# HANDWRITTEN SCRIPT TO TEXT CONVERTER

LOKESH SINGH SHEKHAWAT
Department of Computer Science and Engineering
Apex Institute Of Technology Chandigarh University
Mohali -140413, Punjab
20BCS6692@cuchd.in

***Abstract*— In this digital era, computers and phones are more important than ever, still many people prefer to use handwritten documents instead of digital format. Sometimes, it is difficult to store, carry and read through all handwritten documents at a time so there need for converting handwritten documents into digital versions. Here we may use two classifications to achieve this: Classifying words directly and character segmentation. This model is based on Neural Network, where this will be trained using a machine learning approach on word images from the (IAM) dataset. The input layers and other layers as well can be kept small for word images, Neural networks, in this process training will be done on the CPU instead of the GPU. This project model will be consisting of CNN layers, RNN layers, and CTC loss and decoding layer.**

**The result for this model will be displayed as an output in the digital version of handwritten documents after recognition. This model is all for building your own handwritten recognition system with TensorFlow. It covers detailed intuition about architecture and how I reach the solution and increase accuracy.**

***Keywords*: - *Machine learning, Segmentation, Neural Network, IAM dataset, CNN, RNN, CTC, Digital version***

## INTRODUCTION

There are so many digital writing tools, still, people use to write handwritten notes. There are so many drawbacks of handwritten documents. It is difficult to store, share, carry, read, and search through effectively and efficiently. Thus, this research paper will help to turn handwritten scripts into text format. Text format will be easy to share, analyze and search through and will help people to access it easily.

The aim of this model is to explore the work of classifying handwritten documents and converting them into digital text format.


*Figure 1:* Image of the word (taken from IAM) and its transcription into digital text

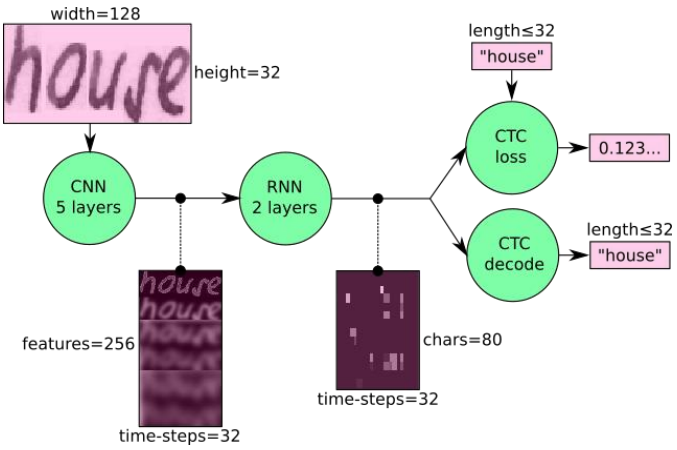



*Figure 2: Usage of Neural Networks in recognition*


*Figure 3: Examples of handwriting from IAM dataset*

Machine Learning is a data analytics technique that trains computers to learn from experience in the same way that humans and animals do.

Machine Learning algorithms employ computational methods to "learn" information directly from data rather than depending on a model based on a preconceived equation. [1]
Machine Learning has emerged as a crucial tool for resolving issues in fields such as finance and healthcare.
Pattern Recognition: Pattern recognition is the process of identifying patterns using machine learning techniques. It categorizes data using statistical data or knowledge derived from patterns and their representation.
Credit scoring and algorithmic trading are two examples of computational finance.
Processing of images and Face recognition, motion detection, and object detection are all examples of computer vision.
Tumor detection, medication discovery, and DNA sequencing are all examples of biology.
Voice recognition applications based on natural language processing

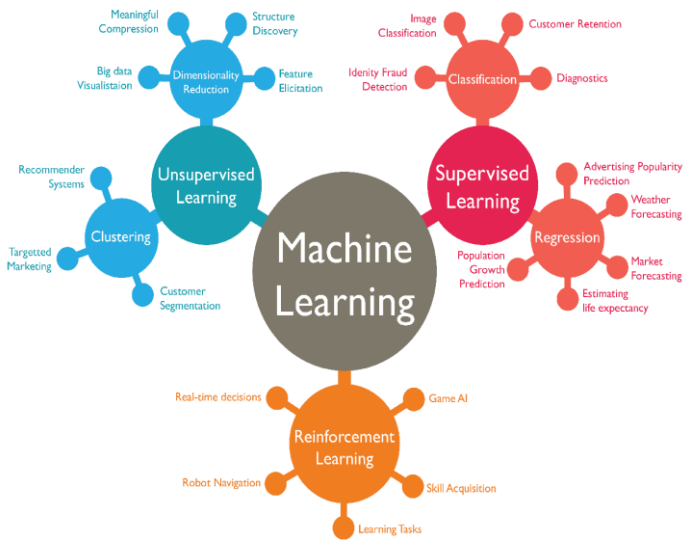



*Figure 4: - Machine Learning Representation*

A neural network is made up of neurons, which are the processing units. Each neuron is linked to a number of other neurons as well as the input nodes. [2]
Neural networks are a simple computer paradigm for the real-time recognition of complex jobs.
Single-layer networks, multilayer feedforward networks, and feedback networks are the three types of neural networks classified by the NN.
The two underlying mathematical concepts for deriving learning algorithms are gradient descent and the relaxation approach.
Because of their essential role in neural networks, learning algorithms are the focus of a lot of research.
As we can see in the given figure [2] there multiple layers for neural networks and deep neural networks. These layers represent the processing of data in neural networks and their integrity

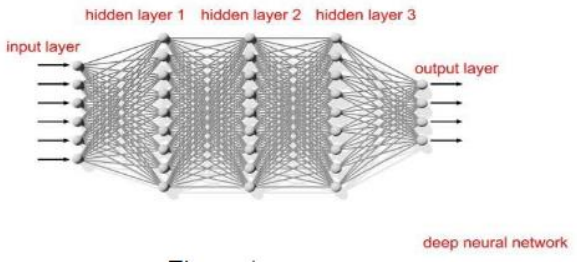



*Figure 5: - neural network representation*

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take an image as input, give significance (learning weights and biases) to different aspects of the image, and distinguish one from the other. [3]
When compared to other classification algorithms, the amount of pre-processing required by a ConvNet is significantly less. Whereas primitive methods require hand-engineering of filters, ConvNets can learn these filters/characteristics with enough training. [4]
The structure of a ConvNet is inspired by the organization of the Visual Cortex and is similar to the connectivity pattern of Neurons in the Human Brain.
Individual neurons can only react to changes in a small area of the field of vision called the Receptive Field.

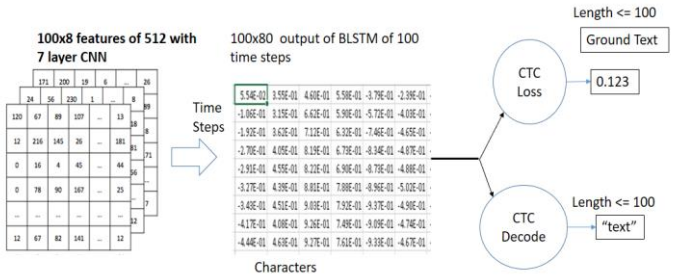



*Figure 6: CNN and CTC loss and usage representation*

Connectionist Temporal Classification (CTC) is a Neural Network output that can be used to solve sequence problems such as handwriting and speech recognition where the time frame varies.
Using CTC eliminates the need for a pre-aligned dataset, making the training process much simpler.
For each time step, they generate a character score, which is represented by a matrix.
This matrix will now be used to train the neural network, i.e. to calculate the loss.
Decoding the Neural Network's Output
The CTC operation aids in the completion of both tasks.
A Connectionist Temporal Classification Loss, or CTC Loss, is developed for jobs that require sequence alignment but are difficult to achieve.
Using CTC eliminates the need for a pre-aligned dataset, making the training procedure much simpler.

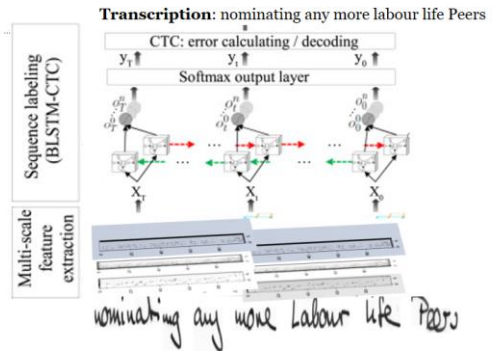



*Figure 7: Transcription procedure*

Recurrent neural networks (RNN) are a type of NN that is widely used in sequence analysis because they are designed to extract contextual information by trying to define dependencies between different time stamps. [5]
RNN is made up of a series of recurrent layers that are modeled sequentially in order to locate the sequence to other sequences. RNN has a powerful ability to extract contextual information from a sequence.
The contextual cues in the core network, on the other hand, are stable and effective in achieving the data classification process. RNN is capable of processing sequences of any length.

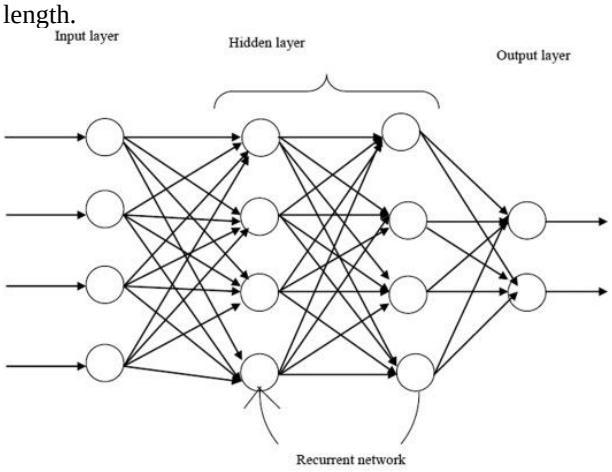



*Figure 8: - Recurrent neural network representation*

Early Feature Extraction is required by Machine Learning for features, and classification is conducted on it. Deep Learning, on the other hand, functions as a "black box," extracting and classifying features on its own.
The basic job is to classify the supplied image as face or non-facial, as shown in the diagram above. Machine learning requires an image attribute such as edges, color, shape, and so on, and does classification on its own.
Deep learning, on the other hand, extracts features and classifies them on their own. This given image is an example of how Convolutional Neural Networks work. Each layer of CNN takes a feature, and the Fully Connected Layer does categorization.
The major conclusion is that Deep Learning uses deep neural networks to extract data and classify itself.

As we can see the representation of deep learning in the given figure [4] below using the example of face recognition and its feature extraction and classification.

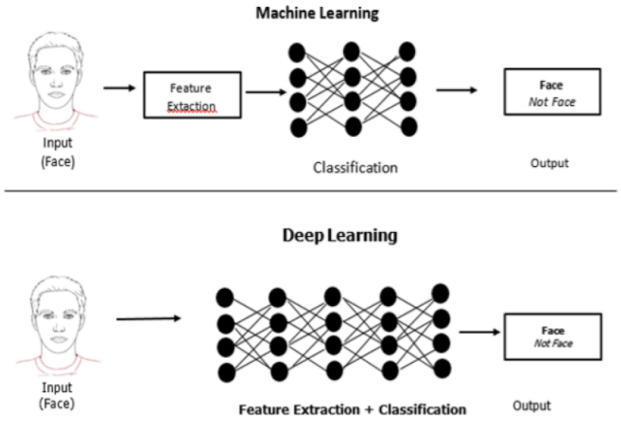



*Figure 9: Deep Learning Representation*

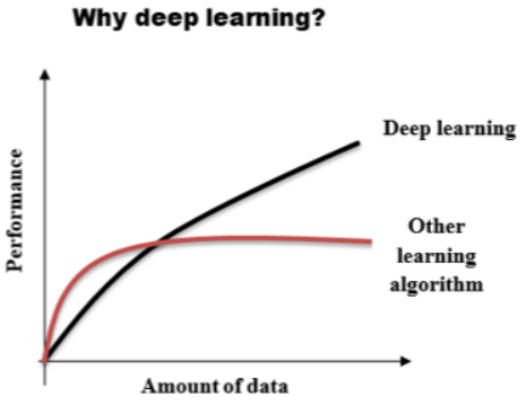



*Figure 10: Why Deep Learning?*

Handwritten script to digital text conversion is a very useful system as it is used for various purposes, especially in many real-life scenarios like criminal handwriting detection. This system can recognize and convert the handwriting of a criminal into digital text just by preprocessing and training the image of the handwritten script. This makes this model a very useful and simple tool in many ways.
Handwritten document recognition is one of the important research areas in the field of image processing and pattern recognition in recent years. It contains an automation process and improves the interface between humans and machines.
Many research works are focusing on reducing process time and having high recognition accuracy via new methods.
Handwriting recognition is a very tough task as: people have different handwriting and there are different characters like small letters and capital later, symbols likewise.
The handwritten document is converted to text format either by scanning a handwritten document or by writing with a special pen on the electrical surface. These two methods are respectively offline and online processes.

While convolutional neural networks have made tremendous progress in the field of handwriting recognition in recent years, obtaining decent results requires a large amount of training data. There is a way to increase the quantity of images without increasing labour by using general image processing methods, known as data augmentation, to prepare vast amounts of image data for training. However, using traditional data augmentation approaches, it is difficult to develop character representations that seem like those written by diverse persons and to solve challenges associated to a shortage of training data. In this research, we offer a method for obtaining the probability distribution of variables linked to character structure and using that distribution to generate character pictures of distinct handwritings.

By learning from character image data, the proposed method generates statistical character structure models built of probability distributions of strokes. It is feasible to construct character images of diverse handwriting samples without being impacted by the original images by creating strokes based on the probability distribution of each stroke and combining them into a character. In comparison studies with a convolutional neural network for handwritten character recognition, good results were obtained using both traditional data augmentation approaches and the suggested method simultaneously.

The most critical component in getting excellent recognition performance in character recognition systems is choosing a feature extraction strategy. For different representations of the characters, such as solid binary characters, character contours, skeletons (thinned characters), or gray-level subimages of each individual character, different feature extraction algorithms are proposed. The invariance qualities, reconstructability, and predicted distortions and variability of the characters are explored in terms of feature extraction approaches. The difficulty of determining which feature extraction method is best for a particular application is also highlighted. Once a few interesting feature extraction methods have been identified, they must be tested in order to determine which method is appropriate for the current application. We can undertand the process of feature extraction procedure by understading the figure [11] given below.

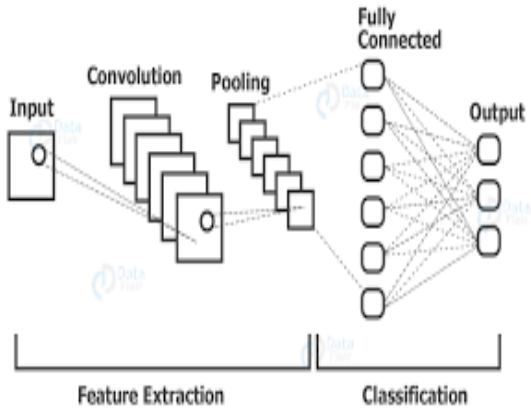



*Figure 11: Feature Extraction model*

**LITERATURE REVIEW**

The first research paper I reviewed was - Handwritten Character Recognition using Convolution Neural Networks in Python with Keras by Indiran H.P in the year (2020). [1]

This research paper aims to classify each individual handwritten word in order to convert handwritten text to a digital format. It shows how neural networks can be used to create a system that can recognize handwritten English alphabets. This model uses Python, Machine Learning, ML development IDE softwares and Computer Vision, CNN, Character Recognition, Classification, Deep Learning techniques to achieve the desired training of the machine learning model.

The dataset used in this model was constructed from a number of scanned document datasets available from the National Institute of Standards and Technology (NIST).
Based on the final findings with handwritten alphabets, the program attempted to improve by extracting characters from the handwritten image and then classifying each character separately in order to reconstruct the digital letter. The evaluation parameters used in this model are backpropagation and resilient propagation.
In this model I discovered that, when experimenting with different hyperparameters, keeping the number of epochs low was a good method. This method saves us a lot of time in training, but it also had certain drawbacks. This data augmentation strategy assisted in making the model more resistant to a few tiny but common details that may appear in our test set. This model used several preprocessing and data augmentation techniques on our dataset before training the models with it in order to make the data more compatible with the models and more robust to real-world circumstances. I identified the model's requirements and planned the solution using the knowledge I had described.

The technique for recognizing handwritten English alphabets and numerals was proposed and developed. It has been experimented with all English alphabets and numerical digits, as well as a variety of handwriting styles. The machine recognized the alphabets and numerals with an average accuracy of 82.5 percent, which is noteworthy and may be acceptable in some applications, according to the findings of the experiments. The machine was shown to be less accurate when classifying similar alphabets, although this misclassification of similar patterns may improve in the future, and a similar experiment can be conducted across a larger data set and with additional optimal network parameters to increase the machine's accuracy.
Using a language-based model, each of the possible final beam search candidate routes is given a penalty/benefit score, as well as their combined individual SoftMax probabilities, which represent the chance of the sequence of characters/words. If the SoftMax layer shows that the most likely candidate word is probably the most likely candidate word, and beam search is quite unlikely given the context so far, the model can correct itself accordingly.

The second research paper I reviewed was - Transfiguring Handwritten Text and Typewritten Text by Keerthana M., Hima Varshini P., Sri Thanvi K., Vijaya G., Deepa V. in the year (2022). [2]

This application can be used to convert handwritten text to editable text or typed text, as well as to convert typewritten text to handwritten text. To make the output more practical, the typed text is converted to handwritten text with randomization of handwritten character styles.
This model uses Python, PyCharm, GUI, PIL (Python Pillow) softwares and Transfiguring, Randomization, Pattern recognition, Python pillow (PIL), Tesseract, Validation techniques to get the desired prediction for the converted handwritten script.
The findings of this research paper show that the output of the converted handwritten script into digital text is shown as a GUI based image on the screens of the user. The model also displays the accuracy of the prediction and the possible error rate for the text. This model uses validation and model accuracy as evaluation parameters to determine the error rate and precision of the model.

The major advantage of this model is that the user controls the entire process through a graphical user interface (GUI). Processing and converting both handwritten and typewritten texts take less time and effort with this application.
The Disadvantage of this machine learning model is that the model focuses less on the accuracy and more on the graphical representation of the output and the text which makes the model less useful for real life purposes and it decreases the experimentation value for the model.

The final research paper I reviewed was - Handwritten Character Recognition Using Deep-Learning by R. Vaidya, D. Trivedi, S. Satra, and P. M. Pimpale in the year (2018) [3]

This research paper aims to develop a handwritten character recognition system based on image segmentation. This model used OpenCV for image processing and TensorFlow for neural network training in the system. This application used the Python programming language to create this system. The software used to develop this model are Python, OpenCV, Android, JAVA, and TensorFlow.

This model requires many different types of machine learning techniques to get the desired prediction as output and some of the core techniques used in this machine learning model are Neural networks, Handwriting recognition, Character recognition, Image segmentation, Smartphones, and Text recognition. The datasets used to construct this model were taken from google cloud services and amazon cloud services. Classification accuracy and character recognition accuracy are the two main evaluation parameters for this model and its training dataset.

The findings of this research paper show that the output of this model is an android application which allows the user to scan an image of the particular handwritten document and then recognize the text, finally converting it into digital format. The accuracy of the prediction of this model is about 92% and this can be further improved by using larger datasets and using deep learning algorithms on the extracted datasets which have least error rate in characters and more clean handwritten character data.

The advantages of this model are: The prediction accuracy of this model is very high which is 92%, highest accuracy amongst the other two models Transfiguring Handwritten Text and Typewritten Text [2] and Handwritten Character Recognition using Convolution Neural Networks in Python with Keras [1]. This model is also very portable and user friendly to use as it is the form of an android app which is also very accessible.
The disadvantages of this model are: This model uses cloud-based services as a dynamic dataset and it might not contain all the required data for handwritten character recognition and pattern recognition which makes the scope of the training model very small and limits the model to recognize only a few types of handwritten scripts.

**PROPOSED WORK**

Manually transcribing large amounts of handwritten data is a very difficult process that's bound to be fraught with errors and a lot of time. The recognition of handwritten text automatically is widely used in many applications, where to process huge handwritten text. This application is used to only transform handwritten script to digitized text but not to edit or manipulate it in any form. This model demonstrates the use of neural networks for developing a system that can recognize handwritten English alphabets. The CNN approach is used to accomplish this task: classifying words directly and character segmentation.
For the former, the Convolutional Neural Network (CNN) is used with various architectures to train a model that can accurately classify words.

The Real-Time Handwritten Text conversion With TensorFlow and Deep Learning model is a raw python and deep learning-based application which includes a few datasets and libraries to train a model.
A detailed This Real-Time Handwritten Text conversion With TensorFlow and Deep Learning model will be done.
Installation and hands-on experience on existing approaches of the This Real-Time Handwritten Text conversion With TensorFlow and Deep Learning model will be done. Relative pros and cons will be identified.
Various parameters will be identified to evaluate the proposed system.
A comparison of newly implemented approaches with existing approaches will be done.

The main purpose of this model is to transcribe and convert the handwritten script contained in scanned images or text files into a digitized form of text. For this application to function we will add our own training model and achieve a completely trained learning algorithm with less error rate.

We will use the (IAM) dataset and also (CTC) decoders to integrate them with word beam search.
The output for this model will also display the probability of the word being recognized including the word itself.

The proposed work for this research paper is aimed to carry out work leading to the development of an approach for the Real-Time Handwritten Text conversion With TensorFlow and Deep Learning.

The proposed aim will be achieved by dividing the work into the following:
Firstly, extract the (IAM) dataset from its website and download the other required datasets for converting Handwritten scrip to digitized text. The (IAM) dataset contains all the samples of handwritten script to train our machine learning model to recognize and convert the script as shown in the figure [12] below.

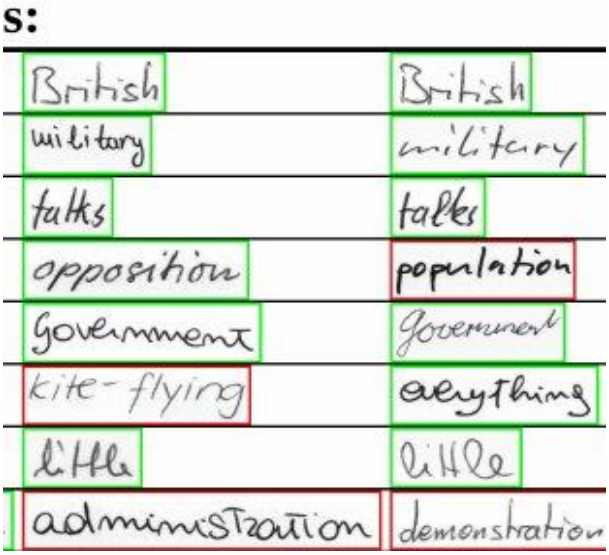



*Figure 12: IAM Dataset*

Download and import all the required libraries and datasets for training this machine learning model and also create modules for this application.
Now create and execute the 6 modules for this application to start predicting and converting the handwritten script into text. We also need to build a sample preprocessor for all the sample stored in the system to be processed. Now load all the data and integrate it by creating a module named (dataloader.py).
Now for the user to upload the image of the handwritten script and convert it into the digitized text we need to create a module using flask to integrate our machine learning model to a web page, which allow the user to upload the images of their handwritten script and convert them into digitized text.
Finally create the main module to run the application which will execute all the modules and trained model and give us the desired prediction output.
The objective of this model is to achieve the final output with the least amount of error rate in the recognition of handwritten script and print the final output as digital text.

This Real-Time Handwritten Text conversion With TensorFlow and Deep Learning model can be trained and executed in two ways:
The first way to is to train the model with all the samples of the handwritten script and use the (IAM) samples as the main training data. The training model consists of 66 epochs and each iteration of an epoch consists 1258 batches to train and this process of training might take about 7 – 8 hours depending upon the system and the GPU speed. After the training process is complete the trained model gives us an output of the predicted text after converting the saved image of a handwritten script as shown in the figure [13] below.

```
Validation character error rate of saved model: 8.654728%
Python: 3.6.4
Tensorflow: 1.8.0
Init with stored values from ../model/snapshot-24
Without Correction clothed leaf by leaf with the dioappoistmest
With Correction clothed leaf by leaf with the dioappoistmest
```

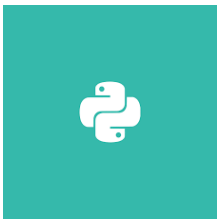
*Figure 13: Predicted output of the trained model*

This trained model has a character error rate of 8.6547% after performing validation on the saved model.

The second method to implement the Real-Time Handwritten Text conversion With TensorFlow and Deep Learning model is to let the user upload an image of their own handwritten script sample onto a web page created using flask and a module named (upload.py).
The user is only allowed to upload the images of their sample in the format of (.png) or (.jpg) onto the web page from their system memory. After uploading the image, the pretrained model predicts the text and converts the handwritten script into digital text and shows the converted text as output onto the user's screen.
The prediction of the converted text has a moderate accuracy and depends on the sample of the handwritten script uploaded as it is converting the image of handwritten script in real – time and very quick.


*Figure 14: Predicted and converted text of the user uploaded image*

The softwares and libraries used in the development of this machine model are as follows:

| Software Tool Used | Description | Logo |
|---|---|---|
| **Jupyter Noebook** | Jupyter Notebook is an open-source web-based application used to edit, create, run and share documents containing live code, visualizations, texts, and equations. The main supported programming languages are Julia, R, and Python. |  |
| **IAM Dataset: -** | An IAM handwritten dataset is a collection of multiple authors' handwritten text. They use this information to categorize authors based on their writing styles. |  |
| **Editdistance** | The Levenshtein distance, also known as distance editing, is one of the most well-known string metrics. editdistance determines how many substitutions and deletions are necessary to convert one string to another. |  |
| **Lmdb** | Lightning Memory-Mapped Database is the full name of this database. Using a single database file can also reduce the overhead of the data duplication/transfer process. |  |
| **OpenCV** | OpenCV is a Python open-source library for computer vision applications such as AI, machine learning, and facial recognition. Computer vision enables computers to perform human-like tasks with much the same effect as humans. |  |
| **TensorFlow** | Google created and released TensorFlow, a Python library for performing fast numerical calculations. This is a base library that can be used to build deep learning models directly or to wrap TensorFlow-based processes with a wrapper library. |  |
| **Matplotlib** | Matplotlib is a cross-platform data visualization and graphical plot library for Python and its numerical extension NumPy. |  |
| **Flask** | Flask is considered more Pythonic than the Django web framework. Because the similar Flask web application is more explicit in many cases, Flask is very straightforward to learn as a newbie due to the lack of boilerplate code required to get a small app up and running. |  |

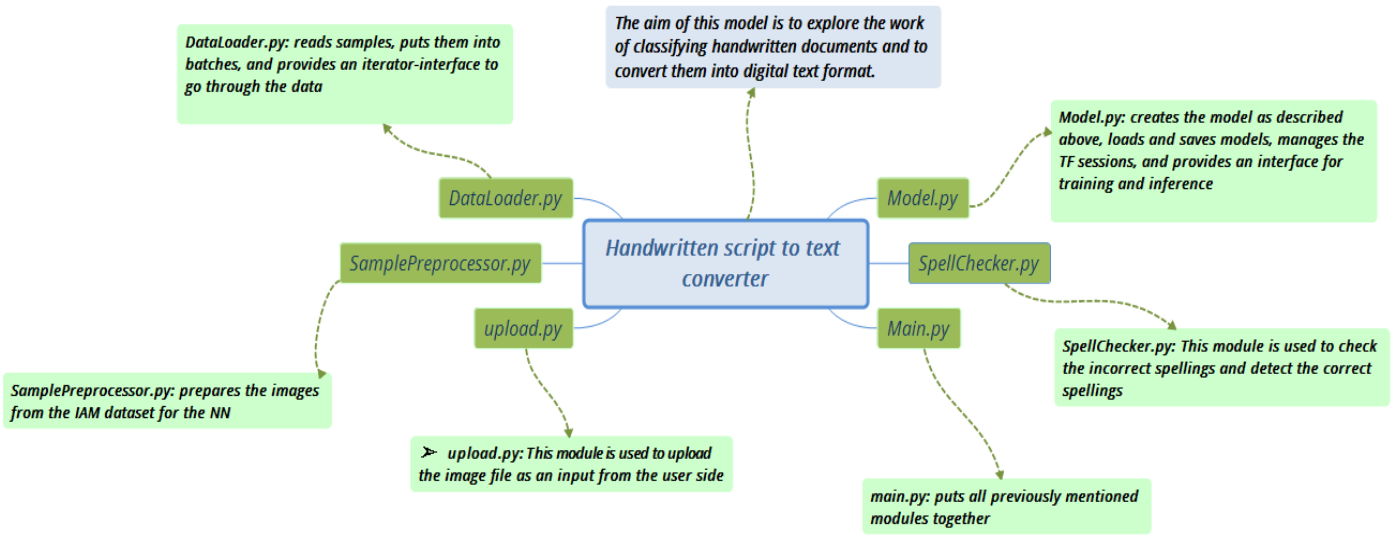



*Figure 15: System Flow Diagram*

The above figure [15] represents the system flow diagram of the machine learning model [Real-Time Handwritten Text conversion With TensorFlow and Deep Learning].

As we can see in the system flow diagram the main aim/goal of this model is to explore the work of classifying handwritten documents and to convert them into digital text format. This model consists of 6 modules and each module holds a code of python using different types of datasets and libraries used to predict the desired output of the text. The system flow diagram describes the purpose of each and every module and shows the importance of their own functions in order to train the machine learning model.

**METHODOLOGY**

The following methodology will be followed to achieve the objectives defined for the proposed research work: -

STEP 1 -> Firstly, we use CNN (Convolutional Neural Network) to analyze the Handwritten script image without pre-segmentation into words or characters. To train this model, use the CTC loss function.

STEP 2 -> Now, to extract the relevant elements from the handwritten script image, use a Convolutional Recurrent Neural Network.

STEP 3 -> The output from the CNN FC layer (512x100x8) is sent to the BLSTM, which is used to perform sequence dependence and time-sequence operations.

STEP 4 -> Now, we employ CTC LOSS to train the RNN, which eliminates the Alignment problem with handwritten documents, as each writer's alignment differs. We simply gave the image's text (Ground Truth Text) and the BLSTM output, and it calculates loss as -log("gtText"); the goal is to minimize the negative maximum likelihood route.

STEP 5 -> Finally, CTC decodes the output during Prediction by determining the feasible pathways from the given labels.

STEP 6 -> This model consists of three major steps:

Multi-scale feature Extraction --> Convolutional Neural Network 7 Layers

Sequence Labelling (BLSTM-CTC) --> Recurrent Neural Network (2 layers of LSTM) with CTC

Transcription --> Decoding the output of the RNN (CTC decode)

STEP 7 -> Now, we start building the 6 modules required to train the [Real-Time Handwritten Text conversion With TensorFlow and Deep Learning] model.

STEP 8 -> The first module is used to preprocess the samples extracted from the (IAM) dataset.

SamplePreprocessor.py: prepares the images from the IAM dataset for the NN

STEP 9 -> The second module is used to load all the datasets, libraries, samples, templates, images etc.

DataLoader.py: reads samples, puts them into batches, and provides an iterator-interface to go through the data.

STEP 10 -> The third module is used to structure all the data together and creates a model interface to train the model.

Model.py: creates the model as described above, loads and saves models, manages the TF sessions, and provides an interface for training and inference.

STEP 11 -> The fourth module is used to verify and correct the spelling mistakes present in the handwritten script data while training the model.

SpellChecker.py: This module is used to check the incorrect spellings and detect the correct spellings and also correct them using a library [autocorrect]

STEP 12 -> The fifth module is used to combine all the other modules together and integrate them to execute the training of the model.

main.py: puts all previously mentioned modules together

STEP 13 -> The sixth module is used to provide a web interface for the user to upload their own handwritten script sample in the form of an image.

upload.py: This module is used to upload the image file as an input from the user side.

I compared this research paper [Handwritten Character Recognition using Convolution Neural Networks in Python with Keras] to my model and I found out that there are a lot of differences between them. My model uses deep learning and CTC loss methods to train the model whereas this research paper describes that it uses: A neural network is made up of neurons that are connected to each other; at the same time, each connection in our neural network is associated with a weight that, when multiplied by the input value, determines the importance of this relationship in the neuron.

The activation function of each neuron determines the neuron's output. The activation function is used to add non-linearity to the network's modeling capabilities.
In this research paper, we'll go over a few different activation function options. The most genuine part of Deep Learning is training our neural network, which is learning the values of our parameters, and we can see this learning process in a neural network as an iterative process of learning. [1]

The second research paper which I compared my model to was Transfiguring Handwritten Text and Typewritten Text. My model was a raw python and machine learning based program but the second research paper [2] described that they have used android app development and GUI based software to develop their application. The methods used in the Transfiguring Handwritten Text and Typewritten Text are: This application can not only convert handwritten text to editable text or typed text, but it can also convert typewritten text to handwritten text. [2]
To make the output more practical, the typed text is converted to handwritten text with randomization of handwritten character styles.

Both transfigurations output text files are automatically translated into the corresponding audio format for easy output validation by the user and emailed to the user to avoid data loss. The user is in charge of the entire input and is guided by a graphical user interface (GUI). Processing and converting both handwritten and typewritten texts take less time and effort with this application.

I performed the comparison study on the research paper [Handwritten Character Recognition Using Deep-Learning] which was different from my model [Real-Time Handwritten Text conversion With TensorFlow and Deep Learning].
My model predicts and converts the handwritten script after training of the model and all the data with neural networks and TensorFlow including methods of deep learning. The output of my model can be shown in two ways: the first way is to directly run the trained model and see the converted text. The second way is to take the sample of the handwritten script image as input and convert into digitized text in real – time using a web page developed using flask in python.

The research paper [Handwritten Character Recognition Using Deep-Learning] uses: Handwritten text recognition is one of the areas of pattern recognition which is also known as text recognition. [3]
Pattern recognition is used to classify or categorize data or objects into one of several classes or categories.

The task of transforming a language represented in its spatial form of graphical marks into its symbolic representation is defined as handwriting recognition. Each script has a set of icons known as characters or letters, each of which has a basic shape.

Handwritten Character Recognition's goal is to correctly identify input characters or images, which are then analyzed by a variety of automated process systems. This system will be used to detect various types of writings.
Handwriting character recognition has advanced to the point where various types of handwritten characters, such as digits, numerals, and cursive scripts, can be found.

## CONCLUSION

This research work described mostly focuses on the working and implementation of my machine learning model [Real-Time Handwritten Text conversion With TensorFlow and Deep Learning].
I have shown the methodology used to create and execute this model with clear objectives and goals. I have reviewed three research papers mainly to take an idea of the current working systems of handwritten script recognition and conversion. I intend to perform more changes to my current working model and make it a lot more efficient and portable with less character error rate and more accuracy in prediction.

### Future Work: -

I intend to expand this research to a larger scale in the future so that different embedding models can be considered on a wider range of datasets.
No one will write on paper or with a pen in the future because everything is based on technology.
They wrote on touchpads in that scenario, so the inbuilt software could automatically detect the text they were writing and convert it to digital text, making searching and understanding much easier.

### Improvements: -

If we want to improve your recognition accuracy, follow the below steps:
Increase the size of the dataset by applying more (random) transformations to the input images.
In the input images, remove the cursive writing style (see DeslantImg)
Increase the size of the input (if the input of NN is large enough, complete text lines can be used)
Increase the number of CNN layers
Use token passing or word beam search decoding (see CTCWordBeamSearch) to constrain the output to dictionary words instead of LSTM.
Text correction: if the recognized word isn't in a dictionary, look up the closest synonym.

**REFERENCES**

[1] Indiran, H. P. (2020). Handwritten Character Recognition using Convolution Neural Networks in Python with Keras. Asian Journal for Convergence In Technology (AJCT) ISSN -2350-1146, 5(3), 123-131. Retrieved from https://asianssr.org/index.php/ajct/article/view/946

[2] Keerthana M., Hima Varshini P., Sri Thanvi K., Vijaya G., Deepa V. (2022) Transfiguring Handwritten Text and Typewritten Text. In: Ranganathan G., Fernando X., Shi F. (eds) Inventive Communication and Computational Technologies. Lecture Notes in Networks and Systems, vol 311. Springer, Singapore. https://doi.org/10.1007/978-981-16-5529-6_49

[3] R.Vaidya, D. Trivedi, S. Satra, and P. M. Pimpale, "Handwritten Character Recognition Using Deep-Learning," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 772-775, DOI: 10.1109/ICICCT.2018.8473291.

[4] C. Bahlmann, B. Haasdonk, H. Burkhardt., "Online Handwriting Recognition with Support Vector Machine – A Kernel Approach", In proceeding of the 8th Int. Workshop in Handwriting Recognition (IWHFR), pp 49- 54, 2002

[5] Hannun, A. (2017). Sequence Modeling with CTC. Distill, [online] 2(11). Available at: https://distill.pub/2017/ctc/.

[6] Bluche, T., Louradour, J. and Messina, R. (2016). Scan, Attend and Read End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. arXiv:1604.03286 [cs].

https://arxiv.org/abs/1604.03286