International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 05 | May 2023 www.irjet.net p-ISSN: 2395-0072

Medical Prescription Scanner using Optical Character Recognition

Swapnil Ghosh[1], Vedang Pokharkar[2], Sanjeevani Parida[3], Saiem Pathan[4], Prof. Palash Sontakke[5]

[1,2,3,4] *Undergraduate Students, Dept. of Information Technology, MIT SOC, MIT Art Design Technology University, Pune, India*
[5]*Professor, Dept. of Information Technology, MIT SOC, MIT Art Design Technology University, Pune, India*

Abstract - *We aim to assist people in the process of prescription management and referencing. For any person, the process of going to the hospital involves receiving a single or multiple prescriptions, said prescriptions aren't always standardized and vary from hospital to hospital, some might be handwritten and others might be digital. This makes the process of handling these documents rather tedious and complicated as most people either lose track or are forced to maintain a file for these documents. This is the problem that we aim to solve, where using deep learning, OCR (Optical Character Recognition) and handwriting recognition, we can digitize and store prescriptions thus liberating the user from their hardship by introducing a medical prescription manager that stores all the prescriptions. All the user has to do is scan the document once using the camera of their device. How does one achieve this? By the implementation of deep learning techniques and algorithms such as 'xyz', we can effectively identify characters and whitespaces whether handwritten or digital and using a predefined vocabulary of medicinal terms accurately predict the terms such as name of medicine, dosage, etc. thus proving itself useful for the patient as well as doctors and pharmacists.*

Key Words: OCR, handwriting, medicine, prescription, scanning

1. INTRODUCTION

Every visit to the doctor's office involves the generation of one or multiple prescriptions. For those who frequent the doctor's office often, the prospect of keeping track of yet another prescription is dreadful as it needs to be maintained for a long duration to avoid any side effects in case new medicines that have been prescribed generate an unforeseen reaction. From the point of view of the doctor or the hospital at large, they also need to maintain comprehensive records of the hundreds if not thousands of patients that visit them, this makes it exceptionally tedious to maintain all these records. Prescriptions in general are a rather archaic solution, while some hospitals have transitioned into using digital prescriptions, they still need to print them out while solving the problem only in a limited capacity, that of the indistinguishable handwriting of the doctors while still contributing to the paper trail. As OCR is capable of identifying both handwritten and digital writings, it solves both problems at once.

2. MOTIVATION

Since time immemorial, the relationship between doctor and patient has included a prescription as an intermediary for the benefit of the patient. It is through this prescription that doctors communicate the remedy of the patient's ailment to not just the patient but also the pharmacist who is going to provide those medicines and to other doctors that the patient may ever visit. The problem arises when someone who needs to visit the doctor frequently has to keep track of their medicines and allergies. The higher the number of visits, the greater the volume of the paper trail that needs to be maintained. It slowly becomes a very cumbersome task to store all these documents as they become the backbone of the medical history and record of the patient. Even in the modernized 21st century most establishments still rely on traditional methods to generate these prescriptions. To provide a solution, we can use the help of OCR.

3. METHODOLOGY

3.1 GATHERING DATA

The first step is to collect/use a large dataset of medical prescriptions that includes a variety of handwriting styles, a dictionary of medical terminology, and prescription formats. This dataset will be used to train and test the OCR model.

3.2 PRE-PROCESSING

To ensure accurate recognition of medical prescriptions using OCR technology, the collected data must undergo pre-processing to eliminate unwanted elements such as background noise, stains, and irrelevant marks. The pre-processing step includes various techniques to ensure that the prescription is aligned correctly and ready for accurate OCR recognition. This step is crucial in enhancing the quality of the image and reducing noise, leading to improved OCR performance.

Some of the basic techniques used are:

1) Normalization: It is a pre-processing technique in Handwritten OCR that aims to standardize the size, orientation, and position of the input image. This technique helps to make the input images more consistent and uniform, which can lead to improved OCR accuracy. The process of

normalization involves scaling, rotation, and translation, which corrects any inconsistencies in the input image.

2) Grayscaling: It is a technique where the image in converted into different shades of gray from different color formats. This helps to increase the contrast between the background and the actual handwritten data.

3) Binarization: It is a technique where the image in converted into different shades of gray from different color formats. This helps to increase the contrast between the background and the actual handwritten data.

One may also use advanced techniques like:

4) Sobel Edge Detection: This technique detects the edges in an image and can be used to segment the text from the background.

5) Contrast Limited Adaptive Histogram Equalization (CLAHE): This technique enhances the contrast of an image by redistributing the pixel intensities in a way that improves local contrast.

6) Absolute Difference (absdiff): It calculates the absolute difference between two images and can be used to highlight the changes between two versions of an image. This can be useful for detecting changes in text between scanned pages.

3.3 IMAGE SEGMENTATION AND STANDARDIZATION

The process of segmenting handwritten data using OCR is vital in optical character recognition systems. Segmentation involves identifying and separating individual characters or words from an image, which OCR algorithms can then recognize. Handwritten data can be difficult to segment due to the various handwriting styles, character shapes, sizes, and inconsistencies in strokes and line thicknesses. Features like character shape, size, and orientation are extracted after segmentation. These features are used to create a representation of each character, which is then classified using machine learning algorithms. The classification process assigns a label or character to each segmented image region based on its features. The steps for Image segmentation include:

1) Region of Interest (RoI) Extraction: In the process of extracting hand-written text from a prescription document, regions of interest (RoIs) are identified as the blocks containing the text. By performing this step, all the RoIs in the prescription document can be obtained.

2) Contour Detection: In image segmentation for OCR, contour detection is a technique used to identify the boundaries of objects or shapes in an image, which can be useful for identifying individual characters or words in a handwritten medical prescription. This involves connecting points along the boundary to form a continuous curve or contour, which can be used to separate the object from the background. There are several algorithms available for contour detection, including: Sobel, Canny, and LoG, and the choice of algorithm depends on the specific needs of the OCR system and the image being processed. Using these, we can define a boundary for each alphabet.

3) Cropping and Alphabet Recognition: Cropping is the process of identifying the boundaries of each character in the image and extracting it as a separate image. This is typically accomplished using techniques like contour detection to locate the edges of the characters and then selecting the area of the image contained within those boundaries. Once each character has been isolated, alphabet recognition techniques can be used to identify the specific letter or symbol represented by each character. These methods use pattern recognition algorithms to compare the shape and structure of the character to a database of known characters and determine the closest match. Various techniques are available for alphabet recognition, including neural networks, support vector machines (SVM), and template matching. These methods can recognize a wide range of handwritten characters, including letters, numbers, and symbols.

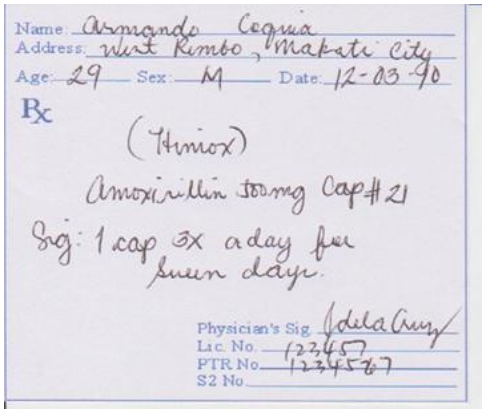

Fig -1: Example of a Handwritten Prescription

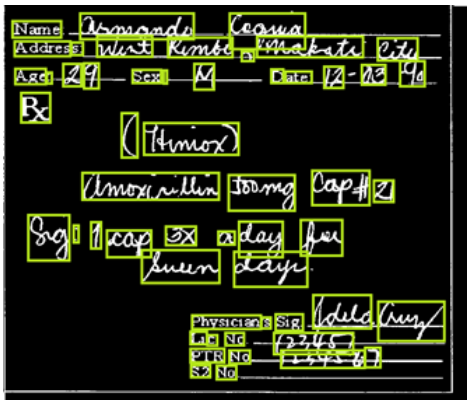

Fig -2: Image Segmentation using Gap Classifiers, Grouping, etc. along with boundaries

Fig -3: Character Recognition

Standardization: Since a medical prescription has various elements in it, it is important for us to understand the context of each and every line of prescription. Usually, the name of the clinic and doctor is on the top of the prescription. Next to it is the date of the prescription. The patient's name, age and gender is below it. Below that comes the Superscription, the Inscription and the Subscription respectively. The signature on the bottom right can be ignored for our case.

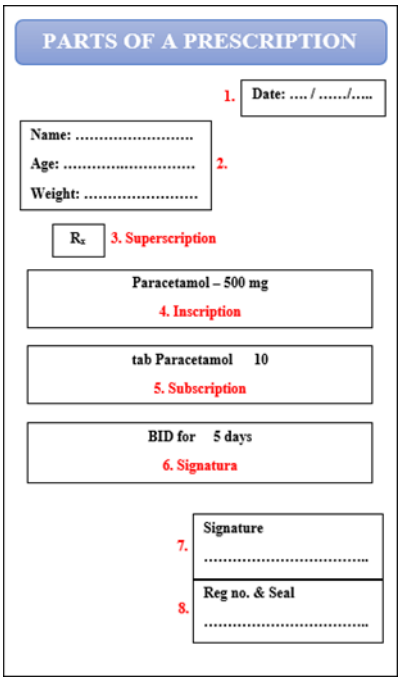

Fig -4: Prescription Format

3.4 TRAINING AND TESTING MODEL

Data preparation is the first step in training and testing an OCR model. To create a robust dataset, it is essential to collect and organize a large and diverse set of handwritten text samples that accurately reflect the range of handwriting styles and variability that the system will encounter in real-world applications. It is crucial to ensure that the dataset is representative and balanced to prevent bias in the OCR model. After the features have been extracted, it is time to choose an appropriate machine learning algorithm and train it on the dataset. There are several types of machine learning algorithms that can be used for OCR, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and support vector machines (SVMs). The selection of the appropriate algorithm depends on the specific use case and the nature of the data. During the training phase, the OCR model learns to identify and classify the extracted features based on the patterns in the dataset. The goal is to develop an accurate and effective OCR system that can reliably recognize and classify handwritten characters. Once the OCR model has been trained, it is time to evaluate its performance using a separate dataset of handwritten text samples. This allows the accuracy and effectiveness of the system to be assessed, and any errors or issues to be identified and addressed. Evaluation metrics such as precision, recall, and F1 score are commonly used to evaluate the performance of the OCR system. To ensure the accuracy and reliability of the OCR system, it is important to continually refine and optimize the model based on feedback from real-world use cases. This might involve adjusting the feature extraction process, fine-tuning the machine learning algorithm, or collecting additional data to expand the dataset.

- Using Rectified Linear Unit (ReLU) is a very useful suggestion for Training and Testing models. ReLU (Rectified Linear Unit) is an activation function used in deep learning models, including those used for OCR. In OCR, the activation function is applied to the output of the convolutional and pooling layers in a neural network. ReLU is a non-linear activation function that replaces any negative value with zero and leaves positive values unchanged. This non-linearity helps the model to learn more complex representations of the input data. During the training phase, the OCR model learns to adjust the weights and biases of the neural network to minimize the difference between its predicted output and the actual ground truth. The activation function, including ReLU, is applied to the output of each layer of the network to introduce non-linearity and enable the network to learn more complex features. When testing the OCR model, the input image is passed through the network, and the output is a set of predicted characters. The ReLU activation function ensures that the output of the network is non-linear and capable of capturing complex relationships between the input image and its corresponding text. This boosts the overall accuracy of the model.

Models we used are:

•Seq2Seq

•Seq2Seq Using CNN

•Connectionist temporal classification (CTC)

•Bidirectional Recurrent Neural Network (BiRNN) along with CNN

3.5 IMPLEMENTATION

In this phase, we integrate the trained OCR model into a software application or system that can automatically scan and recognize text from handwritten medical prescriptions. Here are some steps to consider during the implementation phase:

1) Programming Language: Choose a programming language that is suitable for implementing the OCR model

2) OCR Library: Choose an OCR library that can integrate with the chosen programming language. Tesseract OCR, OpenCV OCR, and GOCR are some of the popular OCR libraries available.

3) UI: Develop a user interface for the application that allows users to upload images of handwritten medical prescriptions and view the recognized text.

4) Integration: Integrate the trained OCR model into the application using the selected OCR library. This involves passing the image through the OCR engine and obtaining the recognized text output.

5) Post-processing: Perform post-processing of the recognized text to remove any errors or inconsistencies.

6) Validation: Validate the output of the OCR system to ensure that the recognized text accurately reflects the hand-written medical prescription.

We will first be using Python along with Tensorflow and OpenCV and then move on to build an Android application.

4.TECHNOLOGY USED

4.1 TENSORFLOW

TensorFlow is a popular open-source machine learning framework developed by Google for building and training deep learning models. It offers a range of tools and libraries that allow developers to build and train various neural network architectures for different tasks, including object detection, natural language processing, and handwriting recognition. For handwritten OCR, TensorFlow can be used to build and train a deep learning model that can recognize characters and words from scanned handwritten documents. The TensorFlow library provides APIs for image processing, data augmentation, model creation, training, and evaluation. In addition to these APIs, TensorFlow provides a range of other tools and libraries that can be used for OCR tasks. For example, the TensorFlow Object Detection API can be used to detect and segment individual characters or words in a handwritten document, while the TensorFlow Lite library can be used to deploy the OCR model on mobile devices or embedded systems.

4.2 OPENCV

OpenCV is a software library that provides computer vision and image processing functions. It is widely used in OCR systems for handwritten medical prescription scanners to perform image processing tasks such as image segmentation, filtering, and feature extraction. With OpenCV, medical prescription images can be pre-processed before being fed into the OCR engine to recognize the text accurately. OpenCV can also be used to build OCR engines for recognizing handwritten medical prescription text. By using deep learning techniques such as convolutional neural networks (CNNs), OpenCV can be trained on large datasets of handwritten medical prescriptions to accurately recognize the characters. In addition to pre- processing and OCR, OpenCV can also be used for post- processing tasks such as text layout analysis and character correction, which can improve the accuracy of OCR systems.

4.3 KERAS

Keras is a high-level neural networks API written in Python and designed to run on top of TensorFlow, making it easier to build and train deep learning models. Keras can be used to build and train a deep learning model to recognize characters in the prescription image. Keras provides various pre-trained models which help the user in further improving the models the user is designing.

model	latency	precision	recall
AWS	719ms	0.45	0.48
GCP	388ms	0.53	0.58
keras-ocr (scale=2)	417ms	0.53	0.54
keras-ocr (scale=3)	699ms	0.5	0.59

Fig -5: Keras against other OCR approaches

4.4 NUMPY

NumPy is a powerful library that can be utilized in various ways for the problem of handwritten medical prescription scanner using OCR. One of the significant ways is image preprocessing, where NumPy can be used to load, manipulate, and transform the images. This library enables operations like cropping, resizing, and normalizing images, which helps to prepare them for OCR processing. Another way NumPy can be used in the problem is for data storage and manipulation. The library can be utilized to store and manipulate data for training and testing OCR models. NumPy's efficient tools for handling multi-dimensional arrays and matrices make it an ideal choice for storing and manipulating image data. NumPy can also be used for feature extraction, such as extracting pixel intensity values from the images. These features can serve as inputs to the OCR model, which can be further normalized and transformed before

being used for training or testing. It also provides fast and efficient operations for performing complex computations on large arrays, which can be helpful for improving the performance of the OCR model.

4.4 PANDAS

One of the key ways Pandas can be used is for data storage, manipulation, and analysis. Pandas provides a powerful data frame structure, which is useful for storing and organizing the image and text data used for training and testing OCR models. It also offers various functions for data manipulation, such as filtering, merging, and grouping, which can be used to prepare the data for OCR processing and model training. Additionally, Pandas provides tools for data analysis and visualization, which can aid in understanding the patterns and characteristics of the handwritten prescription data.

4.5 MATPLOTLIB

Very useful for analyzing the results through various data representations. It can also help in plotting various forms of data and inspect various parts of the Prescription image.

5.RESULTS

5.1 CHARACTER SEGMENTATION (ALPHABETS)

We used a data model with 1262 entries. This was the accuracy using RNN and 4200 iterations:

***Red line: Training Accuracy**

Green line: Testing Accuracy

Blue line: Loss

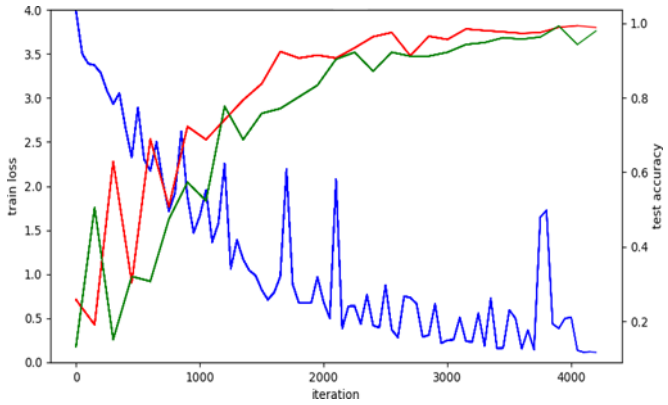

Fig -6: Accuracy of 0.9088 for Character Classification

5.2 GAP CLASSIFICATION

We used a data model with 802 entries over 1900 iterations and using Bi-RNN.

The layers used here are:

- •Convulation + Subsampling
- •ReLU
- •Max Pool
- •Fully Connected layer
- •Dropout
- •Output Layer

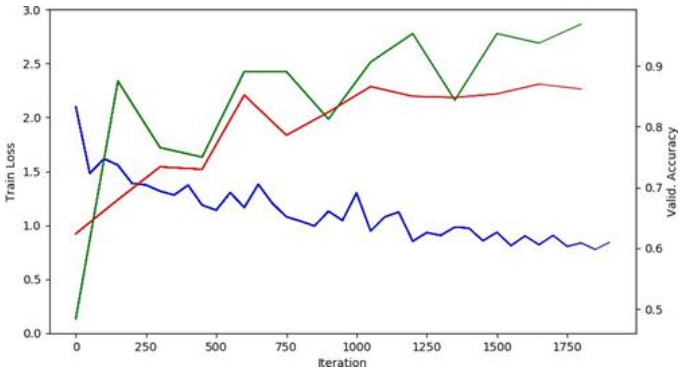

Fig -7: Accuracy of 0.8410 for Gap Classification

5.3 OVERALL OCR ACCURACY

We used a data model with 1356 entries. The overall accuracy accounts for every classifers above along with word classifiers, corrections, line segmentations and a dictionary of 15838 entries.

Table -1: Overall OCR Accuracy

Model	Correct	Total	Letter Accuracy	Overall Accuracy
Seq2Seq	626	1356	46.1652%	29.588%
Seq2Seq2CNN	830	1356	61.2094%	44.1948%
CTC	853	1356	62.9056%	56.5543%
Bi-RNN and CNN	1152	1356	84.9557%	80.1089%

6. ANDROID APP

6.1 TECHNOLOGY USED

Since Python is limited in functionality on the Android Operating System, we have to use alternatives that are Java language based.

1) Flutter: Flutter is a cross-platform mobile application development framework that can be used to develop Android and iOS apps using a single codebase. It provides faster development facilities, along with a large access to Native libraries like OpenCV and TensorFlow. It can also help build application for the iOS platform, thus making the app more accessible. It also provides easy to use UI development tools for quick and easy development.

2) TensorFlow Lite: TensorFlow Lite can be used in a handwritten medical prescription scanner Android app to build and deploy machine learning models for OCR. TensorFlow Lite is a lightweight version of the TensorFlow framework designed for mobile and embedded devices. It allows for the creation of compact and efficient models that can be deployed directly on a mobile device without requiring a server or cloud infrastructure. To use TensorFlow Lite in an Android app, the first step is to train and export a machine learning model in the TensorFlow framework. This model can then be converted to the TensorFlow Lite format using the TensorFlow Lite converter tool. The resulting TensorFlow Lite model can be integrated into an Android app and used to perform OCR on handwritten medical prescriptions. Using TensorFlow Lite for OCR in an Android app offers several benefits. The models are lightweight and can be easily deployed on mobile devices, enabling real-time OCR processing without requiring an internet connection. Additionally, TensorFlow Lite provides several tools for optimizing models, such as quantization and pruning, which can reduce the size of the model and improve its performance on mobile devices.

3) OpenCV4Android: As OpenCV is written in C/C++, it is limited for android apps. Also, android devices have less processing power and low memory as compared to a Desktop computer/Laptop. OpenCV for Android (OpenCV4Android) is a specially optimized version of the library that is tailored for use in Android apps. OpenCV4Android is designed to work on mobile devices with limited resources, such as memory and processing power. It includes a set of precompiled Java bindings that provide access to the OpenCV functions, making it easy to integrate computer vision capabilities into Android apps. Addition- ally, OpenCV4Android includes support for various image formats, camera interfaces, and hardware accelerators, making it an ideal choice for developing image processing and computer vision apps on the Android platform. Furthermore, OpenCV4Android provides support for the Android NDK (Native Development Kit), which allows developers to write computationally intensive code in C/C++ and integrate it with Java-based Android apps. This makes it possible to run computationally intensive computer vision algorithms with optimized performance, which makes it easier to implement OCR.

4) Kotlin: Kotlin is a programming language that is well-suited for developing Android applications, including those that require image processing and OCR capabilities. It offers concise and expressive syntax, which makes it easier to write clean and understandable code. For a Handwritten Medical Prescription Scanner app, Kotlin can be used to write the code for image processing, OCR, and user interface, which can be maintained and updated easily. Kotlin is compatible with Java-based OCR libraries, allowing for easy integration with existing code. Kotlin's support for functional programming constructs, such as higher-order functions and lambdas, can also be beneficial for developing OCR apps. These features allow developers to write code that is more modular and composable, which can be useful for handling the complex and varied input data that OCR apps typically deal with. Another advantage of Kotlin is its seamless interoperability with Java- based libraries and frameworks. This makes it easy to use Kotlin with existing OCR libraries written in Java, such as Tesseract OCR or OpenCV, without the need for extensive modifications or rewriting of code.

5) Google Vision OCR API: Google Vision is a cloud- based image analysis tool that can be used for OCR and image processing tasks, including those related to handwritten medical prescription scanners. Google Vision provides an easy- to-use API that allows developers to integrate OCR and other image analysis features into their Android apps. One of the key benefits of using Google Vision for OCR is that it has been trained on a large dataset of handwriting samples, making it highly accurate at recognizing handwritten text. Additionally, Google Vision provides features such as text detection and image labeling, which can be used to enhance the accuracy and functionality of the OCR system. To use Google Vision in a Handwritten Medical Prescription Scanner app, the app can send images to the Google Vision API for analysis. The API returns the recognized text along with additional metadata such as confidence scores, which can be used to improve the accuracy of the OCR results. The text can then be further processed and displayed to the user, allowing them to review and edit the recognized prescription information.

6) Firebase: Firebase can be used to handle User Authentication, so that a user can securely login and use the application. Firebase also provides access to real-time database so that a user can easily access their prescriptions anywhere, and on any device. It also provides a lot of cloud functions that can be used to perform server-side operations such as processing images, performing OCR, and storing and retrieving data from the database. Another important aspect is that it can help connect the user with Doctor/Hospital staff so that they can view the user's Prescriptions.

7) Firestore: Can be used with Firebase to store data. It can be used for its automatic scaling and extra integrated ML algorithms. It stores NoSQL data easily.

6.2 IMPLEMENTATION

Everything is pretty much the same as we did using Python. The only major difference will be that the application will be written in Java. So, all the classifiers, layers, algorithms have to be changed into a Java environment. Kotlin plays a huge role in the development of Android application and it enables the developer to create apps easily as it is derived from Java but faster to compile and shorter lines of code along with good integrations.

• Firebase can be used to Authenticate users safely and securely. There can be multiple users whose data is unique to themselves and they will be able to access it anywhere. One can also connect doctors/hospital staff with the user using Firebase. This can be done by sharing the unique User UID created by firebase to the doctors/hospital staff and giving them special permissions to view and handle the user's medical prescriptions.

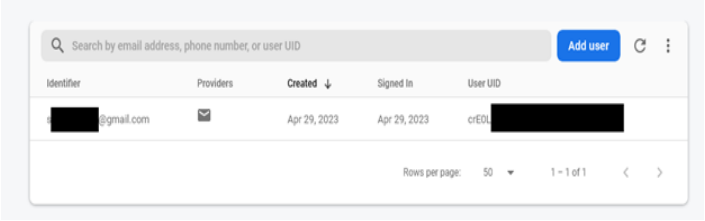

Fig -8: Firebase Authentication and UserUID

• Every type of user (Patient, Doctor, Hospital Staff, etc.) can have their own custom designed app homepage corresponding to their requirements. For example, a doctor can have the option to create or modify a prescription. A patient can have the option to store his/her documents locally or in a cloud storage system. A patient can also have an Automated Reminder added, so that they don't forget to take their medications.

7. CONCLUSION

In this fast paced and modernized world, we need to keep up with the latest technology and implement that in the simplest of situations to replace traditional redundant methods, the method in question being converting paper-based prescriptions to text based digital format documents. It is something that could revolutionize the way the medical field works and ensure smooth interaction among all parties involved in the process, be it the interaction between doctor and patient or patient and pharmacist, it assists in the smooth functioning of the process thus eliminating the need for maintaining physical records. We also concede that the tool might not provide 100% accuracy every time and in some odd and extreme cases might require human oversight. To fix the issue we shall implement a system where manual recognition would be requested if the confidence level drops below a certain threshold. Since the tool is developed specifically for the purpose of assisting medical process, it may produce inaccuracies if non-medical related documents are scanned. All in all, we strongly believe that this project could turn out to be a stepping stone for the modernization process of the overlooked parts of the medical field.

REFERENCES

[1] Pavithiran G, Sharan Padmanabhan, Nuvvuru Divya, Aswathy V, Irene Jerusha P and Chandar B, "Doctor's Handwritten Prescription Recognition System In Multi-Language Using Deep Learning"

[2] Mr.Narendrasing.B Rajput, Prof.S.M Rajput and Prof.S.M.Badave, "Handwritten Character Recognition -A Review"

[3] Esraa Hassan, Habiba Tarek, Mai Hazem, Shaza Bahnacy, Lobna Shaheen and Walaa H. Elashmwai, "Medical Prescription Recognition using Machine Learning"

[4] M. Faridnia and S. Shahriari, "OCR Preprocessing Techniques using Python for Improved Text Recognition"

[5] R. K. Kalra and M. Singh, "Multiscale Image Segmentation using Python and OpenCV"

[6] M. Solanki and S. Patel, "OpenCV with TensorFlow: An End-to-End Deep Learning Workflow"

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification"

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors"

[9] S. M. Islam, M. S. Uddin, and M. S. Rahman, "Building Mobile Applications Using Flutter: An Empirical Study"

[10] Thomas Deselaers, Tobias Gass, and Hermann Ney, "Word Spotting in Handwritten Documents"

[11] M. Dalvi, P. Natarajan, and V. S. Sheng, "OCR Error Correction using Language Modeling"

[12] I. Orovic and M. Galic., "Firebase as a Backend for Android Applica- tions"

[13] Bˇretislav Ha´jek(Breta01) , "Handwriting OCR" on GitHub

[14] E. Oliveira, L. S. Garcia, and P. Dias, "Developing Android Applications with Firebase Realtime Database"