Handwritten Text Recognition Using Machine Learning

Erram Aishwarya Reddy[1], Kasireddy Raghuvardhan Reddy[2], Mohammad Irshad[3]

[1]*Student & Sreenidhi Institute of Science and Technology, Ghatkesar*
[2]*Student & Sreenidhi Institute of Science and Technology, Ghatkesar*
[3]*Professor, Dept. of Computer Science and Engineering, Sreenidhi Institute of Scinence and Technology, Telangana, India*

---***---

Abstract - *The objective of HTR is to automate the process of converting handwritten documents into digital text, which is much easier to store, edit, and search. HTR is used in various applications, including digitizing historical documents, recognizing handwriting in online forms, and improving accessibility for people with visual impairments. We propose a system that uses both the CNN and RNN neural networking algorithms to predict the Handwritten text recognition.*

Key Words: **CNNs, HTR, RNNs, CER, Accuracy, Recognition, Training**

1. INTRODUCTION

Handwritten text recognition (HTR) is an area of artificial intelligence that deals with the development of algorithms capable of recognizing and interpreting handwritten text. HTR aims to automate the process of turning handwritten papers into editable, searchable, and readily stored digital text. HTR can be used for a variety of things, such as digitizing old documents, reading handwriting on the screen, and enhancing accessibility for those with visual impairments. Preprocessing, feature extraction, and classification are some of the processes that make up a typical HTR system. The input image is improved upon and prepared for future processing during the preprocessing stage. The neural network extracts feature from the preprocessed image during the feature extraction stage that are important for reading the handwritten text.

2. LITERATURE SURVEY

2.1 Existing System:

Using CNN's: Handwritten Text Recognition (HT) using Convolutional Neural Networks (CNNs) has become increasingly popular in recent years. The primary benefit of CNNs is their capacity to automatically extract pertinent characteristics from the input picture, which makes them especially well-suited for HTR and other image identification tasks.

Using RNN's: There are various HTR systems in use today that exclusively use RNNs to recognize handwritten text. The Long Short-Term Memory (LSTM) network, a kind of RNN that is intended to better capture long-term dependencies in the data, is one well-known example.

Using both CNN's and RNN's: There are few HTR systems which are built using combination of the CNN's and RNN's where the data is loaded into the training model is first passed into the set of CNN layers and then the outcome of the CNN layers is passed through the RNN layers to train the model and prepare the model.

2.2. Proposed System:

Proposed system contains a set of CNN layers which would take the inputs from the dataset that is given to train the model and that would load the data into 7 layers of CNN (Continuous Neural Networks) and give the output to the set of RNNs (Recurrent Neural Networks) then the output of both CNNs and RNNs are given to CTC a model of Tensor flow.

2.3 Proposed system Architecture:

The architecture depicted in the diagram is a deep learning model used for text recognition. The model takes in a batch of images where each image has dimensions of (batchSize, imgSize[0], imgSize[1]), where imgSize is a tuple that specifies the height and width of the image, and batchSize is the number of images fed into the model at once.

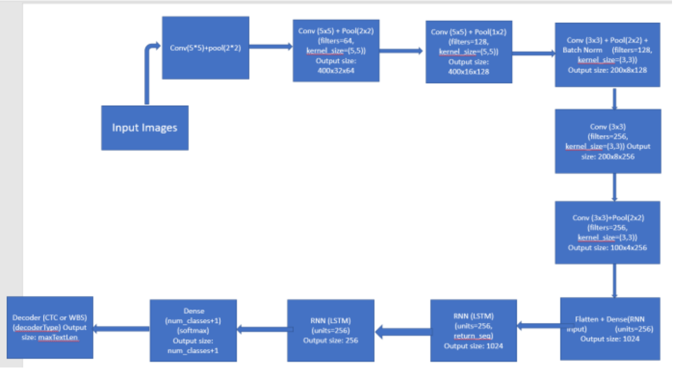

Input Images
Conv(5*5)+pool(2*2)
Conv (5x5) + Pool(2x2) (filters=64, kernel_size=(5,5)) Output size: 400x32x64
Conv (5x5) + Pool(1x2) (filters=128, kernel_size=(5,5)) Output size: 400x16x128
Conv (3x3) + Pool(2x2) + Batch Norm (filters=128, kernel_size=(3,3)) Output size: 200x8x128
Conv (3x3) (filters=256, kernel_size=(3,3)) Output size: 200x8x256
Conv (3x3)+Pool(2x2) (filters=256, kernel_size=(3,3)) Output size: 100x4x256
Flatten + Dense(RNN input) (units=256) Output size: 1024
RNN (LSTM) (units=256, return_seq) Output size: 1024
RNN (LSTM) (units=256) Output size: 256
Dense (num_classes+1) (softmax) Output size: num_classes+1
Decoder (CTC or WBS) (decoderType) Output size: maxTextLen

Fig -1: Design of the Model

3. UML DIAGRAMS

3.1 Use case Diagram

In the Unified Modelling Language (UML), a use case diagram is a particular kind of behavioral diagram that shows how a system interacts with users or other entities. This diagram provides a simple-to-understand representation of a system's functioning and potential applications. It is made up of actors, use cases, and the connections among them. Use cases define how the system behaves from the perspective of the user, whereas actors represent the various user types or outside entities that interact with the system. Use cases and actors may be linked together by straightforward associations or more intricate linkages like generalizations, extends, and includes.

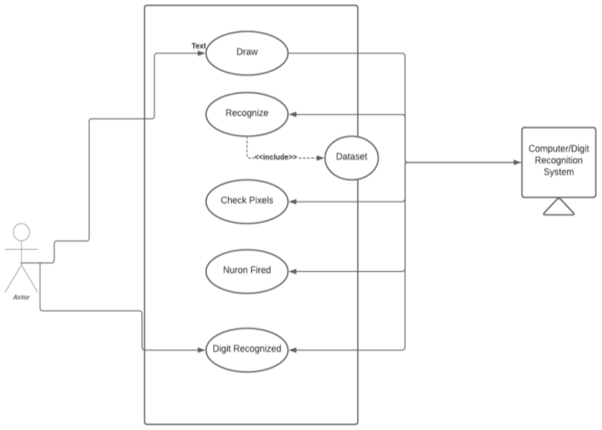


Fig -2: Use Case Diagram

3.2 Activity Diagram

A sort of behavioral diagram in Unified Modelling Language (UML) called an activity diagram shows how activities or actions move through a system. It offers a visual depiction of the procedures or actions necessary to finish a particular job or use case within the system. The nodes in the activity diagram stand in for the various activities or actions, while the edges show the order in which those activities or actions are carried out. The nodes might represent straightforward processes, like sending a message or making a choice, or they can be more intricate processes with several phases. The boundaries might be solid lines to denote a predetermined order of events or dotted lines to denote other routes or conditional branching.

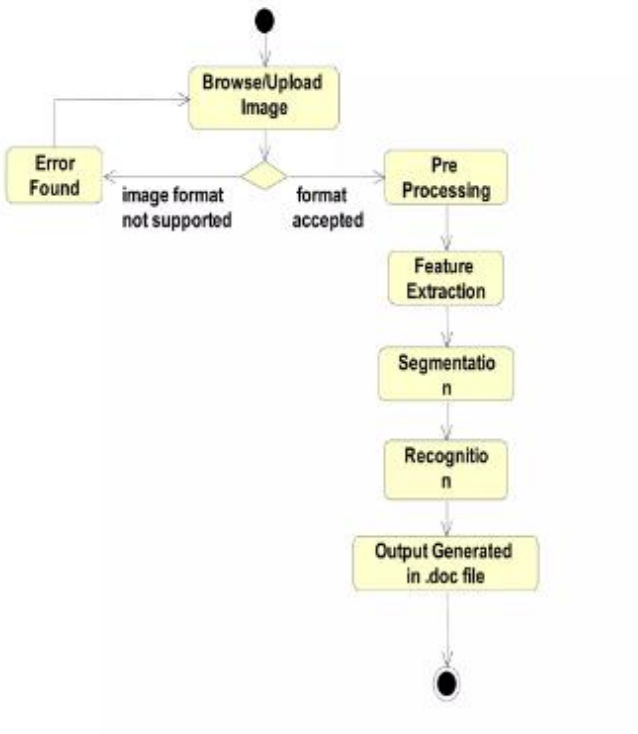


Fig -3: Activity Diagram

3.3 Class Diagram

A class diagram is a type of UML diagram that is used to represent the structure of a system by illustrating its classes, attributes, methods, and the relationships between them. It is a graphical representation that enables developers to visualize the different components of a system and how they interact with one another. The primary purpose of a class diagram is to describe the classes in a system and their relationships with each other. The class diagram can be used during various stages of software development, from design to implementation and maintenance.

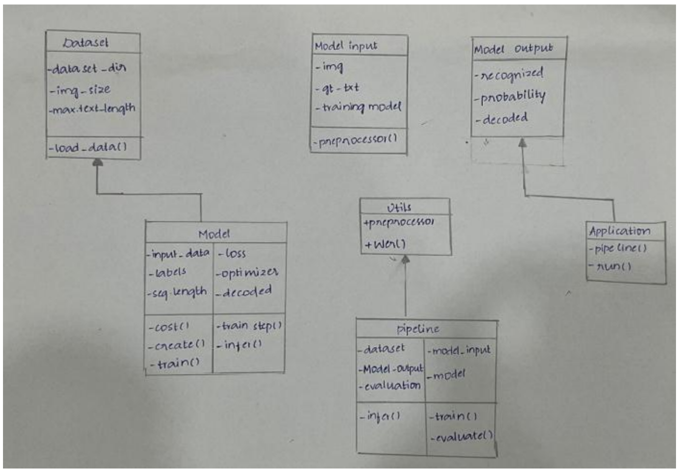


Fig -4: Class Diagram

4. REQUIREMENTS

4.1 Hardware Requirements

- Processor: Intel Core i5 and above
- RAM: 8GB and above
- Memory: 512GB SSD and above

4.2 Software Requirements

- Windows or macOS
- Visual Studio
- Python 3.6

5. METHODS USED

5.1. Data Collection:

Data collecting is a crucial component of machine learning, and the caliber of the data has a direct bearing on the precision of the models that are produced. The practice of acquiring information about user access to and activity inside the systems and applications of an organization is referred to as data collection in the context of identity and access management (IAM). Timestamps for login and logout attempts, unsuccessful login attempts, access requests, permissions, and other information can be included in this data. In general, gathering data is a crucial step in utilizing machine learning to improve IAM capabilities inside an organization. The accuracy and use of the data obtained may be ensured by careful planning, execution, and preprocessing, resulting in more efficient machine learning models and a better security posture.

5.2. Data Preprocessing:

A crucial step in every machine learning project is data preprocessing, which involves a number of methods and procedures used to convert raw data into a format that can be utilized for training a model. This step comprises extensively cleaning and pre-processing the obtained data, which entails locating and fixing problems including missing data, outliers, inaccurate values, and duplicate characteristics. The next step is to change the data's format so that it is acceptable for analysis. This may require normalizing, standardizing, scaling features, and encoding categorical data. By minimizing mistakes and inconsistencies in the data, this stage helps the model function more accurately and produce findings that can be trusted. Overall, data preparation is a crucial part of machine learning, and the precision and success of the final model may be greatly influenced by the caliber of the input data and the efficiency of the preprocessing methods used.

5.3. Training the model:

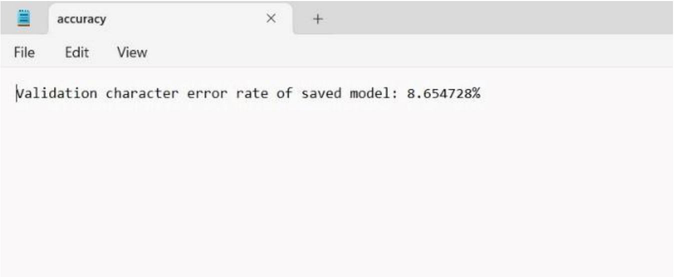
It is also essential to select the right algorithm for the job at hand. Machine learning algorithms come in many different varieties, such as supervised learning, unsupervised learning, and reinforcement learning. When a labelled dataset is available, supervised learning techniques are applied; the system is trained on the input and output data to predict future output. When there are no labels on the data and the computer must figure out patterns and connections on its own, unsupervised learning approaches are utilized. Models are taught to make decisions based on input from their surroundings through reinforcement learning. The efficacy and precision of the trained model are evaluated using a test dataset. The model may also be applied to fresh data if it successfully predicts the test data.In conclusion, there are several steps involved in using machine learning algorithms to train a model. Among these processes are data preparation, picking the best strategy, and model optimization. These methods may be used to create precise and potent machine learning models for a range of applications.

5.4. Recognition:

The model may occasionally need to be tweaked or retrained with new data to improve performance. The approach may be applied in different circumstances to identify trends or make predictions in real-time applications. A trained image recognition model may be used to identify objects in images or videos, in contrast to a taught language model, which can be used to generate text or make language-based predictions. Overall, the ability to analyze data quickly and accurately can lead to significant improvements in efficiency and decision-making, which is why the use of trained machine learning models for prediction and recognition tasks has a wide range of real-world applications in sectors like manufacturing, healthcare, and finance.

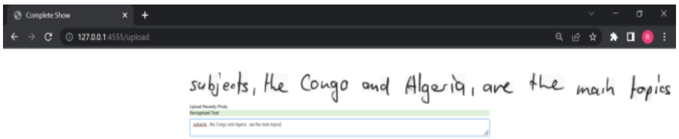
5.5. Output Generation:

A few critical components are required for Handwritten Text Recognition output creation on an intuitive interface to guarantee an effective and user-friendly process. The recognition system's accuracy should be considered initially. A highly precise system will generate output that is more dependable, which will boost user happiness and confidence. Users want the result to be created promptly and in real-time, therefore speed of the system is equally important to accuracy. The user interface itself is another crucial factor for output production. The user should be provided with clear instructions and feedback via an intuitive and simple-to-use interface. The result must be presented clearly and succinctly, ideally with formatting choices so that the user may adjust it to their own requirements. A few critical components are required for Handwritten Text Recognition output creation on an intuitive interface to guarantee an effective and user-friendly process. The recognition system's accuracy should be considered initially. A highly precise system will generate output that is more dependable, which will boost user happiness and confidence. Users want the result to be created promptly and in real-time, therefore speed of the system is equally important to accuracy. The user interface itself is another crucial factor for output production. The user should be provided with clear instructions and feedback via an intuitive and simple-to-use interface. The result must be presented clearly and succinctly, ideally with formatting choices so that the user may adjust it to their own requirements.



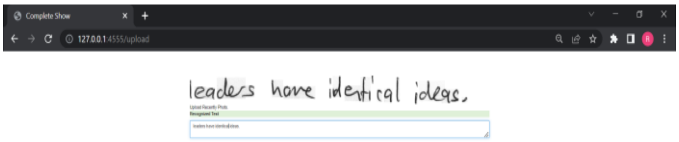
```
validation character error rate of saved model: 8.654728%
```

Fig -5: Accuracy



subjects , the Congo and Algeria , are the main topics

Fig -6: Output 1



leaders have identical ideas.

Fig -7: Output 2

6. APPLICATION

- Document Digitization
- Postal Services
- Education
- Banking
- Historical Archives
- Medical Services
- Law Enforcement

7. CONCLUSION

Handwritten Text Recognition is an important application of machine learning that has the potential to significantly improve productivity and efficiency in various industries. The use of Convolutional Neural Networks and Recurrent Neural Networks has proved to be an effective approach to achieve high accuracy in recognizing handwritten text. CNNs are used for feature extraction from images of text, while RNNs are used to model the sequential nature of handwriting. Now that several neural network designs have been merged, it is feasible to develop sophisticated models that have a high degree of handwriting recognition accuracy. However, the quality of the training data and the tuning of the hyperparameters have a significant impact on how well the Handwritten Text Recognition system performs, just like with any machine learning model. It is essential to thoroughly test and assess the model to ensure that it performs well with a variety of handwriting inputs and styles. Planning and executing tests for handwritten text recognition should consider the unique qualities and challenges of this application as well as the need to test for accuracy, robustness, and scalability.

8. FUTURE SCOPE

With the use of CNNs and RNNs, handwritten text recognition has advanced significantly in recent years and may continue to do so. The accuracy and effectiveness of handwriting recognition models may be increased in a few ways with the development of deep learning techniques. Convolutional and recurrent neural network hybrid models, which can capture both spatial and sequential information in the input, are one field of research. Enhancing the models' capacity to recognize handwriting patterns from many languages and cultures is another area of emphasis, since doing so may open the door to a more all-encompassing and universal method of text identification.

REFERENCES

[1] "Handwritten Text Recognition: An Introduction" by Stefan Jaeger, Stephan Rusinol and Ernest Valveny.

[2] "Handwriting Recognition: Soft Computing and Probabilistic Approaches" by Sargur N. Srihari.

[3] "Handwriting Recognition: A Comprehensive Guide" by Sargur N. Srihari.

[4] "Deep Learning for Computer Vision" by Adrian Rosebrock.

[5] "Python Machine Learning" by Sebastian Raschka.

[6] "TensorFlow for Deep Learning" by Bharath Ramsundar and Reza Bosagh Zadeh.

[7] "PyTorch Deep Learning Hands-On" by Sherin Thomas and Sudhanshu Passi.

[8] The TensorFlow website (https://www.tensorflow.org/) and PyTorch website (https://pytorch.org/) offer a wealth of tutorials and documentation for deep learning, including handwriting recognition using CNNs and RNNs.