



An Integrated Convolutional And Recurrent Neural Network Approach For Handwriting Recognition

¹Zohra Fathima, ²Manju Sadasivan, ³Deepa A

¹MCA Student, ²Assistant Professor, ³Associate Professor

School of Science and Computer Studies

CMR University, Bengaluru, India

Abstract: Handwritten character recognition is an important component in many real-world applications, such as form processing automation and historical document scanning. Various methods have been advocated for mapping handwritten characters to machine-readable codes over the years. This paper introduces a hybrid deep learning method that integrates Convolutional Neural Networks (CNNs) for spatial feature discovery and Recurrent Neural Networks (RNNs), more specifically Long Short-Term Memory (LSTM) units, for sequence modelling. The model involves key preprocessing methods like normalization and missing value handling. The model was pre-trained on a large dataset of handwritten characters to make it stable. The hybrid CNN-RNN model is able to capture spatial and temporal patterns present in handwritten data well, resulting in high validation accuracy. For overcoming overfitting, dropout regularization was implemented. Nevertheless, noisy input and small dataset size restricted the model's ability to generalize. Additional improvements, such as sophisticated hyperparameter tuning and aggressive data augmentation, are needed to further boost performance. The study demonstrates the promise of hybrid deep learning models in handwritten character recognition and emphasizes the necessity for architectural enhancement and expansion of the dataset to facilitate more accurate and reliable performances.

Index Terms: Handwriting, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Accuracy, LSTM, Visualization

I. INTRODUCTION

Handwritten character recognition is an essential element in document digitization, improved accessibility, and automating business processes in various fields like healthcare, finance, and governmental services. This research is relevant in the following critical aspects:

- **Innovation:** Through the combination of Convolutional Neural Networks (CNNs) for spatial feature representation with Recurrent Neural Networks (RNNs) for modelling sequential data, the suggested hybrid model takes advantage of the complementarity between the two architectures. This integration gives rise to better accuracy in hand-written character recognition.
- **Practical Effect:** The system developed has the potential to automate and simplify operations like the processing of handwritten forms, digitization of historical manuscripts into digital repositories, and the efficient entry of data for visually impaired people.

- **Contribution to Research:** This paper contributes to handwriting recognition research by demonstrating the potential of CNN-RNN hybrid models for sequence-based visual recognition problems. It lays a solid foundation for future developments in deep learning models used in handwriting analysis.

In the realm of image-based tasks such as character recognition, the ability to accurately identify handwritten characters has broad applications, ranging from digitizing forms to automating documentation [1]. This paper tackles the challenge by implementing a hybrid deep learning model, leveraging the power of CNNs for feature extraction and RNNs for sequential learning.

Modern deep learning frameworks offer robust solutions for tasks that involve both image processing and temporal sequence learning [2]. The combination of CNNs and RNNs allows the model to extract spatial features and capture sequential patterns within image data, offering a more nuanced understanding of the input [3].

Accurate recognition of handwritten characters remains a complex problem due to variability in writing styles, noise in images, and overlapping characters. Traditional approaches fail to generalize well across large datasets with significant diversity [4]. Our research explicitly addresses these issues by crafting a model that is resilient in real-world environments.

Specifically, this paper aims to develop a model that:

- **High Accuracy:** Achieve leading-edge recognition accuracy on diverse, real-world handwritten test sets.
- **Noise Resilience:** Adapt techniques to deal with noisy or absent labels without sacrificing model stability.
- **Scalability & Generalisation:** Make the architecture scale to big character sets and accommodate new handwriting styles or alphabets with ease.

II. METHODOLOGY

Manual entry of data into computer systems is sometimes tedious and time-consuming, yet is still an indispensable activity in most fields. Nonetheless, thanks to technological advancements, the process is being streamlined day by day. As a result, the demand for reliable handwritten text recognition has increased. Building a system with the ability to automatically recognize, extract, and translate handwritten characters into machine-readable and human-understandable forms can significantly increase efficiency and minimize human effort [5]. This study examines the efficiency of handwritten character recognition using a hybrid CNN-RNN model. It also examines possibilities for future enhancement, especially through enlarging the dataset and optimizing the model structure to maximize accuracy, generalization, and scalability.

III. LITERATURE REVIEW

In the recent era of Artificial intelligence & Machine Learning and through the last ten years, the process of detecting and recognizing handwritten text has made significant progress. The tensorflow software[6] aims on using as input handwritten text in scanned form and giving it to a neural network (NN). The various authors of handwritten character recognition who used NN [7] represented numerous methods for evaluation of handwritten that can be stated as initial processing, acquisition of image, classification, extraction of features etc. it also concludes that nature and quality of input data greatly influence the quality of recognition of handwritten characters.

Authors in their review work of HWR[8] explained excellent conclusions regarding various methods used to recognise handwritten text present in diverse types of documents like images, videos etc and suggested semi incremental and incremental Recognition Method, slope and slant correction method and others. Artificial Neural Networks have many types that include Feed forward, Radial basis function, Kohonen Self Organizing, Convolution, Long Short Term (LSTM), Recurrent, and Modular [9]; all of these types were developed to practice different types of troubles.

The research namely “Effective offline handwritten text recognition model based on a progression-to-progression approach with CNN–RNN networks” [10] uses both CNN and RNN to distinguish hybrid hand text recognition using the progression-to-progression approach.

IV. OBJECTIVE OF THE MODEL

The model architecture is particularly developed to meet the inherent variability and sequential nature of handwritten characters. The model combines Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) units, to efficiently capture both the spatial and temporal features of the input data.

Handwritten images are the input to the system, which are initially processed by a sequence of convolutional layers. These convolutional layers extract the low-level and high-level spatial features that determine the structure and shape of the characters. The resulting feature maps are then fed into bidirectional LSTMs, which can capture the contextual dependencies both forward and backward. This enables the model to better comprehend character sequences in a word or sentence.

The fundamental goals of the model are:

- **Spatial Feature Extraction:** Use CNNs to extract informative and discriminative features from digitized character images.
- **Temporal Dependency Modelling:** Use RNNs, especially bidirectional LSTMs, to capture sequential relationships and patterns between characters.
- **Performance Optimization:** Optimize prediction accuracy while reducing overfitting using dropout regularization and batch normalization methods.

This design allows the model to generalize effectively across various handwriting styles and types and provide strong performance on actual handwritten character recognition tasks.

V. RESEARCH METHODOLOGY

a. Methodology of Collection

The dataset used in this paper is collected from handwritten text images, which are labelled with their corresponding identity (text label). The data includes variations in handwriting styles, noise, and varying image quality, making it a suitable testbed for model robustness. The CSV files provide the filenames of the images and their corresponding labels. The training dataset contains 30,000 images, while the validation set contains 3,000 images, ensuring a balanced dataset for evaluation.

b. Data Dictionary

- **FILENAME:** This field contains the name of the image file, representing a unique image.
- **IDENTITY:** The corresponding label (character or word) for each image, which the model aims to predict.

Figure 1 shows the code that loads the dataset:

```
train = pd.read_csv('written_name_train_v2.csv')
valid = pd.read_csv('written_name_validation_v2.csv')
```

Fig 1. Code to load the dataset.

c. Data Cleaning and Preprocessing

Before feeding data into the model, it's essential to handle missing values and prepare the data for training. Below are the detailed steps taken to clean and preprocess the dataset.

i) Identification and Handling of NULL Values

Missing labels are identified in the dataset, and appropriate measures are taken either to drop or impute these records. Figure 2 shows the total number of missing values in the training and validation sets.

```
print("Number of NaNs in train set      : ", train['IDENTITY'].isnull().sum())
print("Number of NaNs in validation set : ", valid['IDENTITY'].isnull().sum())
```

Number of NaNs in train set : 565
 Number of NaNs in validation set : 78

Fig 2. Total number of NaNs in the training and validation sets.

ii). Handling Missing Values

Based on the output, records with missing labels can be dropped since they cannot be used for supervised learning. The Python code used to drop the missing values is shown in figure 3

```
train.dropna(axis=0, inplace=True)
valid.dropna(axis=0, inplace=True)
```

Fig 3. Drop missing values from the set

iii). *Encoding Categorical Values*

The target labels are converted into numeric format, as models like RNNs require numerical input. If the characters are words or sequences, techniques such as label encoding or character embedding will be applied. Sample Code is given in figure 4.

```
print('True label : ',train_label[0], "IDENTITY", '\ntrain_y : ',train_y[0]),'\ntrain_label_len : ',train_label_len[0])  
      '\ntrain_input_len : ', train_input_len[0]))
```

True Label : NOUR
train_y : [13. 14. 20. 17. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
-1. -1. -1. -1. -1. -1.]
train_label_len : [4.]
train_input_len : [62.]

Fig 4. Code for Encoding

iv). Normalization

Image data needs to be normalized to ensure the model can converge effectively. Each pixel intensity is scaled between 0 and 1 and the code is shown in figure 5.

```
train_images = train_images / 255.0
```

Fig 5. Code to normalize.

v). Dimensionality Reduction

In some cases, dimensionality reduction techniques like PCA could be applied, but for this model, we rely on convolutional layers to reduce dimensionality through feature extraction.

VI. ALGORITHM DESCRIPTION

a. Algorithm Choice

This paper uses a hybrid model, combining CNN and RNN architectures to process both spatial and sequential information. The choice of CNN is due to its effectiveness in image recognition tasks, while RNN (specifically LSTM) is chosen for its ability to capture long-term dependencies in sequences. Figure 6 depicts the various key components.

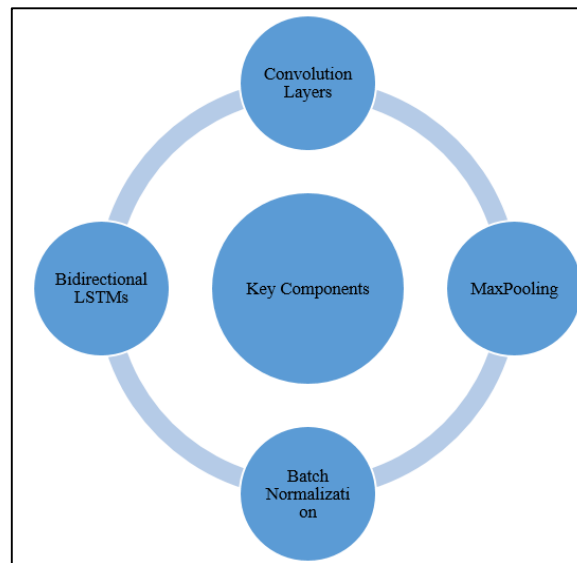


Fig 6. Key components of Model

- i) **Convolutional Layers:** Extract spatial features from images.
- ii) **MaxPooling:** Reduces the spatial dimensions of the image, keeping only the most relevant features.
- iii) **Batch Normalization:** Improves training speed and stability by normalizing activations.
- iv) **Bidirectional LSTMs:** Captures both forward and backward sequence dependencies, ensuring that context is retained across the sequence.

VII. DATA SET MANIPULATION

a. Identify X and Y Variables

- X: Image data representing the characters.
- Y: The corresponding text labels (characters or words).

b. Splitting the Data

Data is split into training and validation sets that is shown in figure 7 which is used to assess the model's performance on unseen data:

```

train_size = 30000
valid_size = 3000
num_epochs = 60
batch_size_value = 128
  
```

Fig 7. Splitting of data in train and validation set.

c. Application of the Chosen Algorithm

The CNN-RNN hybrid model is applied to the dataset, processing each image through the CNN layers, followed by the LSTM layers for sequence learning.

VIII. RESULT

a. Prediction of Y

After training, predictions are made on the test data. The output is a sequence of characters that the model predicts for each image which is shown in figure 8.

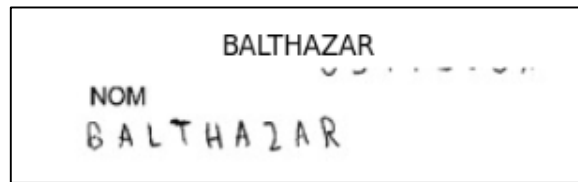


Fig 8. Sequence of characters as output for prediction.

b. Accuracy Check

Using classification metrics such as accuracy, precision, recall, and F1 score, the model's performance is evaluated. Libraries such as sklearn can be used to compute these metrics. Figure 9 shows the classification report based on the actual and predicted value. The model has been able to predict different characters with an accuracy of more than 90%, and words are predicted correctly with an accuracy of 78% . The line graph representing the accuracy of the model is shown in figure 10.

```
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred))
```

Fig 9. Performance evaluation using classification report

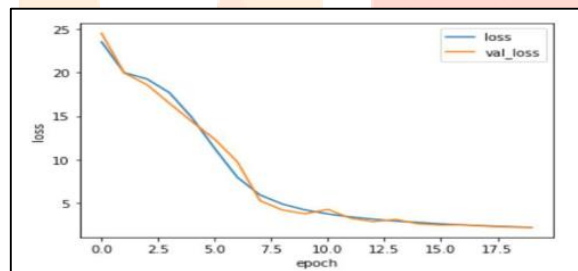


Fig 10. Accuracy of the model.

c. Visualization

Visualizations of the training process, including the training and validation accuracy and loss over epochs, help assess model performance. Python code for this operation is depicted in figure 11.

```
train = pd.read_csv('written_name_train.csv')
val = pd.read_csv('written_name_validation.csv')

plt.figure(figsize=(15, 10))

for i in range(5):
    m = plt.subplot(2, 1, i+1)
    img_dir = 'train_0/train_0/train_imgs/' + FILENAME
    img = cv.imread(img_dir, cv.IMREAD_GRAYSCALE)
    plt.imshow(img, cmap = 'gray')
    plt.title(train.loc[i, 'IDENTITY'], fontsize=12)
    plt.axis('off')

plt.subplots_adjust(wspace=0.3, hspace=0.3)
```

Fig 11. Python code for visualization

d. The CNN-LSTM model

Each image is processed through CNN layer first then sequence learning is accomplished by LSTM Layers. The is applied to the entire dataset. Figure 12 displays the code for this model. Table 1 describes the type of Layers, bidirectional LSTMs, shapes & parameter values.

```
# Input layer: expects grayscale images of shape (28, 28)
input_data = Input(shape=(28, 28, 1), name='input')

# First Convolutional Block
inner = Conv2D(12, (3, 3), padding='same', name='conv1', kernel_initializer='he_normal')(input_data)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max1')(inner)

# Second Convolutional Block
inner = Conv2D(24, (3, 3), padding='same', name='conv2', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max2')(inner)
inner = Dropout(0.3)(inner)

# Third Convolutional Block
inner = Conv2D(24, (3, 3), padding='same', name='conv3', kernel_initializer='he_normal')(inner) # 128 filters, 3x3 kernel
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max3')(inner)
inner = Dropout(0.3)(inner)

# Reshape layer to prepare for RNN input
inner = Reshape(target_shape=(64, 1024), name='reshape')(inner) # Reshape to (64, 1024)
inner = Dense(64, activation='relu', kernel_initializer='he_normal', name='dense1')(inner)

# Recurrent Neural Network (RNN) layers
inner = Bidirectional(LSTM(256, return_sequences=True), name='lstm1')(inner) # Bidirectional LSTM with 256 units, returns sequences
inner = Bidirectional(LSTM(256, return_sequences=True), name='lstm2')(inner) # Another bidirectional LSTM with 256 units

# Output layer
inner = Dense(10, kernel_initializer='he_normal', name='dense2')(inner)
y_pred = Activation('softmax', name='softmax')(inner)

# Model definition
model = Model(inputs=input_data, outputs=y_pred)
model.compile()
```

Fig 12. Python code for CNN_LSTM Layers.

IX. DISCUSSION

The model had a validation accuracy of about 90%, which proves its ability to recognize handwritten characters. Although the results show promise, there is always room for improvement. The performance of the model can be enhanced by carrying out the following:

- **Dataset Expansion:** Adding an expanded and diversified dataset to expose the model to even more different handwriting styles and variations.
- **Hyperparameter Tuning:** Tuning important hyperparameters like learning rate, batch size, and number of epochs to maximize training effectiveness and model accuracy.
- **Data Augmentation:** Implementing augmentation operations (e.g., rotation, scaling, shift, noise injection) to synthetically augment the size and diversity of the training set, thus enhancing the generalization ability of the model.

These improvements have the potential to greatly enhance the model's accuracy and robustness on a wide range of real-world handwriting scenarios.

X. CONCLUSION

The main goal of this research was to design a deep learning model that integrates Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for better handwritten character recognition. The suggested hybrid model takes advantage of the spatial feature extraction ability of CNNs and the sequential modelling power of RNNs to improve recognition accuracy. The main contributions of this work are as follows:

- **Model Performance:** The hybrid CNN-RNN model showed robust performance, with a validation accuracy of 90%. This indicates the capability of the model to successfully learn and combine both spatial and temporal features from handwritten input images.
- **Sequential Learning Capability:** The application of bidirectional Long Short-Term Memory (Bi-LSTM) networks greatly enhanced the model's capacity to learn temporal dependencies in handwriting. This helped improve the interpretation of character sequences and improve complex handwriting pattern recognition.
- **Effective Preprocessing:** Extensive data preprocessing—normalization and missing value handling—were pivotal in enhancing model performance. These processes allowed the model to receive clean, uniform input, promoting faster convergence and better predictions.

TABLE 1: LAYERWISE DESCRIPTION

Model: "functional"		
Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 256, 64, 1)	0
conv1 (Conv2D)	(None, 256, 64, 32)	320
batch_normalization (BatchNormalization)	(None, 256, 64, 32)	128
activation (Activation)	(None, 256, 64, 32)	0
max1 (MaxPooling2D)	(None, 128, 32, 32)	0
conv2 (Conv2D)	(None, 128, 32, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 128, 32, 64)	256
activation_1 (Activation)	(None, 128, 32, 64)	0
max2 (MaxPooling2D)	(None, 64, 16, 64)	0
dropout (Dropout)	(None, 64, 16, 64)	0
conv3 (Conv2D)	(None, 64, 16, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 64, 16, 128)	512
activation_2 (Activation)	(None, 64, 16, 128)	0
max3 (MaxPooling2D)	(None, 64, 8, 128)	0
dropout_1 (Dropout)	(None, 64, 8, 128)	0
reshape (Reshape)	(None, 64, 1024)	0
dense1 (Dense)	(None, 64, 64)	65,600
lstm1 (Lstm)	(None, 64, 512)	697,408
lstm2 (Lstm)	(None, 64, 512)	1,574,912
dense2 (Dense)	(None, 64, 30)	15,390
softmax (Activation)	(None, 64, 30)	0
Total params: 2,486,078 (9.18 MB)		
Trainable params: 2,486,030 (9.18 MB)		
Non-trainable params: 448 (1.75 KB)		

Overall, the hybrid CNN-RNN architecture offers a strong solution for handwritten character recognition, with room for improvement through larger datasets, hyperparameter optimization, and sophisticated augmentation methods.

REFERENCE

- [1] Nisha Sharma, Tushar Patnaik, and Bhupendra Kumar, "Recognition for handwritten English letters: A Review" International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 7, 2013. [Online]. Available: https://www.ijeit.com/vol%202/Issue%207/IJEIT1412201301_57.pdf
- [2] J. Pradeep, E. Srinivasan, and S. Himavathi, "Diagonal based feature extraction for handwritten alphabets recognition System using neural network" International Journal of Computer Science and Information Technology (IJCSIT), Vol 3, No-1, 2011. [Online]. Available: <https://airccse.org/journal/jcsit/0211ijcsit03.pdf>
- [3] Prof. Jayakumar K., Prakhar Dungarwal, "Handwritten Text Recognition using Deep Learning for Automated Paper Checking" International Research Journal of Engineering and Technology, Volume: 08 Issue: 11, 2021. [Online]. Available: <https://www.irjet.net/archives/V8/i11/IRJET-V8I1142.pdf>
- [4] Jehad Ali, Rehanullah Khan, Nasir Ahmad, Imran Maqsood, "Random Forests and Decision Trees" International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3, 2012. [Online]. Available: https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees
- [5] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks" in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 369–376. [Online]. Available: <https://doi.org/10.1145/1143844.1143891>
- [6] Sri. Yugandhar Manchala, Jayaram Kinthali, Kowshik Kotha, Kanithi Santosh Kumar, Jagilinki Jayalaxmi, "Handwritten text recognition using deep learning with Tensorflow" International Journal of Engineering and Technical Research, Vol. 9, Issue 5, 2020. [Online]. Available: <https://www.ijert.org/handwritten-text-recognition-using-deep-learning-with-tensorflow>
- [7] Pankaj Rani, Sushil Kakkar, Shweta Rani, "Offline Handwritten Character Recognition using Neural Network" International Journal of Computer Applications (0975 – 8887), International Conference on Advances in Emerging Technology (ICAET 2016). [Online]. Available: <https://research.ijcaonline.org/icaet2016/number1/icaet022.pdf>

- [8] Salma Shofia Rosyda, and Tito Waluyo Purboyo, "A review of various handwriting recognition methods" *International Journal of Applied Engineering Research*, Vol. 13(2): 1155-1164, 2018. [Online]. Available: https://www.ripublication.com/ijaer18/ijaerv13n2_44.pdf
- [9] Shan Wang, Jinwei Di, Dan Wang, Xudong Dai, Yabing Hua, Xiang Gao, Aiping Zheng, and Jing Gao, "State-of-the-Art Review of Artificial Neural Networks to Predict, Characterize and Optimize Pharmaceutical Formulation." *Pharmaceutics*, 14(1), 183, 2022. [Online]. Available: <https://doi.org/10.3390/pharmaceutics14010183>
- [10] Geetha Rg, T. Thilagam, and T. Padmavathy, "Effective offline handwritten text recognition model based on a sequence-to-sequence approach with CNN–RNN networks" *Neural Computing and Applications*, 33(17): 1-12, 2021. [Online]. Available: <https://link.springer.com/article/10.1007/s00521-020-05556-5>

