**Frontiers in Computing and Intelligent Systems**
ISSN: 2832-6024 | Vol. 9, No. 1, 2024

# AutoEncoder-Based Data Completion Model for Theft Cases of Items Inside Cars

**Enqi Cao, Fanliang Bu * and Zhuxuan Han**

School of Information Technology and Cyber Security, People's Public Security University of China, Beijing, China
*** Corresponding author:** Fanliang Bu (Email: 2021211461@stu.ppsuc.edu.cn)

**Abstract:** Accurate and reliable prediction results depend on high-quality data, and missing data is one of the key factors affecting data quality. This paper proposes a data completion model for car interior theft cases based on an AutoEncoder, which adopts an AutoEncoder architecture combining Re-parameterized Convolutional Neural Network and Recurrent Neural Network. Experimental results show that RepConv-RNN-AE can effectively complete the missing values in car interior theft case data and demonstrates superiority in completing missing data in car interior theft cases.

**Keywords:** AutoEncoder; Theft Cases; Data Completion.

## 1. Introduction

Accurate and reliable prediction results depend on high-quality data, with missing data being a key factor affecting data quality [1]. In most data collected across various social domains, there is often some degree of missing data, which is a common and unavoidable issue [2]. Missing data directly leads to reduced data completeness and decreased correlation among different attributes within the data, thereby degrading data quality and subsequently impacting data analysis and prediction tasks [3].

In recent years, although data completion techniques have been applied in crime research, most methods for completing missing data in crime case data still rely heavily on traditional machine learning algorithms. With the rapid growth of data volumes in the public safety field, traditional machine learning-based data completion methods have shown significant limitations in dealing with complex high-dimensional crime datasets, resulting in poor data completion performance and low efficiency.

Re-parameterized Convolutional Neural Network (Rep-CNN) is an improved convolutional neural network architecture with higher performance and lower complexity [4]. This convolution operation enhances the feature extraction capability of time series data for car interior theft cases, efficiently extracting local spatial features from the data. Although car interior theft case data is essentially one-dimensional, the location information can still be transformed into multidimensional features through one-hot encoding. This allows Rep-CNN to perform convolution operations on these multidimensional features, capturing the spatial distribution patterns of car interior theft cases and identifying high-incidence areas for such crimes. Recurrent Neural Network (RNN) excels in processing time series data [5], capturing the temporal dependencies within car interior theft case data, and identifying high-incidence periods for such crimes. AutoEncoder (AE) effectively handles missing field values in car interior theft case data by compressing the input data into a low-dimensional representation and then reconstructing the original data from this low-dimensional representation [6]. By combining the spatial feature extraction capabilities of Rep-CNN, the temporal dependency modeling capabilities of RNN, and the data reconstruction ability of AE, RepConv-RNN-AE effectively addresses the issue of missing data in car interior theft cases.

## 2. RepConv-RNN-AE Data Completion Model

RepConv-RNN-AE consists of an encoder and a decoder, which can complete the missing data in existing car interior theft case data containing missing field values, resulting in complete car interior theft case data. Suppose the input data is $X = \{x_i\}_{i=1}^{N}$. Here, $x_i = [x_{i,1}, x_{i,2}, \dots, x_{i,M}]$ represents the $i$-th car interior theft case record in the input data $X$, which contains $M$ field values. $N$ represents the number of car interior theft case records in $X$. For a certain field value $x_{i,j}$ in a car interior theft case record $x_i$, there may be missing data, where $j \in \{1,2,\dots,M\}$.

### 2.1. Encoder Part of RepConv-RNN-AE

The encoder of RepConv-RNN-AE consists of three layers of RepConvBlock and four layers of RNN. During the encoding process, three layers of RepConvBlock are first used to extract the local features of car interior theft case data to capture the spatial distribution characteristics of car interior theft cases. The output features of each RepConvBlock layer are used as the input for the next RepConvBlock layer, as follows:

$$H_{\text{rcb}}^{l} = \text{RepConvBlock}^{l}\left(H_{\text{rcb}}^{(l-1)}\right) \tag{1}$$

Where $H_{\text{rcb}}^{0} = X$, representing the input of the initial first RepConvBlock layer. $H_{\text{rcb}}^{(l-1)}$ represents the output of the $(l-1)$-th RepConvBlock layer.

For the $l$-th RepConvBlock layer, different network structures are used during training and inference to improve the efficiency of the network. Specifically, the calculation process during the training phase is as follows:

The $3 \times 3$ convolution branch extracts local features, and the calculation formula is as follows:

$$h_{\text{conv1}}^{l}(t,k) = \sigma\left(\sum_{c=0}^{C-1} \sum_{u=-1}^{1} W_{\text{conv1},k}^{c}(u) \cdot H_{\text{rcb}}^{(l-1,c)}(t+u) + b_{\text{conv1},k}\right) \tag{2}$$

Where $h^l_{\text{conv1}}(t,k)$ represents the output feature of the $3\times3$ convolution branch at position $t$ in the input sequence and feature channel $k$. $H^{(l-1,c)}_{\text{rcb}}(t+u)$ represents the output feature of the $(l-1)$-th RepConvBlock layer at channel $c$ and position $t+u$. $W^c_{\text{conv1},k}(u)$ represents the weight of the $3\times3$ convolution kernel. $b_{\text{conv1},k}$ represents the bias of the $3\times3$ convolution kernel. $C$ represents the number of input channels. $u$ represents the offset of the convolution kernel. $K$ represents the number of output feature channels. $\sigma$ represents the activation function (e.g., ReLU activation function). $t$ represents the time step position in the input sequence, with each car interior theft case record corresponding to a time step.

The $1\times1$ convolution branch reduces the number of parameters and fuses features, and the calculation formula is as follows:

$$h^l_{\text{conv2}}(t,k)=\sigma\left(\sum_{c=0}^{C-1} W^c_{\text{conv2},k}(u)\cdot H^{(l-1,c)}_{\text{rcb}}(t)+b_{\text{conv2},k}\right) \quad (3)$$

Where $h^l_{\text{conv2}}(t,k)$ represents the output feature of the $1\times1$ convolution branch at position $t$ in the input sequence and feature channel $k$. $H^{(l-1,c)}_{\text{rcb}}(t)$ represents the output feature of the $(l-1)$-th RepConvBlock layer at channel $c$ and position $t$. $W^c_{\text{conv2},k}(u)$ represents the weight of the $1\times1$ convolution kernel. $b_{\text{conv2},k}$ represents the bias of the $1\times1$ convolution kernel. $C$ represents the number of input channels. $K$ represents the number of output feature channels. $\sigma$ represents the activation function (e.g., ReLU activation function). $t$ represents the time step position in the input sequence, with each car interior theft case record corresponding to a time step.

The identity mapping branch directly transmits information, and the calculation formula is as follows:

$$h^l_{\text{conv3}}(t,k)=H^{(l-1),k}_{\text{rcb}}(t) \quad (4)$$

Where $h^l_{\text{conv3}}(t,k)$ represents the output feature of the identity mapping branch at position $t$ in the input sequence and feature channel $k$. $H^{(l-1),k}_{\text{rcb}}(t)$ represents the output feature of the $(l-1)$-th RepConvBlock layer at channel $c$ and position $t$. $t$ represents the time step position in the input sequence, with each car interior theft case record corresponding to a time step.

The outputs of each branch are finally accumulated, and the specific calculation formula is as follows:

$$H^l_{\text{rcb}}(t,k)=h^l_{\text{conv1}}(t,k)+h^l_{\text{conv2}}(t,k)+h^l_{\text{conv3}} \quad (5)$$

Finally, the above formula is applied to all time steps and feature channels of the entire input sequence, resulting in the overall output of the $l$-th RepConvBlock layer during the training phase, as follows:

$$H^l_{\text{rcb}}=\left\{H^l_{\text{rcb}}(t,k)\mid t=1,\dots,T;k=1,\dots,K\right\} \quad (6)$$

Where $T$ represents the total number of time steps, i.e., the number of data points in the time series, with each car interior theft case record corresponding to a time step. $K$ represents the number of output feature channels.

The specific calculation process during the inference phase is as follows:

The convolution kernels are merged, and the calculation formula is as follows:

$$W^c_{\text{conv,new},k}(u)=W^c_{\text{conv1},k}(u)+\text{Pad}\left(W^c_{\text{conv2},k}(u)\right)+I_k \quad (7)$$

Where $W^c_{\text{conv,new},k}(u)$ represents the merged convolution kernel weight. $W^c_{\text{conv1},k}(u)$ represents the weight of the $3\times3$ convolution kernel. $\text{Pad}\left(W^c_{\text{conv2},k}(u)\right)$ represents the zero-padding of the $1\times1$ convolution kernel weight $W^c_{\text{conv2},k}(u)$. $I_k$ represents the convolution kernel of the identity mapping (identity matrix).

The bias terms are merged, and the calculation formula is as follows:

$$b_{\text{conv1,new},k}=b_{\text{conv1},k}+b_{\text{conv2},k} \quad (8)$$

Where $b_{\text{conv1,new},k}$ represents the merged bias term. $b_{\text{conv1},k}$ represents the bias term of the $3\times3$ convolution kernel. $b_{\text{conv2},k}$ represents the bias term of the $1\times1$ convolution kernel.

The calculation formula for the merged single convolution operation is as follows:

$$H^l_{\text{rcb}}(t,k)=\sigma\left(\sum_{c=0}^{C-1}\sum_{u=-1}^{1} W^c_{\text{conv,new},k}(u)\cdot H^{(l-1),c}_{\text{rcb}}(t+u)+b_{\text{conv1,new},k}\right) \quad (9)$$

Finally, the above formula is applied to all time steps and feature channels of the entire input sequence, resulting in the overall output of the $l$-th RepConvBlock layer during the inference phase, as follows:

$$H^l_{\text{rcb}}=\left\{H^l_{\text{rcb}}(t,k)\mid t=1,\dots,T;k=1,\dots,K\right\} \quad (10)$$

Where $T$ represents the total number of time steps, i.e., the number of data points in the time series, with each car interior theft case record corresponding to a time step. $K$ represents the number of output feature channels.

Next, the encoder part employs four layers of RNN to capture the temporal patterns in the car interior theft case data. For the $l$-th layer of RNN in the encoder, the specific calculation process is as follows:

$$H^l_{\text{rnn}},h^l_t=\text{RNN}^l\left(H^{(l-1)}_{\text{rnn}},h^l_{t-1}\right) \quad (11)$$

Where $H^0_{\text{rnn}}=H^3_{\text{rcb}}$, i.e., the output of the last RepConvBlock layer is the input of the initial RNN layer in the encoder. $H^l_{\text{rnn}}$ represents the output of the $l$-th layer RNN over the entire sequence. $h^l_t$ represents the hidden state at the current time step $t$ in the $l$-th layer, which is computed based on the hidden state $h^l_{t-1}$ of the previous time step and the current input feature $H^{(l-1)}_{\text{rnn}}$. $\text{RNN}^l$ represents the RNN unit calculation.

The high-level temporal feature representation $H_{\text{rnn}}^{4}$ generated by the last RNN layer is the final output of the encoder.

## 2.2. Decoder Part of RepConv-RNN-AE

In the decoder part of RepConv-RNN-AE, four layers of RNN with the same structure as the encoder are first used for decoding to ensure the temporal continuity and reconstruction effect of the car interior theft case data. For the $l$-th layer of RNN in the decoder, the specific calculation process is as follows:

$$H_{\text{decoded}}^{l}, h_t^l = \text{RNN}^l\left(H_{\text{decoded}}^{(l-1)}, h_{t-1}^l\right) \tag{12}$$

Where $H_{\text{decoded}}^{0} = H_{\text{encoded}}$, i.e., the output of the encoder is the input of the initial RNN layer in the decoder. $H_{\text{decoded}}^{l}$ represents the output of the $l$-th layer RNN over the entire sequence. $h_t^l$ represents the hidden state at the current time step $t$ in the $l$ -th layer, which is computed based on the hidden state $h_{t-1}^l$ of the previous time step and the current input feature $H_{\text{decoded}}^{(l-1)}$.

Next, the decoder part continues to use two upsampling layers and two RepConvBlock layers to further process the output of the four layers of RNN in the decoder. The upsampling (Upsample) layers are used to gradually restore the temporal dimension of the feature maps, and the RepConvBlock layers are used to restore and enhance the feature maps. For the $l$-th upsampling and RepConvBlock, the specific calculation process is as follows:

$$H_{\text{upsample-rcb}}^{l} = \text{RepConvBlock}^l\left(\text{Upsample}^l\left(H_{\text{upsample-rcb}}^{(l-1)}\right)\right) \tag{13}$$

Where $H_{\text{upsample-rcb}}^{0} = H_{\text{decoded}}^{4}$, i.e., the output of the last RNN layer in the decoder is the input of the first upsampling layer. Upsample represents the upsampling operation.

Finally, a 1D-CNN layer is used to convert the feature maps to the size of the original input, completing the reconstruction of the car interior theft case data and achieving the completion of missing field values. The specific calculation process of the final 1D-CNN layer is as follows:

$$H_{\text{conv}} = \text{Conv1D}\left(H_{\text{upsample-rcb}}^{2}\right) \tag{14}$$

Where $H_{\text{upsample-rcb}}^{2}$ represents the output after two upsampling and RepConvBlock layers. $H_{\text{conv}}$ represents the output after a layer of 1D convolution, which is the reconstructed car interior theft case data $\tilde{X}$. $\tilde{X}$ represents the complete car interior theft case data without missing field values, completed by RepConv-RNN-AE.

# 3. Experimental Validation and Result Analysis

## 3.1. Data Preprocessing

To accurately evaluate the model, 5119 complete car interior theft case records without missing field values from a city in China, spanning from January 1, 2018, to December 31, 2023, after anonymization and desensitization processing, were selected as the experimental dataset. Specifically, 3157 theft case records from January 1, 2018, to December 31, 2021, were used for the training set, and 1962 theft case records from January 1, 2022, to December 31, 2023, were used for the test set.

In the experiment, four key feature fields were selected for data completion, specifically: "Time of Crime", "Location of Crime", and "Police Substation". For the "Time of Crime" feature field, based on people's daily activities and routines, the year, month, and date were extracted from the timestamp and converted into discrete categorical features. Additionally, the 24-hour time format was divided into six time periods per day: early morning (00:00-04:00), morning (04:00-08:00), forenoon (08:00-12:00), afternoon (12:00-16:00), evening (16:00-20:00), and night (20:00-00:00). After converting the "Time of Crime" feature field, all feature fields were one-hot encoded. The number of categories and examples for each selected feature field in the theft case data are shown in Table 1.

**Table 1.** Theft Case Data Feature Fields, Number of Categories, and Examples

| Field Name | Number of Categories | Specific Examples |
|---|---|---|
| Time of Crime | 9 | Year, Month, Date, Early Morning, Morning, etc. |
| Location of Crime | 19 | Public Parking Lot, Urban Residential Area, etc. |
| Police Substation | 6 | Substation A, Substation B, Substation C, etc. |

## 3.2. Experimental Design

To evaluate the effectiveness and superiority of the model, the results of data completion for car interior theft cases using RepConv-RNN-AE were compared with those using other data completion models. This validates the superiority of RepConv-RNN-AE in completing car interior theft case data. The comparison models include Mean Imputation (MI) [7], K-Nearest Neighbors (KNN) [8], AutoEncoder (AE) [9], and Generative Adversarial Imputation Nets (GAIN) [10].

## 3.3. Experimental Parameter Settings

In the experiment, RepConv-RNN-AE uses the cross-entropy loss function. Specifically, the loss function consists of two parts: missing data completion loss and overall data loss. The missing data completion loss calculates the cross-entropy loss only for the missing data that needs to be completed and assigns it a higher weight (set to 2). The overall data loss calculates the cross-entropy loss over the entire output data. The final loss function is the weighted sum of these two parts.

In the experiment, the data missing rate for the theft case dataset is set to 30% to simulate the missing data situation in real-world theft cases. A continuous sequence of 32 car interior theft case records is used as a sample, and data completion is performed for the missing field values in the sample records.

### 3.4. Experimental Evaluation Metrics

In the experiment, accuracy, precision, recall, and F1 score are selected as the evaluation metrics for data completion of car interior theft cases. To ensure comprehensiveness and fairness of the evaluation, the macro average method is used to calculate precision, recall, and F1 score.

### 3.5. Experimental Results and Analysis

Table 2 shows the experimental results of data completion for theft cases using RepConv-RNN-AE and other data completion models.

**Table 2.** Comparison of Data Completion Results for Theft Cases Using RepConv-RNN-AE and Other Models

| Model Name | Evaluation Metrics | | | |
|---|---|---|---|---|
| | Accuracy (%) | Precision (%) | Precision (%) | F1 score (%) |
| MI | 23.28 | 19.94 | 17.55 | 18.67 |
| KNN | 45.52 | 41.47 | 39.51 | 40.47 |
| GAIN | 51.13 | 45.04 | 45.16 | 45.10 |
| AE | 52.81 | 47.72 | 44.28 | 45.94 |
| RepConv-RNN-AE | 60.34 | 56.51 | 56.14 | 56.32 |

The experimental results in Table 2 indicate that in terms of data completion for car interior theft cases, the MI model performs the worst, with all evaluation metrics significantly lower than those of the other models, demonstrating its inability to effectively complete the missing data in car interior theft cases. The KNN model shows significant improvement in data completion performance for car interior theft cases compared to the MI model, but its evaluation metrics are still not high, highlighting the limitations of distance-based imputation methods (such as KNN) under high missing rates. The GAIN model shows improvement in all evaluation metrics for data completion of car interior theft cases compared to the MI and KNN models, particularly in precision and recall. This indicates that the GAIN model has certain advantages in handling missing data in car interior theft cases, but its completion effect is still inferior to the AE and RepConv-RNN-AE models. The AE model performs relatively well, especially in precision and accuracy, but its recall is relatively low, indicating that the AE model has a certain degree of omission during data completion. RepConv-RNN-AE performs the best among the five models, with all evaluation metrics significantly higher than those of the other models, demonstrating its superiority in completing missing data in car interior theft cases.

## 4. Summary

This paper combines a Re-parameterized Convolutional Neural Network (Rep-CNN), a Recurrent Neural Network (RNN), and an AutoEncoder (AE) architecture to propose a data completion model for car interior theft cases. By integrating the spatial feature extraction capabilities of Rep-CNN, the temporal dependency modeling capabilities of RNN, and the data reconstruction ability of AE, this model effectively addresses the issue of missing data in car interior theft cases. Experimental results demonstrate that the proposed RepConv-RNN-AE can effectively solve the missing data problem in car interior theft cases and has certain advantages in completing missing data in such cases.

## References


[1] Taleb I, Serhani M A, Dssouli R. Big Data Quality: A Survey. 2018 IEEE International Congress on Big Data (BigData Congress). IEEE, 2018: 166-173.

[2] Dong Y, Peng C Y J. Principled missing data methods for researchers. SpringerPlus, 2013, 2: 222.

[3] Laranjeiro N, Soydemir S N, Bernardino J. A Survey on Data Quality: Classifying Poor Data. 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC). IEEE, 2015: 179-188.

[4] Ding X, Zhang X, Ma N, et al. RepVGG: Making VGG-Style ConvNets Great Again. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2021: 13733-13742.

[5] Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 2020, 404: 132306.

[6] Chen S, Guo W. Auto-Encoders in Deep Learning—A Review with New Perspectives. Mathematics, 2023, 11(8): 1777.

[7] Zhang Z. Missing data imputation: focusing on single imputation. Annals of Translational Medicine, 2016, 4(1): 9.

[8] Murti D M P, Wibawa A P, Akbar M I. K-Nearest Neighbor (K-NN) based Missing Data Imputation. 2019 5th International Conference on Science in Information Technology (ICSITech). IEEE, 2019: 83-88.

[9] Chen S, Guo W. Auto-Encoders in Deep Learning—A Review with New Perspectives. Mathematics, 2023, 11(8): 1777.

[10] Yoon J, Jordon J, Schaar M. GAIN: Missing Data Imputation using Generative Adversarial Nets. Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018: 5689-5698.