International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
DOI: https://doi.org/10.17762/ijritcc.v11i4.6376
***Article Received**: 28 January 2023 **Revised**: 22 February 2023 **Accepted**: 02 March 2023*

Handwritten Digits and Optical Characters Recognition

Kartik Sharma, S.V. Jagadeesh Kona, Anshul Jangwal[1], Dr. Aarthy M, Dr.Prayline Rajabai C, Dr. Deepika Rani Sona[2]

[1]School of Electronics Engineering, Vellore Institute of Technology, Vellore,India
kartiksharma.ksha@gmail.com, jagadeeshkona20@outlook.com, anshuljangwal@gmail.com
[2]School of Electronics Engineering, Vellore Institute of Technology, Vellore,India
aarthy.m@vit.ac.in, prayline.c@vit.ac.in, deepika.rs@vit.ac.in

Abstract- The process of transcribing a language represented in its spatial form of graphical characters into its symbolic representation is called handwriting recognition. Each script has a collection of characters or letters, often known as symbols, that all share the same fundamental shapes. Handwriting analysis aims to correctly identify input characters or images before being analysed by various automated process systems. Recent research in image processing demonstrates the significance of image content retrieval. Optical character recognition (OCR) systems can extract text from photographs and transform that text to ASCII text. OCR is beneficial and essential in many applications, such as information retrieval systems and digital libraries.

Keywords- Optical Character Recognition, CNN model, Handwritten character recognition, hidden layer.

I. INTRODUCTION

One major application of pattern recognition is the recognition of handwritten text. To label or classify data or objects into one of the classes or categories is the goal of pattern recognition. There are now many different types of handwritten characters, including digits, arithmetic, cursive writing, symbols, and fonts in both English and other languages. Handwriting has also advanced to a more complex level with many applications where it is essential to deal with massive amounts of handwritten data, such as the understanding of amounts on bank checks, document analysis, and signature verification, can benefit greatly from the automatic recognition of handwritten text.

The OCR system assists in the automation of the workplace, which saves a lot of time and human work. It also essentially eliminates the keyboard interface between people and machines. An OCR system's accuracy can occasionally be influenced by text pre-processing and segmentation techniques. The degree of difficulty in text extraction depends on factors such as text size, intricacy of image background, orientation, and text style.

II. MOTIVATION

Number of Handwriting character recognition systems are in place today, being developed using a variety of approaches and methods. Few of them, meanwhile, concentrate on neural networks. In comparison to other computing methods, using neural networks to recognize handwritten letters is more effective and reliable. The project describes the development process, architecture, and design of the Handwriting Character Recognition system, as well as testing and development outcomes. The objective is to show how well neural networks recognize handwriting characters.

Tesseract has been used in several applications to create real-world scenarios involving documents, data exploration and archiving from pictures, efficient image database manipulation, language processing, and numerous other processes. In our project we aim to make a vernacular language model that is highly accurate and involves minimal complexity due to which it is easy to use and also provides a mechanism of storing the extracted information from an image. Although Tesseract helps with more accurate data extraction from photos, it also has several drawbacks, such as the over-segmentation of certain characters and the inadequate segmentation or rejection of cursive word segments, which is what our project tries to fix.

III. OBJECTIVES

Primary objective is to introduce a system which uses the MNIST dataset and observe the variation of the accuracies to classify the digits using deep neural networks. This research revolves around the study of developing a neural network-based expert system for handwriting character identification and to assess the differences in accuracy between a simple CNN model and a model with an additional hidden layer. Our main goal is to design a system that uses effective technology to identify handwritten characters and words from image media in order to address the issue of accuracy in handwriting character recognition systems.

The OCR technology currently deployed in the applications such as invoice management travels document processing,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
DOI: https://doi.org/10.17762/ijritcc.v11i4.6376
***Article Received**: 28 January 2023 **Revised**: 22 February 2023 **Accepted**: 02 March 2023*

document management etc. It is not multilingual in nature. The current technology even if it provides multilingual support has low accuracies and complex algorithms, due to which chances of mismanagement are very high. Our project aims to reduce these complexities while making it a highly accurate multilingual technology

IV. BLOCK DIAGRAM

The system pre-processes the data using Tensorflow and Keras, by using MNIST dataset as a training sample. Then, to evaluate the accuracy differences between CNN's classification of handwritten digits using a single layer at first and a hidden layer later on. (Fig. 1)

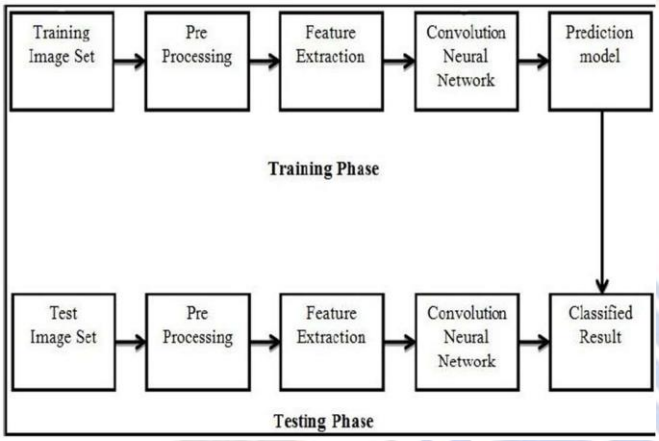

***Figure 1-** Data Testing and training phases*

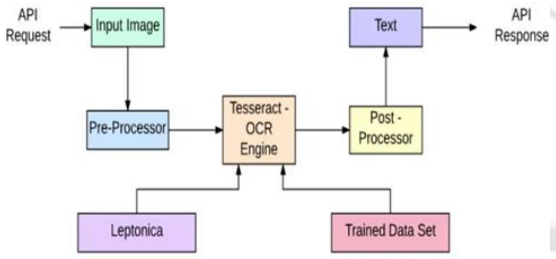

***Figure 2-** OCR process flow*

As the block flow diagram suggests, (Fig. 2) Using the OCR method, text may be extracted from a picture and translated into an editable format step-by-step. Digitization, pre-processing, text localization, text splitting, categorization, and character recognition are the fundamental phases. There can be variations in the text due to the size, color, background image, alignment etc. but all of that can be rectified by using the appropriate pre-processing techniques. Additionally, we make use of the Leptonica library, an open-source library of software, offers tools that are generally beneficial for applications involving computer vision techniques and image processing.

V. SOFTWARES USED

Following software were used for the project:

A) **Keras** is a strong neural network API with an emphasis on user simplicity, rapid development, versatility, and adaptability. We can immediately begin designing and training a neural network because it is compatible with frameworks for deep learning like Tensorflow, Theano, and CNTK. Deep learning can be rapidly and easily developed and tested with Keras, and it runs smoothly on both CPU and GPU.

B) The machine learning software library **Tensor Flow** is open-source and free. Deep neural network training and inference are given considerable attention, whilst it being a vast and adaptable ecosystem of resources, services, and packages that offer processes with high-level APIs.

C) A library of programming functions primarily geared at real-time computer vision is known as **Open CV (Open-Source Computer Vision Library).** The technique of understanding videos and photos, the way they are stored, and how to change and extract data from them is known as computer vision.

D) **Pytesseract** is the Tesseract-OCR Engine's wrapper. It can read any image types provided by the Pillow and Leptonica imaging libraries, namely jpeg, png, gif, bmp, tiff, and others, making it helpful as an independent invocation tool to tesseract.

VI. METHODOLOGY

A) HDR:

- First, we import the required libraries.
- We can use the dataset's existing, well stated data-train and test datasets. The images are all pre-aligned due to the fact that each one is merely a hand-drawn digit, is a square of 28X28 pixels in size, is grayscale, and only includes one digit. The dataset is loaded, and the MNIST data is partitioned into a train set and a test set. The MNIST dataset can be easily loaded using the Keras deep learning toolkit.
- Grayscale values for pixels range from 0 to 255. When deploying neural network models, scaling the input values is often a smart idea. We can rapidly normalize the pixel values to the range of 0 and 1 by dividing each value by the maximum of 255 because the scale is widely understood and well controlled.
- The pixel vector serves as input and we use Keras Sequential Model to create the model and sigmoid function acts as the activation function. The model is fit over 5 Epochs.
- To visualize the accuracies and predictions results in a graphical way with the concept of confusing matrix.
- To Add a Hidden layer in our previous model and to compare the resulting accuracies and efficiency in a repeated manner.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
DOI: https://doi.org/10.17762/ijritcc.v11i4.6376
***Article Received**: 28 January 2023 **Revised**: 22 February 2023 **Accepted**: 02 March 2023*

B) OCR:

STEP-1(Taking the input images and Text Localization): Amplifying the text area by deleting non-text areas is a step in the localization process. The text in an image has the characteristic that all of the characters cluster together near to one another. A structural dilation technique can be used to group text pixels together and remove pixels that are too far away from the candidate text area after taking this aspect into considerations.

STEP-2(Carrying out the process of Text Segmentation): Identification of every glyph is being done at this point (Glyphs are basically units that represent one or more characters). Due to the changes in paragraphs, words in a line, characters in a word, skew, slant, size, and curves, segmenting handwritten characters into multiple zones, such as upper, middle, and lower zone, and characters, the process is more challenging than it is for regular printed documents. In this procedure, binarized areas' horizontal histogram profiles are evaluated to separate them into text lines, and then we separate the letters and words inside each text line using a vertical histogram.

Step-3(Classification): At this point, the OCR engine is being trained by being given some sample data pertaining to the data that is needed to extract and recognise the presented document but is not yet existing in the system. The character recognition from the retrieved text picture will be done using the feature values that were taught.

Step-4(Character Recognition): The process of extracting text from images ends with character recognition. At this point, we have the character arrays discretely segmented and positioned within the pre-processed image. In this stage, segmented characters are smoothed and scaled before being compared to the training set data that has been stored and the OCR engine's closest match is shown.

VII. RESULTS

A) HDR:

```
model = keras.Sequential([
    keras.layers.Dense(10, input_shape = (784, ), activation = 'sigmoid')
])

model.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)
model.fit(X_train_flattened, y_train, epochs = 5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 5s 79us/sample - loss: 0.4875 - accuracy: 0.8772
Epoch 2/5
60000/60000 [==============================] - 4s 72us/sample - loss: 0.3065 - accuracy: 0.9154
Epoch 3/5
60000/60000 [==============================] - 4s 71us/sample - loss: 0.2850 - accuracy: 0.9214
Epoch 4/5
60000/60000 [==============================] - 4s 71us/sample - loss: 0.2748 - accuracy: 0.9244
Epoch 5/5
60000/60000 [==============================] - 4s 72us/sample - loss: 0.2673 - accuracy: 0.9261

<tensorflow.python.keras.callbacks.History at 0x254ded7af08>
```

***Figure 3**- Test Results of HDR*

- Evaluation of the accuracies is always one the test dataset. (Fig. 3) The result came to be the same as during the creation of our simple neural model.
- The model is fit over 5 Epochs.
- The minimum training accuracy is found to be 87.72% with the total loss of 0.4875.
- The maximum training accuracy is found to be 92.61 % with the total loss of 0.2673.

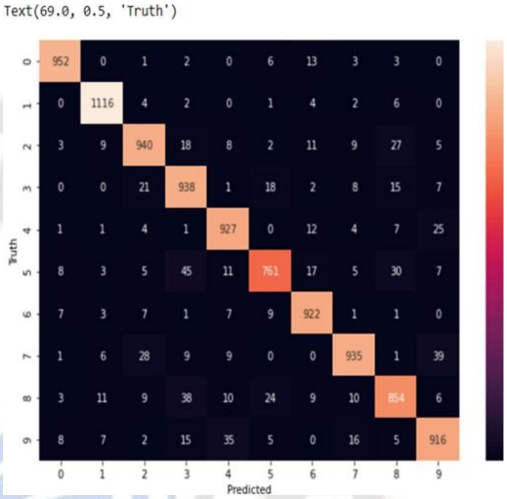

***Figure 4**- Visualization of a confusion matrix from the predictions of CNN Model*

Addition of a hidden layer-

The basis of CNN's architecture is its hidden layers of neural network. A sequence of convolution, pooling, and activation functions are applied during the extraction of features. At this step, the distinguishing characteristics of handwritten digits are discovered.

```
model = keras.Sequential([
    keras.layers.Dense(100, input_shape = (784, ), activation = 'relu'),
    keras.layers.Dense(10, activation = 'sigmoid')
])

model.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy']
)
model.fit(X_train_flattened, y_train, epochs = 5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 6s 94us/sample - loss: 0.2899 - accuracy: 0.9194
Epoch 2/5
60000/60000 [==============================] - 5s 90us/sample - loss: 0.1309 - accuracy: 0.9616
Epoch 3/5
60000/60000 [==============================] - 5s 86us/sample - loss: 0.0942 - accuracy: 0.9717
Epoch 4/5
60000/60000 [==============================] - 5s 87us/sample - loss: 0.0754 - accuracy: 0.9772
Epoch 5/5
60000/60000 [==============================] - 5s 88us/sample - loss: 0.0606 - accuracy: 0.9816

<tensorflow.python.keras.callbacks.History at 0x254e0e40e08>
```

***Figure 5**- Test results after adding hidden layer*

- A SoftMax is incorporated with the output layer to output digits from 0 to 9 through 9.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
DOI: https://doi.org/10.17762/ijritcc.v11i4.6376
***Article Received**: 28 January 2023 **Revised**: 22 February 2023 **Accepted**: 02 March 2023*

- The CNN is fit overall for 5 epochs and training accuracies and test loss is found.
- The accuracies and losses will vary each time if we give a new run to our code. (Fig. 5)
- Now, the minimum accuracy is found to be 0.9215 for the epoch 1 and loss is 0.2899.
- The maximum accuracy is found to be 0.9814 for the epoch 5 and loss is 0.0606.

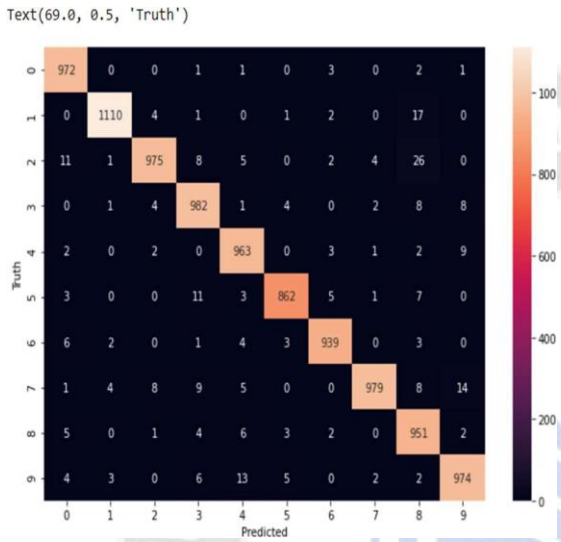

***Figure 6**- Graphical Visualization of predictions after addition of Hidden layer*

The maximum overall performance of the network increased from 92.61% to 98.14%. Moreover, the overall loss varies from 0.2673-0.4875 to 0.2899 to 0.0606. Hence, the proposed method of CNN is more efficient with the addition of hidden layers as compared to with no or less hidden layers for digit recognition.

B) OCR:

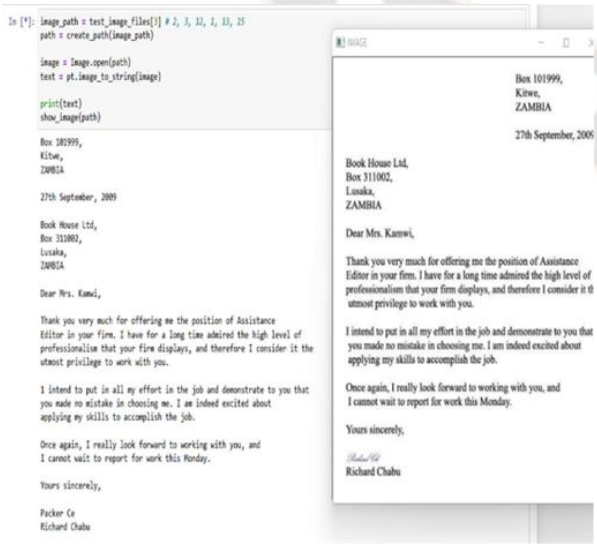

***Figure 7**- Extracting text from an Image: Simple*

***Figure 8**- Extracting text from an Image: Specifying a language*

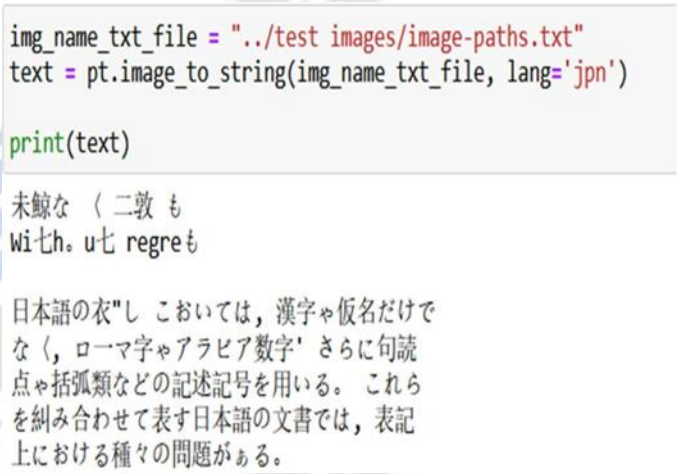

***Figure 9**- Extract text from an image: Multiple images once*

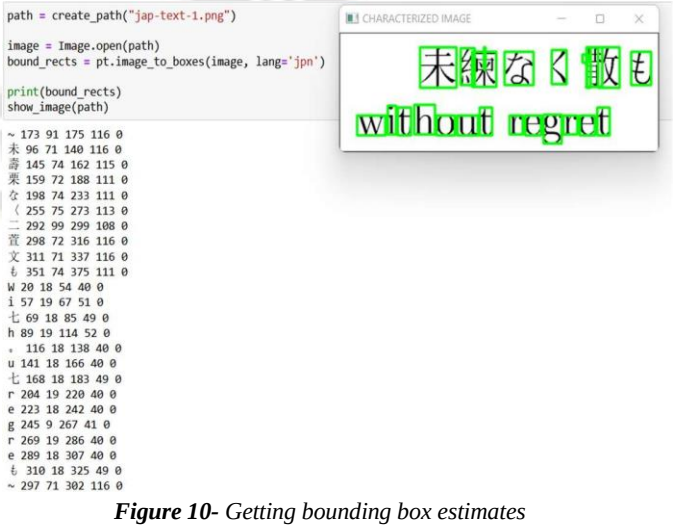

***Figure 10**- Getting bounding box estimates*

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
DOI: https://doi.org/10.17762/ijritcc.v11i4.6376
***Article Received**: 28 January 2023 **Revised**: 22 February 2023 **Accepted**: 02 March 2023*

VIII. CONCLUSION

We used Keras and Tensorflow to create a convolutional neural network for classifying handwritten digits. We also assessed the model's performance and developed a benchmark for a classification test. We further added a hidden layer to our model and compared the accuracies and predictions of both the models. The addition of a hidden layer in the neural network model makes the model more accurate and efficient. The overall performance of the model was enhanced and the errors in the predictions reduced to some extent. Recognition of handwriting is very crucial to aid automation and reduce human efforts. In the sectors of banking, accountancy, education, and other industries, handwriting recognition can play a significant role in establishing significantly decreased costs. So that the most accurate Mode with the lowest likelihood of errors can be used in diverse applications of handwritten digit identification and optical character recognition, we have compared the correctness of all models based on their efficiency (accuracy in this case).

As for the OCR we have concluded that the nature and quality of the text data heavily influences the OCR systems' accuracy, yet it has been determined that the Tesseract is a very effective OCR system. We were able to achieve an accuracy of around 93% on average. We are able to scan and comprehend words with a level of accuracy never previously achieved thanks to contemporary methods like convolution neural networks. With the use of publicly available GPU resources and computation time, we aim to achieve the best level of accuracy that is feasible for a lightweight, simple to train model. The top-performing CNN architecture is to be used to construct our model.

VIII. FUTURE SCOPE

For the OCR and the HDR clean text images with less distortions are required due to which accuracy remains high but clear text images might not be always available for instance in the case of historical texts, we might not get the exact input required for an accurate result so it becomes important to pre-process the digital images and then work with them, which is one area of improvement that can be considered. Also our algorithm is currently able to detect spaced letters and not the connected ones so it is one area that can be worked upon in the future. Future research can examine deep learning optimization and use it to solve challenging image recognition issues.

Building a real-time classifier and an associated interface that will accept user input and instantly recognize and convert it to a digit can be another future project. Following can be the future advancements of this project:

A) Adding more hidden layers and increasing the number of epochs to improve accuracy.

B) Detecting custom handwritten digits.

REFERENCES

[1] R. Anil, K. Manjusha, S. S. Kumar, and K. P. Soman, "Convolutional Neural Networks for the Recognition of Malayalam Characters," in Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014, vol. 328, Springer International Publishing, 2015, pp. 493–500.

[2] Y. Tang, L. Peng, Q. Xu, Y. Wang, and A. Furuhata, "CNN Based Transfer Learning for Historical Chinese Character Recognition," in Proceedings of the 12th IAPR Workshop on Document Analysis Systems (DAS), 2016, pp. 25–29.

[3] Cho Yee Phy, "Tensorflow input image by tfrecord," 2017. [Online]. tensorflow_input_image_by_tfre cord. [Accessed: 4-Mar2018].

[4] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid- level Image Representations Using Convolutional Neural Networks," in Proceedings of the 2014 Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1717–1724

[5] L. Hvass, "Tensorflow tutorial 06 CIFAR- 10," 2016. 3BXfw_1_TF4. [Accessed: 4- Mar- 2018].

[6] Yellapragada SS Bharadwaj, Rajaram P, Sriram V.P, Sudhakar S, Kolla Bhanu Prakash (2020, April). Effective Handwritten Digit Recognition using Deep Convolutional Neural Network. International Journal of Advanced Trends in Computer Science and Engineering, 9(2), 1335-1339.

[7] Vijayalaxmi R Rudraswamimath, Bhavanishankar K (2019, June). Handwritten Digit Recognition using CNN. International Journal of Innovative Science and Research Technology, 4(6), 182-187.

[8] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," arXiv preprint arXiv:1702.05373, 2017.

[9] Haider A. Alwzwazy, Hayder M. Albehadili, Younes S. Alwan, Naz E. Islam (2016, February). Handwritten Digit Recognition Using Convolutional Neural Networks. International Journal of Innovative Research in Computer and Communication Engineering, 4(2), 1101- 1106.

[10] Fabien Lauer, Ching Y. Suen, G´erard Bloch (2007). A trainable feature extractor for handwritten digit recognition. Journal Pattern Recognition, Elsevier, 40(6), 1816- 1824.