types of training features: (1) traditional eigenvalues λ from principal component analysis, and (2) the difference between these eigenvalues and PA reference eigenvalues, i.e., $\lambda_j - \lambda_j^{\text{ref}}$. The first feature type follows conventional practice used by most existing methods. The second feature type is inspired by the gold-standard PA method, considering differences between empirical and random data variances. However, PA' s simple criterion of $I(\lambda_j - \lambda_j^{\text{ref}} > 0)$ for factor retention is vulnerable to random error. Therefore, we avoid using 0 as a fixed threshold and instead train a deep neural network to enable dynamic decision-making by LSTM.

In summary, this study uses both feature types (λ_j and $\lambda_j - \lambda_j^{\text{ref}}$) as sequential data to train an optimal LSTM for accurate factor count estimation.

3.4 Model Training and Evaluation

LSTM training was implemented using hybrid programming in R (4.5.0; R Core Team, 2025) and Python (3.13.3; Python Software Foundation, 2025). Specifically, we used the R package **reticulate** (Kalinowski et al., 2025) and Python library **pyper** (pyper, 2025) for cross-language integration, the Python library **PyTorch** (Paszke et al., 2019; PyTorch, 2025) for LSTM construction and training, CUDA 12.06 for GPU acceleration, and converted the final trained LSTM into a platform-independent ".onnx"file for long-term storage and cross-platform execution (via **onnxruntime**; Microsoft, 2025). Evaluation metrics were computed using the Python library **scikit-learn** (Pedregosa et al., 2011; scikit-learn, 2025). All code is publicly available at https://osf.io/au9vd/. Training was conducted on an Nvidia RTX 4060 GPU with 8GB VRAM, 64GB RAM, and an Intel i7-14700KF CPU.

Based on the generated 1,000,000 sample datasets, we extracted features from each dataset for LSTM training following Goretzko and Bühner' s (2020) procedure. To ensure model validity and prevent overfitting, hyperparameter tuning was performed with the following settings: (1) LSTM layers: 1-2 layers (Hochreiter & Schmidhuber, 1997; LeCun et al., 2015), with 1-40 nodes per layer (Heaton, 2008). (2) Fully connected layers: 1-5 layers (LeCun et al., 2015), with 1-40 nodes per layer (Heaton, 2008). Batch normalization layers were added after fully connected layers to enhance robustness and accelerate training, with node counts matching the preceding fully connected layers. (3) Activation functions: ReLU, Tanh, and Sigmoid (Nair & Hinton, 2010; LeCun

et al., 2015). (4) Learning rates: 0.1, 0.01, 0.001, 0.0001, and 0.00001 (Kingma & Ba, 2014).

For deep neural networks with multiple hidden layers (both fully connected and LSTM layers), grid search is inefficient due to the vast parameter space and minimal impact of many parameters (Bergstra & Bengio, 2012). Random search, which samples random hyperparameter combinations within a fixed iteration budget (e.g., 1,000 iterations), enables rapid and effective training (Bergstra & Bengio, 2012). Therefore, we employed 1,000-iteration random search for hyperparameter tuning, with a 7:3 random split for training and test sets (Qin & Guo, 2024).

Regarding sequence length for eigenvalue training features, longer sequences demand greater computational resources. Based on established factor retention methods (PA, Kaiser criterion, EKC), only the first F eigenvalues are decisive for retaining F factors, with subsequent eigenvalues contributing minimally. We therefore reasonably assumed that for factor counts between 1 and 10, a sequence length of 10 would suffice for LSTM training. Shorter lengths might reduce accuracy due to insufficient decision information, while longer lengths could introduce redundancy, substantially increasing training time without improving accuracy. To validate this assumption, we trained LSTM models with sequence lengths of 5, 10, and 20. Hyperparameter random search results (1,000 iterations each) and classification accuracies are shown in Figure 1 (detailed results available at https://osf.io/au9vd/ in "results/study I/acc.xlsx").

Figure 2

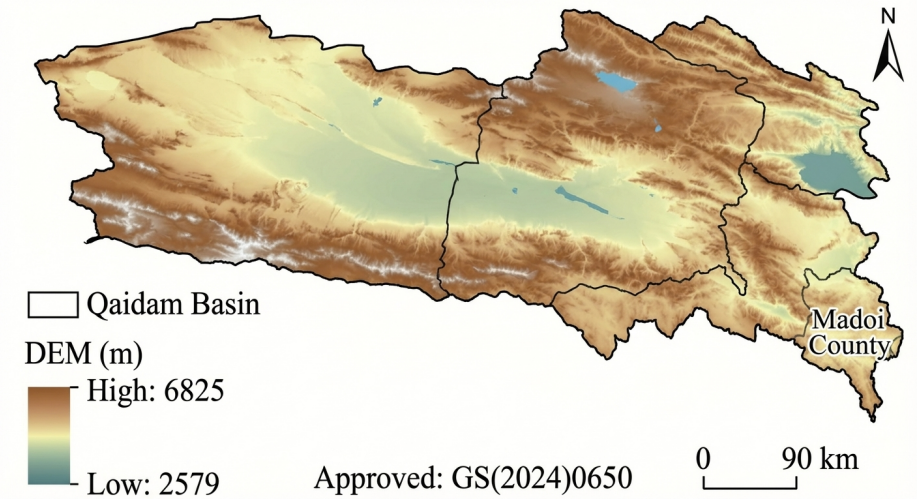

Figure 2: Figure 2

presents training time and optimal accuracy for different sequence lengths. The results clearly support our theoretical assumption: when sequence length ex-

ceeds 10, accuracy improvements become negligible. Increasing length from 10 to 20 yielded only a 0.97% accuracy gain while adding 80,522 seconds (approximately 22.37 hours) of training time. Evidently, excessive sequence length provides no substantial precision benefit while incurring significant computational costs. Therefore, to balance accuracy and practical applicability, we adopted a sequence length of 10 for all subsequent analyses and evaluated the optimal LSTM model.

Model evaluation employed standard metrics: accuracy, precision, recall, F1-score, and Kappa coefficient (calculation details in Online Appendix 5). Higher values indicate better model performance. As shown in Table 1 , the optimal hyperparameter combination includes a learning rate of 0.01 and Tanh activation function, achieving model accuracy of 0.847 with all other metrics exceeding 0.831.

Table 1 Hyperparameter Tuning Results
Optimal Hyperparameter Combination	Optimal Model Evaluation Metrics
LSTM layers: 2	Accuracy: 0.847
LSTM nodes per layer: 32, 38	Precision: 0.831
Fully connected layers: 5	Recall: 0.831
Fully connected nodes: 38, 39, 34, 31, 27	F1-Score: 0.831
Activation: Tanh	Kappa: 0.831
Learning rate: 0.01	

4.1 Study Design

The simulation study compared LSTM performance against three traditional methods across various experimental conditions. PA, EKC, and CDF were implemented using the R package **EFAfactors** (Qin & Guo, 2025a), while LSTM utilized the trained optimal model described in Section 3.4, accessible via the **LSTMfactors** package (Qin & Guo, 2025b). The experiment included seven independent variables: factor count (1, 2, 4, 6, 8, 10), items per factor (4, 7, 10), primary loadings (high: $U(0.65, 0.80)$, medium: $U(0.50, 0.65)$, low: $U(0.35, 0.50)$), cross-loadings (high: $U(-0.20, -0.10) \cup U(0.10, 0.20)$, low: $U(-0.10, 0.10)$), interfactor correlations (0.0, 0.25, 0.50, 0.75), sample size (100, 200, 500, 1000), and retention method (LSTM, PA, EKC, CDF). Each condition was replicated 500 times, totaling $6 \times 3 \times 3 \times 2 \times 4 \times 4 \times 4 \times 500 = 3,456,000$ experimental runs.

4.2 Evaluation Metrics

We employed accuracy and bias in factor count identification as evaluation metrics, calculated as:

$$\text{acc} = \frac{\sum_{l=1}^{L} I(\hat{F}_l = F_l)}{L} \quad (5)$$

$$\text{bias} = \frac{\sum_{l=1}^{L}(\hat{F}_l - F_l)}{L} \quad (6)$$

where L represents total replications under specific conditions, F_l is the true factor count in replication l, $\hat{F}_l$ is the estimated factor count, and $I(\cdot)$ is an indicator function. Accuracy ranges from 0 to 1, with higher values indicating better performance. Bias values closer to 0 are preferable, with positive values indicating overestimation and negative values indicating underestimation.

4.3 Results

Table 2 presents accuracy results for the four factor retention methods across different data conditions.

Table 2 Accuracy of Factor Retention Across Methods and Conditions

Condition	LSTM	CDF	EKC	PA	Improvement Rate
Factor count = 10	0.865	0.313	0.319	0.280	171.09%
Interfactor correlation = 0.75	0.527	0.273	0.298	0.206	76.96%
Items per factor = 4	0.772	0.420	0.380	0.360	85.47%
Low primary loading	0.687	0.420	0.380	0.360	63.70%
High cross-loading	0.785	0.420	0.380	0.360	82.60%
Sample size = 100	0.637	0.286	0.299	0.309	106.58%

Note: Bolded values indicate best performance within each condition. Improvement rate calculated as $(acc_{LSTM} - \max(acc_{CDF}, acc_{EKC}, acc_{PA}))/\max(acc_{CDF}, acc_{EKC}, acc_{PA})$.

The results reveal that LSTM significantly outperforms traditional methods across all conditions, with particular advantages in "extreme" testing scenarios characterized by high factor counts, strong interfactor correlations, few items per factor, or small sample sizes.

Specifically, as factor count increases, traditional methods' accuracy declines substantially. When the true factor count is 10, CDF, EKC, and PA achieve only 0.313, 0.319, and 0.280 accuracy, respectively, while LSTM maintains 0.865—a 171.09% improvement. This indicates limited capacity of traditional methods for complex factor structures, whereas LSTM effectively learns and applies high-dimensional latent structures. While all methods show declining accuracy with

increasing interfactor correlations, LSTM remains superior, achieving 0.527 accuracy at the highest correlation level, significantly exceeding CDF (0.273), EKC (0.298), and PA (0.206). With few items per factor (4 items), traditional methods struggle (all below 0.42 accuracy), whereas LSTM maintains 0.772 (85.47% improvement), highlighting its strong adaptability to short-form scales. LSTM also outperforms traditional methods across loading conditions, achieving 0.687 and 0.785 accuracy under low primary loading and high cross-loading conditions, respectively, demonstrating robustness with weak structures or high noise. Finally, while all methods benefit from larger sample sizes, LSTM achieves 0.637 accuracy even with the smallest sample size, far surpassing CDF (0.286), EKC (0.299), and PA (0.309)—a 106.58% improvement over PA—showcasing its advantage in small-sample scenarios.

Table 3 presents mean bias results across conditions.

Table 3 Mean Bias in Factor Retention Across Methods and Conditions

Condition	LSTM	CDF	EKC	PA
Factor count = 4	0.524	-1.200	-1.100	-1.131
Interfactor correlation = 0.75	0.270	-2.500	-2.422	-2.600
Items per factor = 4	0.300	-2.000	-1.800	-2.100
Sample size = 100	0.270	-2.609	-2.500	-2.695

Note: Bolded values indicate best performance (closest to zero) within each condition.

Regarding bias, LSTM also demonstrates significant advantages, showing the smallest overall estimation bias and more balanced over- and underestimation. With factor counts of 2 or 4, traditional methods tend to underestimate (negative bias), whereas LSTM shows slight overestimation but with substantially smaller absolute bias values. As interfactor correlations strengthen, traditional methods exhibit increasingly negative bias (e.g., EKC bias of -2.422 at correlation = 0.75), while LSTM maintains positive bias below 0.3. Under challenging conditions (low loadings, few items), traditional methods show severe underestimation (CDF and PA bias below -2), whereas LSTM maintains bias within $±$0.5, indicating strong robustness in extreme testing conditions. Although all methods show bias reduction with larger sample sizes, LSTM' s bias remains only 0.270 even at the smallest sample size, compared to -2.609 for CDF and -2.695 for PA.

In summary, LSTM outperforms CDF, EKC, and PA on both accuracy and bias metrics, particularly under extreme conditions involving complex factor structures and poor data quality (high interfactor correlation, high cross-loadings, low primary loadings, small samples, few items per factor). As a deep learning-based approach, LSTM more effectively extracts latent structures from complex data.

5 Empirical Study

This empirical study demonstrates practical LSTM application for factor retention using real data, comparing results with PA, EKC, and CDF. The data come from a parental psychological control scale administered to 987 high school students in a city in 2022 (406 males, 41.1%; 581 females; mean age = 15.823 years, SD = 0.793). This dataset is included in the **LSTMfactors** package. The original scale by Soenens and Vansteenkiste (2010) comprises four dimensions (independence-oriented negative parental reactions, dependency-oriented positive parental reactions, low-achievement-oriented negative parental reactions, high-achievement-oriented positive parental reactions) with 20 items rated on a 5-point Likert scale (1 = strongly disagree to 5 = strongly agree). Deng et al. (2019) adapted and validated the scale for Chinese adolescents. In this study, Cronbach' s α was 0.923 for the total scale and 0.817–0.889 for the four dimensions, indicating good reliability. Descriptive statistics are presented in Table 4 .

Table 4 Descriptive Statistics ($M \pm SD$) and α Coefficients for the Parental Psychological Control Scale

Dimension	Mean ± SD	α
Independence-Negative (F1)	11.090 ± 4.315	0.817
Dependency-Positive (F2)	13.673 ± 4.530	0.845
Low Achievement-Negative (F3)	10.094 ± 4.457	0.862
High Achievement-Positive (F4)	15.484 ± 5.452	0.889
Total	50.341 ± 14.951	0.923

Confirmatory Factor Analysis (CFA) was conducted to verify the scale' s structural validity and provide a reference for the four methods' estimates. As shown in Table 5 and Figure 3

, fit indices (CFI > 0.9, RMSEA and SRMR < 0.08) and factor loadings (0.49-0.93) indicate satisfactory structural validity.

Table 5 CFA Fit Indices for the Parental Psychological Control Scale

Index	Value
CFI	0.95
TLI	0.94
RMSEA	0.06
SRMR	0.05

Subsequently, LSTM, PA, EKC, and CDF were applied to the empirical data. Example code appears in Online Appendix 6; complete code is available at https://osf.io/au9vd/ in "main/realdata/realdata.R". LSTM retained 4 factors,

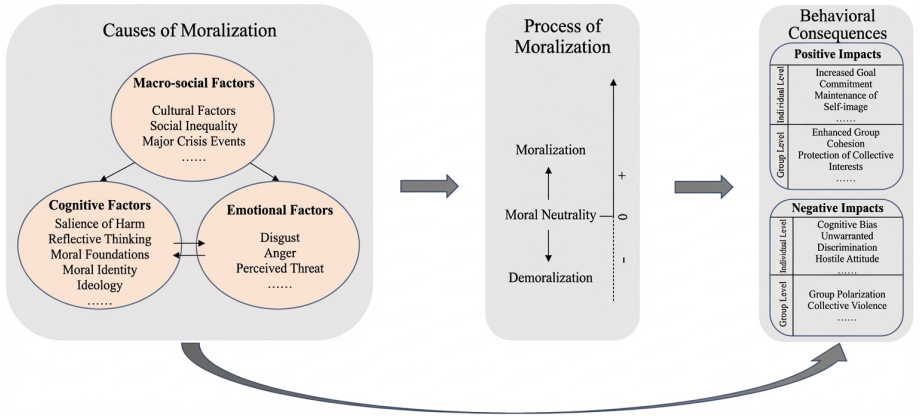

Figure 3: Figure 3

consistent with theoretical expectations and CFA results. Figure 4 [FIGURE:4] visualizes results for all four methods. LSTM (Panel A) and CDF (Panel B), as machine learning classification methods, output probability distributions for factor counts. PA (Panel C) and EKC (Panel D) determine factor count by comparing actual and reference eigenvalues. CDF overestimated factor count as 7 (maximum probability = 0.445), with probability mass distributed across 6-10 factors. PA and EKC both underestimated factor count as 3. In contrast, LSTM showed clearer decision-making, with probability concentrated on 3 and 4 factors, and highest probability for 4 factors, demonstrating greater focus and accuracy—consistent with simulation findings.

6 Discussion

EFA is an indispensable statistical technique for exploring latent structures in psychology and education. Correctly estimating the number of latent factors is fundamental to proper structural exploration and represents the most critical initial step in EFA (Zwick & Velicer, 1986). However, existing factor retention methods struggle to maintain accuracy across diverse data conditions (Auerswald & Moshagen, 2019; Goretzko & Bühner, 2020), and outdated methods (e.g., scree plots, Kaiser criterion) remain widely used, particularly in China where research on factor retention methods is scarce. This study, grounded in EFA theory, treats eigenvalues λ as sequential data with multiple time steps to train a deep neural network built with LSTM, achieving evaluation metrics exceeding 83%. Through simulation studies, the network demonstrated superior accuracy and robustness across extensive data conditions compared to traditional methods. The empirical study further demonstrated strong ecological validity. The trained LSTM has been packaged as the R package **LSTMfactors**, offering convenient implementation and practical value.

In EFA, multiple factors affect factor count estimation accuracy, including fac-

tor number, primary loading magnitude, and sample size. As factor count increases, primary loadings decrease, and sample size diminishes, most retention methods show declining accuracy—consistent with previous research (Auerswald & Moshagen, 2019; Goretzko & Bühner, 2020). This occurs because more factors increase structural complexity, lower loadings reduce factor contributions, and insufficient sample sizes obscure latent structures with random error. Additionally, cross-loading magnitude, interfactor correlation, and items per factor influence accuracy. High cross-loadings increase complexity by loading variables on multiple factors; high interfactor correlations may cause factor homogenization, merging distinct factors; and few items per factor reduce observable indicators, increasing identification difficulty. LSTM demonstrates strong robustness against these adverse conditions.

Notably, unlike traditional methods whose accuracy continuously declines with increasing factor count, LSTM exhibits a U-shaped pattern (see Table 2): accuracy decreases initially but recovers as factor count continues increasing. We hypothesize this occurs because LSTM more readily estimates larger factor numbers than traditional methods. Bias results (Table 3) show that as factor count increases, traditional methods exhibit increasingly negative bias (severe underestimation), while LSTM shows positive bias for factor counts 1–6 that decreases to negative bias at counts 8 and 10. This suggests LSTM tends to produce larger factor counts than traditional methods when true counts are small, but its estimates become closer to true values as factor count increases, resulting in reduced bias and improved accuracy. Goretzko (2025) argues that underestimation is more dangerous than overestimation, as underestimation causes factor merging and distorted loading matrices that may lead to erroneous theories, whereas overestimated factors can be manually eliminated through post-hoc variance explanation and theoretical adjustments. From this perspective, LSTM's tendency offers an advantage.

Furthermore, different theoretical mechanisms show significant differences in handling complex data structures, reflected in accuracy variations. Traditional methods like PA, EKC, and CDF rely on specific statistical calculations (e.g., reference eigenvalue computation, random data simulation), making them effective under favorable conditions (few factors, adequate sample size, low interfactor correlation, high signal-to-noise ratio) but unable to capture complex eigenvalue dependencies under challenging conditions (many factors, small samples, high interfactor correlation, low signal-to-noise ratio), leading to systematic underestimation. As a deep learning technique, LSTM more effectively learns complex associations by treating eigenvalues as sequential data, particularly for small samples and uneven loading distributions. Thus, LSTM' s advantages extend beyond higher accuracy to greater stability in extreme testing scenarios, expanding the applicability boundaries of factor retention methods.

However, this study has limitations and suggests several future directions: (1) As a pretrained model, LSTM's applicability is constrained by training data conditions—a limitation of AI-based factor retention (Goretzko, 2025). Although

simulations demonstrate robustness beyond training ranges (e.g., interfactor correlation up to 0.75 vs. training range of $U(0.00, 0.50)$), the vast space of potential data conditions cannot be fully validated. We cautiously recommend applying the trained LSTM only to data with: 1–10 factors, 3–10 items per factor, primary loadings > 0.35, cross-loadings ≤ 0.20, interfactor correlations ≤ 0.50, and sample sizes of 100–1,000. These ranges cover most research needs. For unobservable conditions like loading magnitudes, we suggest estimating factor loadings and correlations after LSTM estimation to verify applicability. (2) Our training data conditions primarily followed Goretzko and Bühner (2020) with expanded ranges, demonstrating feasibility and reliability. However, when empirical data exceed these ranges, traditional methods like PA remain viable. Given current trends in large models (DeepSeek-AI et al., 2024), future work could train a "factor retention-specific large model" following our approach to achieve "One Model to Rule Them All" (Goretzko & Bühner, 2020). (3) Our training and simulation data used standardized normal distributions. While LSTM can be seamlessly applied to normally distributed Likert-type and continuous data after standardization, its performance with nominal or skewed distributions remains unexamined and warrants future investigation.

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. *Psychological Methods*, 24(4), 468–491.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281–305.

Braeken, J., & van Assen, M. A. (2017). An empirical Kaiser criterion. *Psychological Methods*, 22(3), 450–466.

Brownlee, J. (2018). *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery.

Chen, S., Abhinav, S., Saurabh, S., & Abhinav, G. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *arXiv preprint arXiv:1707.02968*.

de Winter, J. C., & Dodou, D. (2012). Factor recovery by principal axis factoring and maximum likelihood factor analysis as a function of factor pattern and sample size. *Journal of Applied Statistics*, 39(4), 695–710.

DeepSeek-AI, A. L., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., ···, Pan, Z. (2024). DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437*.

Deng, Y., Gao, X., Xu, C., Sun, Z., Yue, Y., & Liu, X. (2019). Reliability and Validity Test of Dependency-Oriented and Achievement-Oriented Psychological Control Scale in Chinese Adolescents. *Chinese Journal of Clinical Psychology*, 27(2), 253–257.

Dinno, A. (2009). Exploring the sensitivity of Horn' s parallel analysis to the distributional form of random data. *Multivariate Behavioral Research*, 44(3), 362–388.

Fava, J. L., & Velicer, W. F. (1996). The effects of underextraction in factor and component analyses. *Educational and Psychological Measurement*, 56(6), 907–929.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goretzko, D. (2025). How many factors to retain in exploratory factor analysis? A critical overview of factor retention methods. *Psychological Methods*, Advance online publication.

Goretzko, D., & Bühner, M. (2020). One model to rule them all? Using machine learning algorithms to determine the number of factors in exploratory factor analysis. *Psychological Methods*, 25(6), 776–786.

Goretzko, D., & Bühner, M. (2022). Factor Retention Using Machine Learning With Ordinal Data. *Applied Psychological Measurement*, 46(5), 406–421.

Goretzko, D., & Ruscio, J. (2024). The comparison data forest: A new comparison data approach to determine the number of factors in exploratory factor analysis. *Behavior Research Methods*, 56(3), 1838–1851.

Heaton, J. (2008). Feedforward neural networks. In K. Smith (Ed.), *Introduction to Neural Networks with Java* (2nd ed., pp. 143–172). Heaton Research.

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2), 179–185.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.

Humphreys, L. G., & Montanelli, R. G. (1975). An investigation of the parallel analysis criterion for determining the number of common factors. *Multivariate Behavioral Research*, 10(2), 193–205.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kaiser, H. F. (1960). The application of electronic computers to factor analysis. *Educational and Psychological Measurement*, 20, 141–151.

Kalinowski, T., Ushey, K., Allaire, J. J., RStudio, Tang, Y., Eddelbuettel, D., Lewis, B., Keydana, S., Hafen, R., & Geelnard, M. (2025). *reticulate: Interface to Python*. R package version 1.34.0.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lange, S., Helfrich, K., & Ye, Q. (2022). Batch normalization preconditioning for neural network training. *Journal of Machine Learning Research*, 23(1), 3118–

3158.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.

Li, Y., Wen, Z., Hau, K.-T., Yuan, K.-H., & Peng, Y. (2020). Effects of Cross-loadings on the Number of Factors to Retain. *Structural Equation Modeling: A Multidisciplinary Journal*, 27(6), 841–863.

Lorenzo-Seva, U., Timmerman, M. E., & Kiers, H. A. (2011). The Hull method for selecting the number of common factors. *Multivariate Behavioral Research*, 46(2), 340–364.

Marčenko, V. A., & Pastur, L. A. (1967). Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1, 457–483.

Microsoft. (2025). *ONNX Runtime*. Retrieved from https://onnxruntime.ai/.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 807–814).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ⋯, Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (pp. Article 721). Curran Associates Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ⋯, Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Peres-Neto, P. R., Jackson, D. A., & Somers, K. M. (2005). How many principal components? Stopping rules for determining the number of non-trivial axes revisited. *Computational Statistics & Data Analysis*, 49(4), 974–997.

pyper. (2025). *pyper: Concurrent Python made simple*. Retrieved from https://github.com/pyper-dev/pyper.

Python Software Foundation. (2025). *Python 3.13.3*. Retrieved from https://www.python.org/downloads/release/python-3133/.

PyTorch. (2025). *PyTorch*. Retrieved from https://pytorch.org/.

Qin, H., & Guo, L. (2024). Using machine learning to improve Q-matrix validation. *Behavior Research Methods*, 56(3), 1916–1935.

Qin, H., & Guo, L. (2025a). *EFAfactors: Determining the Number of Factors in Exploratory Factor Analysis*. R package version 1.2.1.

Qin, H., & Guo, L. (2025b). *LSTMfactors: Determining the Number of Factors in Exploratory Factor Analysis by LSTM*. R package version 1.0.0.

R Core Team. (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org.

Ruscio, J., & Kaczetow, W. (2008). Simulating Multivariate Nonnormal Data Using an Iterative Algorithm. *Multivariate Behavioral Research*, 43(3), 355–381.

Ruscio, J., & Roche, B. (2012). Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. *Psychological Assessment*, 24(2), 282–292.

scikit-learn. (2025). *scikit-learn: Machine Learning in Python*. Retrieved from https://scikit-learn.org/stable/.

Soenens, B., & Vansteenkiste, M. (2010). A theoretical upgrade of the concept of parental psychological control: Proposing new insights on the basis of self-determination theory. *Developmental Review*, 30(1), 74–99.

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika*, 41(3), 321–327.

Wood, J. M., Tataryn, D. J., & Gorsuch, R. L. (1996). Effects of under- and overextraction on principal axis factor analysis with varimax rotation. *Psychological Methods*, 1(4), 354–365.

Zwick, W. R., & Velicer, W. F. (1986). Comparison of five rules for determining the number of components to retain. *Psychological Bulletin*, 99(3), 432–442.

Online Appendices

1 Parallel Analysis

PA factor retention proceeds as follows:

1. Compute eigenvalues from the empirical response data matrix $\mathbf{X}_{N\times J}$ using principal component analysis or factor analysis, yielding the true eigenvalue vector $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_J\}$ $(1 \leq j \leq J)$, where N is sample size and J equals the number of observed variables and eigenvalue sequence length.
2. Generate M (this study uses $M = 100$, consistent with Auerswald & Moshagen, 2019) random simulated response matrices $\mathbf{X}^{\text{ref},m}_{N\times J}$ $(m = 1, 2, \ldots, M)$ matching the dimensions of the empirical data. For each simulation:
    a) For each observed variable j, treat column $\mathbf{X}_{j|N\times 1}$ from $\mathbf{X}_{N\times J}$ as a sampling pool. Perform bootstrap sampling with replacement to randomly draw N values, forming simulated vector $\mathbf{X}^{\text{ref},m}_{j|N\times 1}$.
    b) Iterate across all J variables to obtain J vectors $\mathbf{X}^{\text{ref},m}_{j|N\times 1}$, then column-bind them to form simulated data matrix $\mathbf{X}^{\text{ref},m}_{N\times J}$.

c) Compute eigenvalues from the correlation matrix of $\mathbf{X}_{N\times J}^{\text{ref},m}$ using principal component analysis, yielding reference eigenvalue vector $\lambda_{1\times J}^{\text{ref},m}$.

3. Row-bind the M reference eigenvalue vectors $\lambda_{1\times J}^{\text{ref},m}$ into an $M \times J$ matrix. Compute column-wise means or 95th percentiles as reference eigenvalues λ_j^{ref}.

4. Compare true eigenvalues λ_j with reference eigenvalues λ_j^{ref}. The retained factor count F is:

$$F = \sum_{j=1}^{J} I(\lambda_j > \lambda_j^{\text{ref}}) \quad (1)$$

2 Empirical Kaiser Criterion

Under the null model, eigenvalues from random correlation matrices asymptotically follow the Marčenko-Pastur distribution, whose upper bound serves as the first reference eigenvalue:

$$\lambda_1^{\text{ref}} = \left(1 + \sqrt{\frac{J}{N}}\right)^2 \quad (2)$$

where N is sample size and J is the number of observed variables. Subsequent reference eigenvalues are calculated as:

$$\lambda_j^{\text{ref}} = \max\left[\frac{J - \sum_{k=1}^{j-1}\lambda_k}{J-j+1}\left(1+\sqrt{\frac{J}{N}}\right)^2, 1\right] \quad (3)$$

EKC follows the Kaiser criterion by retaining the maximum of the calculated reference value and 1. The factor count formula matches equation (1).

3 Comparison Data Forest Method

CDF uses the same simulation approach as CD (see Ruscio & Kaczetow, 2008). "Typical" values refer to those used by Auerswald and Moshagen (2019) and Goretzko and Bühner (2020), which share identical simulation parameters.

CDF procedure:

1. Compute correlation matrix from empirical data $\mathbf{X}_{N\times J}$. Generate simulated response matrix $\mathbf{X}_{P\times J}^{f}$ with f factors ($f = 1, 2, \ldots, F$; typically $P = 10,000$ subjects), where F is the maximum factor count under consideration (this study uses $F = 10$):

   a) Use empirical correlation matrix as the target.
   b) Randomly generate common factor scores $\mathbf{S}_{P\times f} \sim N(0,1)$ and residual scores $\mathbf{U}_{P\times J} \sim N(0,1)$.

c) Initialize $_a =$, $\text{RMSR}_a = \infty$, and counter $t = 0$.
d) Extract f factors from $_a$ via principal axis factoring to obtain loading matrix $^{\text{shared}}_{J\times f}$.
e) Compute unique factor matrix $^{\text{unique}}_{J\times 1}$ where element j is $\sqrt{1-\sum_{k=1}^{f}(\Lambda^{\text{shared}}_{j,k})^2}$.
f) Simulate response data for subject p on item j: $X^f_{p,j} = \mathbf{S}_{p,1:f}(\,^{\text{shared}}_{j,1:f})^T + U_{p,j}\Lambda^{\text{unique}}_j$.
g) Compute correlation matrix $_b$ of $\mathbf{X}^f_{P\times J}$ and residual matrix $_{\text{Res}} = - _b$. Calculate RMSR from lower-triangular elements of $_{\text{Res}}$.
h) If $\text{RMSR} < \text{RMSR}_a$, update $_a = _a + r \times _{\text{Res}}$ (learning rate $r = 1$), $\text{RMSR}_a = \text{RMSR}$, and reset $t = 0$. Otherwise, $_a = _a + 0.5\times t\times r\times _{\text{Res}}$ and increment $t = t + 1$.
i) Repeat steps d-h until $t > t_{\max}$ (typically $t_{\max} = 5$). The final simulated data matrix $\mathbf{X}^f_{P\times J}$ will have a correlation matrix approximating with f latent factors.

2. Use bootstrap sampling to draw K simulated datasets (typically $K = 5,000$) matching the original sample size N, denoted $\mathbf{X}^{f,k}_{N\times J}$ $(k = 1, 2, \ldots, K)$.

3. Compute 181 training features for each $\mathbf{X}^{f,k}_{N\times J}$ (see Goretzko & Bühner, 2020) to form training dataset TD.

4. Train a random forest for multi-class classification (factor counts 1-10) using factor count f as labels. Following Goretzko and Ruscio (2024), use 500 trees with maximum depth $\sqrt{181} \approx 13$.

5. Apply the trained CDF model to predict factor count for empirical data.

4 LSTM Principles and Architecture

A typical LSTM memory cell is illustrated in Figure A1. The figure shows input features $\mathbf{X}_j$ at time step j (in this study, $\mathbf{X}_j = \{\lambda_j, \lambda_j - \lambda^{\text{ref}}_j\}$), candidate memory cell $\tilde{C}_j$, previous and current memory cells C_{j-1} and C_j, previous and current hidden states H_{j-1} and H_j, Sigmoid activation σ, tanh activation, element-wise product $\otimes$, element-wise sum $\oplus$, forget gate FG, input gate IG, output gate OG, and gate outputs F_j, I_j, and O_j at time step j.

Gate and candidate cell computations:

$$\mathbf{F}_j = \sigma(\mathbf{X}_j\mathbf{W}_{x,FG} + \mathbf{H}_{j-1}\mathbf{W}_{h,FG} + \mathbf{b}_{FG}) \quad (9)$$

$$\mathbf{I}_j = \sigma(\mathbf{X}_j\mathbf{W}_{x,IG} + \mathbf{H}_{j-1}\mathbf{W}_{h,IG} + \mathbf{b}_{IG}) \quad (10)$$

$$\mathbf{O}_j = \sigma(\mathbf{X}_j\mathbf{W}_{x,OG} + \mathbf{H}_{j-1}\mathbf{W}_{h,OG} + \mathbf{b}_{OG}) \quad (11)$$

$$\tilde{\mathbf{C}}_j = \tanh(\mathbf{X}_j\mathbf{W}_{x,\tilde{C}} + \mathbf{H}_{j-1}\mathbf{W}_{h,\tilde{C}} + \mathbf{b}_{\tilde{C}}) \quad (12)$$

The forget gate discards old information, the input gate adds new information, and together they update the memory cell: $\mathbf{C}_j = \mathbf{F}_j \otimes \mathbf{C}_{j-1} + \mathbf{I}_j \otimes \tilde{\mathbf{C}}_j$. The output gate determines the output: $\mathbf{H}_j = \mathbf{O}_j \otimes \tanh(\mathbf{C}_j)$. Multiple memory units in an LSTM layer share weights but learn different aspects of the input sequence, enhancing representation capacity.

Fully connected and batch normalization layers are simpler. A fully connected layer computes $\mathbf{Y} = AF(\mathbf{XW}+\mathbf{b})$, where AF is any activation function. Batch normalization standardizes mini-batch inputs during training to accelerate convergence and improve generalization. For mini-batch $\mathbf{X} = \{X_1, X_2, \ldots, X_M\}$, it computes mean μ and variance σ^2, normalizes each sample as $\hat{X}_m = (X_m - \mu)/\sqrt{\sigma^2+\epsilon}$, then applies learnable scale γ and shift β: $Y_m = \gamma\hat{X}_m + \beta$. This also provides slight regularization to prevent overfitting.

Figure A2 shows our LSTM + fully connected + batch normalization architecture with two LSTM layers processing sequence length J, one fully connected layer, two batch normalization layers, and Softmax output for multi-class classification.

5 Confusion Matrix and Evaluation Metrics for Multi-class Classification

For multi-class models, macro-averaged accuracy, precision, recall, F1-score, and Kappa can be computed from confusion matrices using scikit-learn. For a three-class task, the confusion matrix is:

Table A1 Example Confusion Matrix for Three-Class Task

Actual	Predicted	C1	C2	C3
C1		a	b	c
C2		d	e	f
C3		g	h	i

Metric calculations:

$$\text{Accuracy} = \frac{a+e+i}{a+b+c+d+e+f+g+h+i} \quad (15)$$

$$\text{Precision} = \frac{1}{3}\left(\frac{a}{a+d+g} + \frac{e}{b+e+h} + \frac{i}{c+f+i}\right) \quad (16)$$

$$\text{Recall} = \frac{1}{3}\left(\frac{a}{a+b+c} + \frac{e}{d+e+f} + \frac{i}{g+h+i}\right) \quad (17)$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (18)$$

$$\text{Kappa} = \frac{P_0 - P_e}{1 - P_e} \quad (19)$$

where $P_0 = \text{Accuracy}$ and $P_e = \sum_{x=1}^{3} \text{row}_x \times \text{col}_x$.

6 Empirical Study Example Code

PA, EKC, and CDF are implemented in the R package **EFAfactors**:

```
R> set.seed(112)         ## Fix random seed
R> library(EFAfactors)       ## Load EFAfactors
EFAfactors R Package (version 1.2.1; 2025-02-15)
R> PA.obj <- PA(response = response, plot = TRUE)         ## Run PA
The number of factors suggested by PA    (quant=0.95)    is 3 .

R> EKC.obj <- EKC(response = response, plot = TRUE)     ## Run EKC
The number of factors suggested by EKC is 3 .

R> CDF.obj <- CDF(response = response, nfact.max = 10, N.pop = 10000,
                  N.Samples = 5000, mtry = "sqrt")     ## Run CDF
CDF is simulating data: nfact=10/10 - N_{rep}=5000/5000
The number of factors suggested by CDF is 7 .
```

LSTM is called via the **LSTMfactors** package:

```
R> library(LSTMfactors) ## Load LSTMfactors
LSTMfactors R Package (version 1.0.0; 2025-06-25)
R> LSTM.obj <- LSTM(response = response, plot = TRUE)     ## Run LSTM
The number of factors suggested by    LSTM    is 4 .
```