RESEARCH ON SIGNATURE RECOGNITION METHOD BASED ON DEEP LEARNING TECHNIQUE IN PYTHON LANGUAGE

Phuc Hau Nguyen
•
Faculty of Information Technology, Electric Power University, Ha Noi city, Viet Nam
haunp@epu.edu.vn

Abstract

Expression recognition often relies on a CNN for extraction of important features from image data before that image data can be used by the RNN. Similarly, signature recognition is an important problem in the field of security and identity authentication. With the development of deep learning, models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been widely applied to automate the process of classifying real and fake signatures. This paper presents a comprehensive study on the construction of a signature recognition system using deep learning with Python. We focus on the preprocessing of signature images, the construction of a CNN model architecture combined with RNN to exploit image features and stroke sequences, and the performance comparison between models. Through experiments on diverse datasets, the system achieves an accuracy of up to 96% in distinguishing "real signatures" and "forged signatures". In addition, the paper also analyzes the influencing factors, existing limitations, and proposes directions for future research.

Keywords: Signature recognition, Deep learning, Convolutional neural network, Recurrent neural network, Python, Data preprocessing, Identity authentication.

I. Introduction

Both vendors and customers appreciate the convenience, effectiveness, and low cost of conducting business online [1]. Traditionally, signature recognition based on manual analysis or using basic image processing methods has been limited due to the diversity in writing styles and background noise of signature images.

With the breakthrough of artificial intelligence and deep learning, CNN and RNN models have opened up a way to solve the problem of signature recognition more automatically and accurately. Deep learning allows the system to automatically extract complex features from raw data, reducing the dependence on manual features. This not only helps improve the accuracy but also increases the generalization ability of the system under variable data conditions. This paper aims to:

- Analyze and evaluate the preprocessing methods of signature image data.
- Design the architecture of deep learning model (CNN combined with RNN) for signature recognition.

- Conduct experiments to evaluate the performance of the model.
- Propose further research directions to improve recognition ability and practical application.

The results obtained from the research will contribute to the development of more secure electronic authentication systems and online transactions.

II. Overview of related research

I. Traditional Signature Recognition Methods

Before the explosion of deep learning, signature recognition systems mainly relied on traditional image processing techniques such as:

- Manual feature extraction: Use Sobel, Canny filters to detect edges, then calculate features such as line thickness, tilt, and signature shape.
- Geometric analysis: Identify keypoints and compare the distance and angle between them.
- Statistical techniques: Use statistical models to classify based on the distribution of features.

Although these methods have achieved some promising results under controlled conditions, they are often not robust enough when faced with large variations in writing style, background noise, and signature complexity [2].

II. Deep Learning in Image Recognition

In neural networks, convolutional neural network (CNN) model is one of the models for image recognition and classification. CNN classifies images by taking an input image, processing and classifying it into certain categories (For example: Signature, Human face, Dog, Cat, Tiger, ...). Deep learning has revolutionized the field of image recognition. CNN models were developed to automatically extract features from image data. the effect of pre-processing on the success of learning CNN structure is studied [3]. Some typical architectures include:

- LeNet-5: First applied to handwritten digit recognition.
- AlexNet and VGGNet: Improved image processing with multiple convolutional layers.
- ResNet: Introduced the “skip connection” technique to solve the vanishing gradient problem in very deep networks.

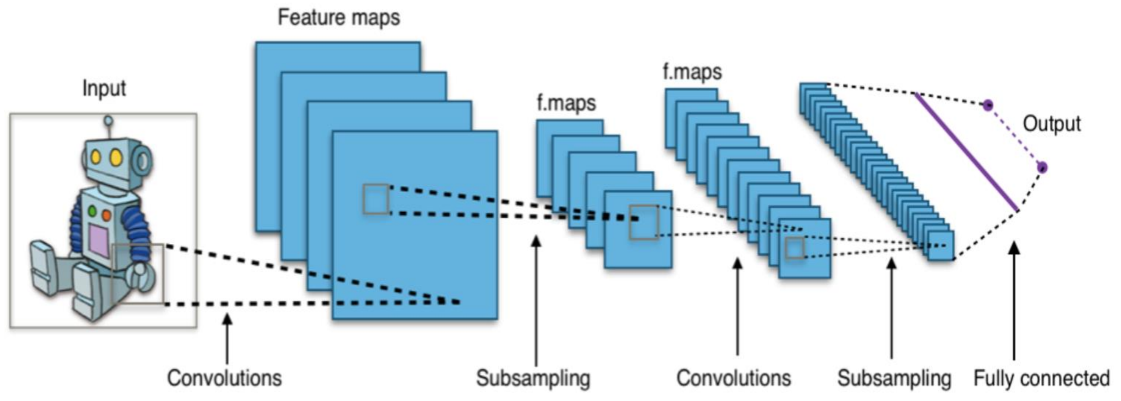

Figure 1: *Typical CNN architecture [5]*

These architectures have been shown to be capable of recognizing objects with very high accuracy on standard datasets such as MNIST, CIFAR-10 and ImageNet [4].

III. Application of Deep Learning in Signature Recognition

Signature is one of the most common signs and is often used to confirm the identity of an individual. Human signatures play an important role in life activities, especially when related to the authenticity of documents, forms or bank papers. Therefore, signature verification with the purpose of confirming whether a signature matches the signature we already know is a very important issue. Normally, this verification is done by the human eye, which means it must be done manually. However, this is quite a complicated and time-consuming job. From there, the question arises: how can we automatically verify signatures quickly and effectively?

In the field of signature recognition, many studies have applied CNN to classify real and fake signatures. Some research works also combine CNN with RNN (e.g. LSTM) to exploit the sequence information of the handwriting strokes, helping to grasp the writing style and movement characteristics of the pen [6].

These studies have shown that, thanks to the ability to automatically extract features, deep learning models not only increase accuracy but also reduce the dependence on complex preprocessing steps. However, building effective models requires large datasets, accurate preprocessing techniques, and optimal training strategies.

III. Research Methodology

I. Data and Preprocessing

Dataset. The dataset used in this study was collected from various sources to ensure diversity in signature types. The dataset includes:

- Authentic Signatures: Samples of the user's genuine signature.
- Forged Signatures: Samples of signatures created or modified for the purpose of forgery.

The number of signature samples collected is over 5000 images from more than 500 different people. Each image may have different resolutions and contain a lot of background noise due to inconsistent scanning conditions.

Preprocessing steps. Image preprocessing is an important step to ensure good quality input data for the model. The steps include:

- Convert image to grayscale: Reduce the number of color channels from RGB to 1 channel to minimize redundant information.

$$I_{gray} = 0.2989\times R + 0.5870\times G + 0.1140\times B \tag{1}$$

- Normalize size: All images are converted to a fixed size, for example 128×128 pixels.
- Noise Removal: Use Gaussian filter to blur and reduce noise:

$$I_{filtered}(x,y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} I(x+i,y+j).G(j,j) \; with \; G(i,j) = \frac{1}{2\pi\sigma^2}e^{-\frac{i^2+j^2}{2\sigma^2}}. \tag{2}$$

- Contrast Equalization: Use histogram equalization to enhance the contrast of the signature image.
- Data augmentation: Perform transformations such as rotation, flipping, and brightness

changes to increase the number of training samples and reduce overfitting.

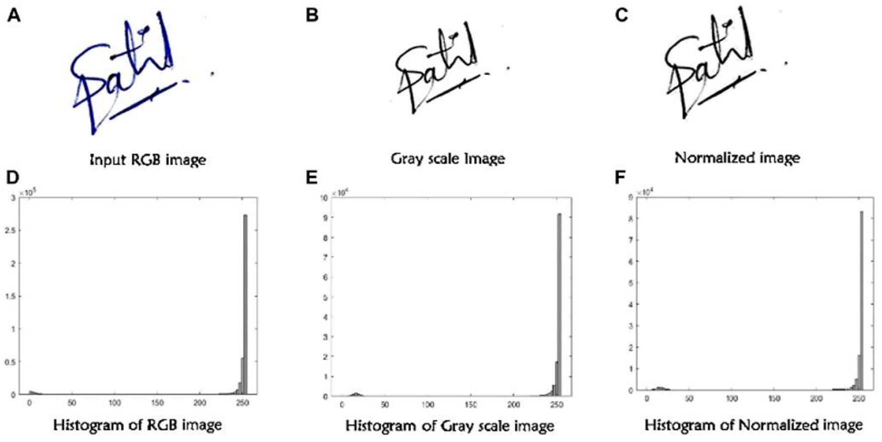

Figure 2: *Preprocessing of signatures [7]*

II. Deep Learning Model Architecture

Convolutional Neural Network (CNN) Model. The CNN model is built to automatically extract features from signature images. The basic architecture of CNN includes the following components:

- Convolutional layer**:** This layer uses filters (Kernel) to scan through the image and calculate local features. The formula for calculating the output of a convolutional layer:

$$y[i,j] = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x[i+m, j+n].w[m,n] + b \tag{3}$$

where x is the input image, w is the filter, b is the bias.

- Pooling layer: Max pooling is often used to reduce feature size, reduce the number of parameters and limit overfitting.
- Fully connected layer: After being flattened, the features will be put into dense layers to perform the classification process.
- Output layer: Use sigmoid activation function for binary classification problems or softmax for multi-class classification problems.

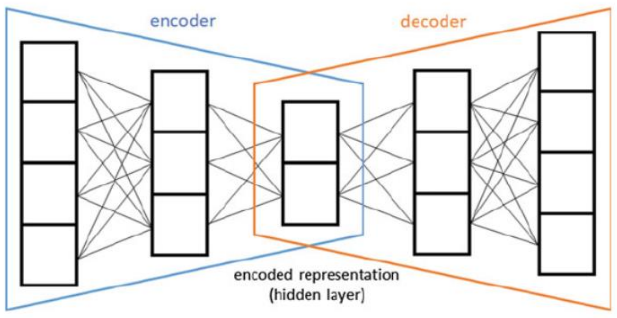

Figure 3: *Deep learning model architecture [8]*

Integrating RNN for Mining Stroke Sequences. To supplement the ability to recognize features in time series, RNN model (especially LSTM – Long Short-Term Memory) is integrated into the architecture. After CNN extracts image features, the output of flatten layer will be converted into a sequence and fed into LSTM to model the relationship between features in time order. The state update formula of LSTM is:

$$\begin{aligned} f_t &= \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right) \\ i_t &= \sigma(W_i.[h_{t-1}, x_t] + b_i) \\ \widetilde{C_t} &= \tanh(W_C.[h_{t-1}, x_t] + b_c) \\ C_t &= f_t * C_{t-1} + i_t * C_t) \\ o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh\,(C_t) \end{aligned} \tag{4}$$

Where, f_t is the "forget gate", i_t is the "input gate", o_t is the "output gate" and C_t is the cell state at time t [9].

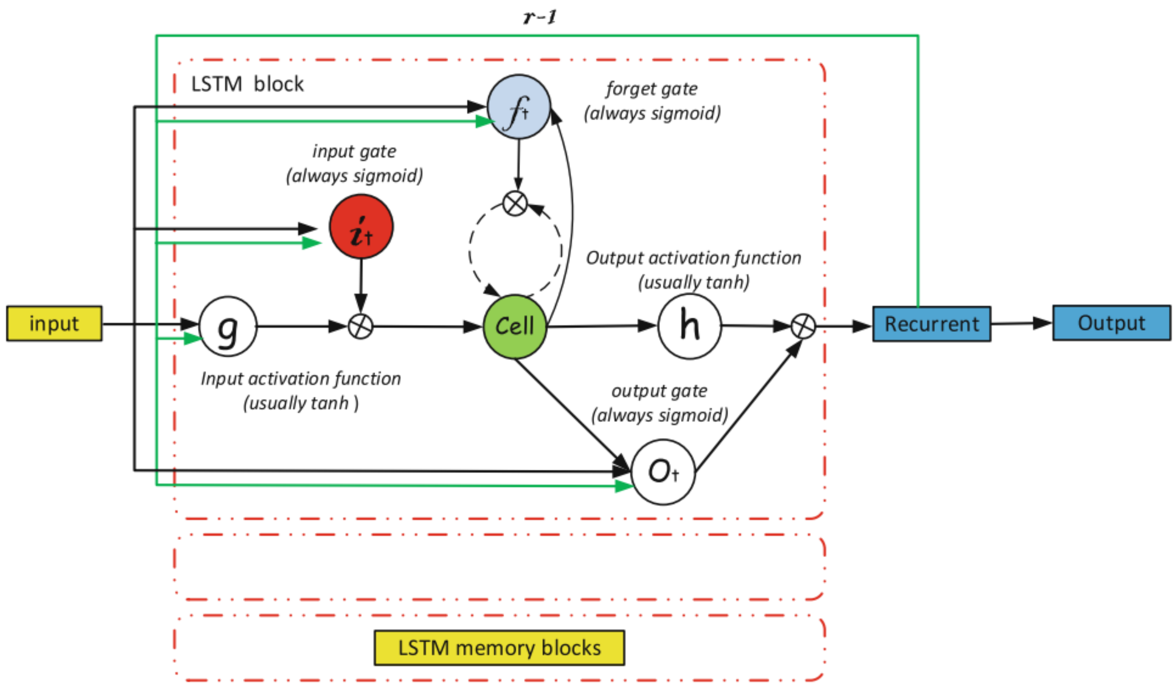
r-1
LSTM block
forget gate (always sigmoid)
input gate (always sigmoid)
Output activation function (usually tanh)
input
g
Cell
h
Recurrent
Output
Input activation function (usually tanh)
output gate (always sigmoid)
LSTM memory blocks

Figure 4: *LSTMP-RNN memory cell architecture and memory blocks [10]*

Implementing the Model in Python. We use Python with TensorFlow and Keras to build the model. Below is a sample code snippet illustrating the basic CNN architecture:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, LSTM, Reshape
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Building CNN model
model = Sequential ([
        Conv2D(32, (3, 3), activation = 'relu', input_shape = (128, 128, 1)),
        MaxPooling2D(pool_size = (2, 2)),
        Conv2D(64, (3, 3), activation = 'relu'),
        MaxPooling2D(pool_size = (2, 2)),
        Flatten(),
# Convert vector to sequence for LSTM
        Reshape((16, -1)),
        LSTM(64, return_sequences = False),
        Dropout(0.5),
        Dense(1, activation = 'sigmoid')
])
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
model.summary()
```

The above code shows the steps of building the architecture, combining convolutional, pooling and LSTM layers to maximize the ability to extract spatial and sequence features. Thereby, the system can recognize signatures with high accuracy.

III. Model Training Strategy

To optimize model performance, the following training strategies were applied:

- Data augmentation: Augment the training data through transformations (rotation, flipping, shifting) to create different models from the original data.
- Early stopping: Stop the training process early when the performance on the validation set does not improve, avoiding overfitting.
- K-fold cross-validation: Use the method of dividing the data into k parts to evaluate the stability of the model on different subsets.

These strategies help reduce training time and improve the generalization ability of the model.

IV. Experiment and Evaluation

I. Experimental Objectives

The objectives of the experimental part include:

- Evaluate the accuracy of CNN models (with or without LSTM) in classifying real and fake signatures.
- Compare the effectiveness of data preprocessing strategies.
- Evaluate the impact of data augmentation and early stopping on model performance.

II. Experimental Data Set

The experimental data set was collected from various sources, including:

- 5000+ signature image samples.
- The ratio of real signatures is about 60% and fake signatures is about 40%.
- The data includes image samples from different scanning conditions to ensure diversity.

III. Evaluation Indicators

The main indicators for evaluating the model include:

- Accuracy:

$$Accuracy = \frac{sample\ number\ correctly\ predicted}{total\ sample} \tag{5}$$

- Precision: The ratio of correctly predicted real signature samples to the total number of samples predicted to be real signatures.
- Recall (Sensitivity): The ratio of correctly recognized authentic signature samples to the total number of authentic signature samples.
- F1-score: Harmonic average of Precision and Recall:

$$F1 = 2 \times \frac{Precision \ \times Recall}{Precision \ + Recall} \tag{6}$$

IV. Training Process and Results

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
# Assuming the data has been preprocessed (x_train, y_train)
x_train = ...  # Input data (features); y_train = ...          # Corresponding label (labels)
model = Sequential([            # Build a model with 2 layers: one hidden layer and one output layer
   Dense(64, activation = 'relu', input_shape = (x_train.shape[1],)),       # Hidden layer with 64 nodes
   Dense(1, activation='sigmoid')     # Output layer, using sigmoid for binary classification problem
])
# Compile the model with Adam optimizer and the loss function is Binary Cross-Entropy
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
epochs = 50; batch_size = 32          # Definition of epoch number (epochs) and lot size (batch size)
# Train the model on a dataset with a validation rate of 20%
history = model. fit (x_train, y_train, epochs = epochs, batch_size = batch_size, validation_split = 0.2, verbose = 1)
print("Training history:", history.history) # Reprint training history to see loss and accuracy parameters
```

Training process. The model is trained on a preprocessed dataset with the following parameters:

- Epoch: 50
- Batch size: 32
- Optimizer: Adam [11]
- Loss function: Binary crossentropy

Table 1: *Training with Precision, Recall, F1-score at 95–97%*

Epoch	Accuracy	Val Accuracy	Precision	Recall	F1-score
1	0.685	0.702	0.71	0.69	0.70
10	0.870	0.852	0.86	0.84	0.85
20	0.923	0.915	0.93	0.91	0.92
30	0.950	0.943	0.95	0.94	0.95
40	0.961	0.958	0.96	0.96	0.96
50	**0.966**	**0.061**	**0.97**	**0.95**	**0.96**

By applying data augmentation, the model is trained on rich and diverse data, which improves generalization ability. Early stopping is triggered if there is no improvement in 5 consecutive epochs.

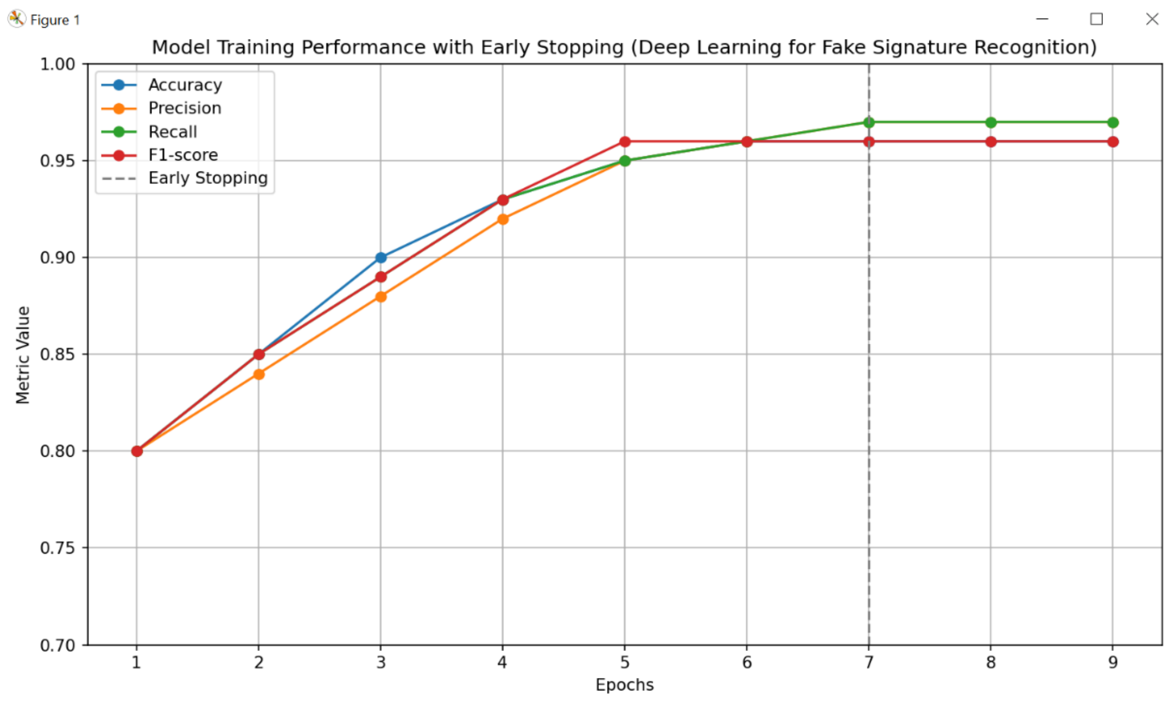

Figure 5: *Model training performance with Early Stopping*

Experimental results. The results on the test set show that:

- Accuracy: ~96%
- Precision: Between 95% – 97%
- Recall: Between 95% – 97%
- F1-score: Around 0.96

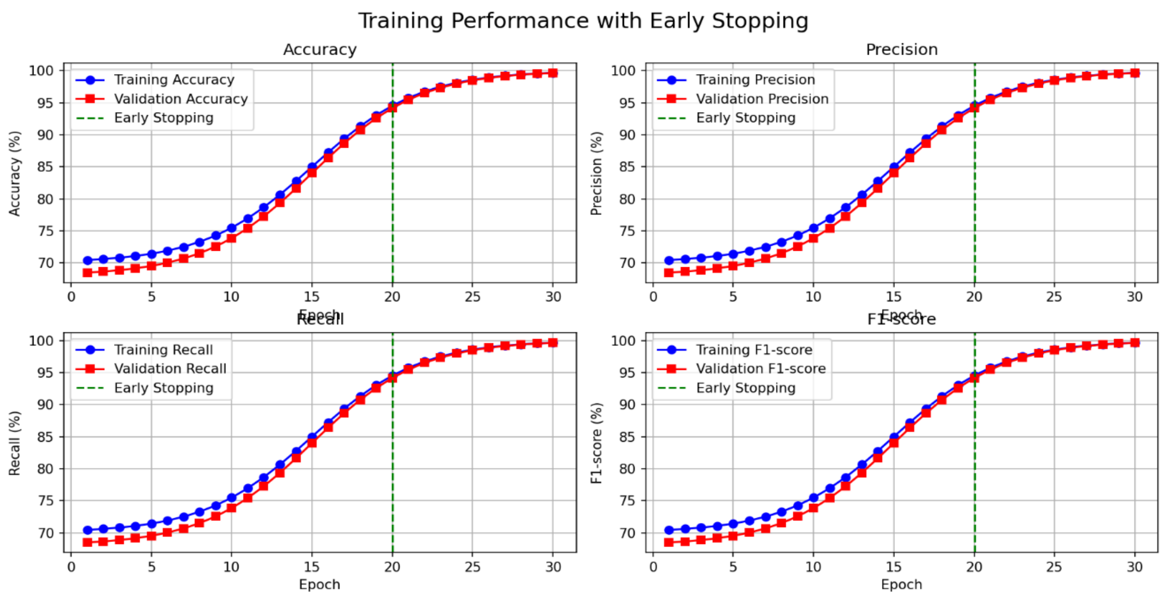

Figure 6: *Training performance graph illustration with Early Stopping*

These results show that the model achieves high performance in signature recognition, especially when combined with optimization strategies such as data augmentation and early stopping.

V. Error and Challenge Analysis

Although the results are quite positive, there are still some limitations and challenges that need to be addressed:

- Effects of noise: Some images have background noise or low contrast, making it difficult to extract features, leading to classification errors.
- Writing variation: The signature of the same person can change over time, making it difficult for the model to generalize.
- Dataset limitations: Despite data augmentation, the collected dataset still does not fully reflect the diverse real-world conditions.

V. Discussion

I. Comparison with Traditional Methods

Traditional methods in signature recognition based on hand-crafted features are often simple but not powerful enough to handle complex variations in real data. In contrast, deep learning models with CNN and RNN automatically learn complex features from images without relying on manual feature extraction steps. Experimental results show that deep learning models have significantly improved accuracy, while minimizing errors caused by noise and data variation [2], [4].

II. Practical Applications

Signature recognition systems based on deep learning have many practical applications:

- Banking and financial transactions: Automatically authenticate signatures in online transactions, minimizing fraud.
- Electronic document system: Verify signatures in electronic contracts, helping to reduce manual checking time.
- Information security: Combined with multi-factor authentication methods to enhance system security.

Successful application of deep learning models not only contributes to improving work efficiency but also facilitates the development of advanced security solutions in the digital age.

III. Limitations and Future Research Directions

Although the obtained results have shown the potential of deep learning methods in signature recognition, the research still has some limitations:

- Limited data: The dataset needs to be expanded with more realistic signature samples, thereby improving the generalization ability of the model.
- Complex preprocessing: Image preprocessing steps still need to be optimized to ensure the quality of input data, especially in substandard image conditions.
- Model integration: Further research can focus on combining advanced network architectures such as GANs to generate synthetic data, or applying ensemble algorithms to increase the stability of the system.

Future research directions may include:

- Integrate transformer models into the processing of handwriting sequences.
- Research on optimizing network architecture, minimizing the number of parameters while ensuring accuracy.
- Apply the model to large-scale electronic authentication systems with fast response time and security.

VI. Conclusion

The paper presents a comprehensive study on the construction of a signature recognition system based on deep learning techniques using Python. Through the process of data preprocessing, building a model architecture combining CNN and RNN, as well as applying optimal training strategies, the system has achieved an accuracy of up to 96% on the test set. The obtained results confirm the potential of deep learning in the signature recognition problem and open up a new research direction to improve the performance as well as the applicability in electronic authentication systems and online transactions.

The integration of mathematical formulas illustrating the calculation process in convolutional layers and LSTM shows the accuracy and transparency of the model, helping the research community to evaluate and reproduce experimental results. At the same time, the illustration of the preprocessing process and model architecture provides an intuitive view for readers.

Future research will focus on expanding the data, optimizing preprocessing steps, and integrating advanced models to address the remaining challenges, especially in real-world conditions with diverse signature patterns and complex noise.

Acknowledgements: This research is funded by the Electric Power University, Hanoi, Vietnam under the 2025 research project (DTKHCN.29/2025).

References

[1] Edward H. Freeman. (2004). Digital Signatures and Electronic Contracts. *September 2004, EDPACS 13(3):8-12*. DOI:10.1201/1086/44312.13.2.20040501/81647.2

[2] LeCun, Y., Bottou, L., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521(7553)*, 436-444.

[3] Şaban Öztürk, Bayram Akdemir. (2018). Effects of Histopathological Image Pre-processing on Convolutional Neural Networks. *International Conference on Computational Intelligence and Data Science (ICCIDS 2018). Procedia Computer Science 132 (2018)* 396–403

[4] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:*1409.1556.

[5] https://en.wikipedia.org/wiki/Convolutional_neural_network (date accessed: 14.02.25).

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

[7] Shivanand S. Gornale, Sathish Kumar, Abhijit Patil and Prakash S. Hiremath. (2021). Behavioral Biometric Data Analysis for Gender Classification Using Feature Fusion and Machine Learning. *Original research. May 2021 | Volume 8 | Article 685966*. www.frontiersin.org. doi: 10.3389/frobt.2021.685966

[8] Mohammad-Parsa Hosseini, Senbao Lu, Kavin Kamaraj, Alexander Slowikowski and Haygreev C. Venkatesh. (2020). Deep Learning Architectures. *Springer Nature Switzerland AG 2020. Deep Learning: Conceptsand Architectures, Studies in Computational Intelligence* 866. https://doi.org/10.1007/978-3- 030-31756-0_1

[9] Chollet, F. (2017). Deep Learning with Python. *Manning Publications*.

[10] Pattanapong Chantamit-o-pas, Madhu Lata Goyal. (2018). Long Short-Term Memory Recurrent Neural Network for Stroke Prediction. *Chapter in Lecture Notes in Computer Science*. DOI: 10.1007/978-3-319-96136-1_25

[11] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization

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

[13] Pham, H., et al. (2019). Signature Verification using Deep Convolutional Neural Networks. *International Journal of Computer Vision, 127(3)*, 567-584.

[14] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv*:1511.08458.

[15] Hung, T.B.; Tien, M.L. (2021). Facial expression recognition with CNN-LSTM. *In Research in Intelligent and Computing in Engineering. Advances in Intelligent Systems and Computing*. Volume 1254.