Arogya -An Intelligent Ayurvedic Herb Management Platform

N.J. Pathiranage
Faculty of Computing, Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it17129404@my.sliit.lk

M.S.F. Nilfa
Faculty of Computing, Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it17145930@my.sliit.lk

R.A.M. Nithmali
Faculty of Computing, Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it17089500@my.sliit.lk

K.A.G.Y.N. Kumari
Faculty of Computing, Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it17014250@my.sliit.lk

K.M.L.P. Weerasinghe
Faculty of Computing, Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
lokesha.w@sliit.lk

L.I.E.P. Weerathunga
Faculty of Computing, Department of Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
ishara.w@sliit.lk

***Abstract*—Ayurvedic means a science of life and well-being with its unique approaches to social and spiritual life. Especially in Sri Lanka we have our own set of rare Ayurvedic herbs which have been utilized by generations as medicinal treatments for a variety of diseases. Absence of specialists in this area makes proper identification as well as classification of valuable herbal plants a tedious task, which is essential for better treatment. Hence, a fully automated system for herb detection and classification, information visualization regarding them is highly desirable. There are existing applications which can identify plants with low prediction accuracies, as well as to give information regarding them. However, these applications are based on foreign plant data sets that do not include valuable herbs and shrubs with medicinal qualities. Hence this research proposes an application unique to medicinal plants, which can perform all these functionalities in both online and offline approach. Here, a new Ayurvedic plant dataset prepared from scratch, and preliminary results for classification of 5 types of herbs, compared with several deep Convolutional Neural Network (CNN) models based on transfer learning are presented. Experimental results indicate Marker-based Watershed algorithm as the best object detection algorithm in a complex background, VGG-16 as the best deep CNN classification model which reached a promising testing accuracy of 99.53%, and Seq2Seq LSTM model as the best deep learning model with optimum accuracy in abstractive information summarization.**

Keywords*— abstractive information summarization, Ayurvedic medication, classification, detection, geographical information system, transfer learning

I. INTRODUCTION

Ayurveda is an ancient medicinal system evolved in India around thousands of years ago, still followed by many people as it is purely natural and has no side effects [1]. It is very relevant from ancient to this most modern time because of its power to cure chronic diseases [1]. Ayurvedic medicine mainly focuses on the Ayurvedic herbal plants & their Ayurvedic medicinal value. Accordingly, every plant on earth has some medicinal value, so it is important to protect the plant and identify its medicinal values [2]. Almost all general diseases can be cured through Ayurveda using parts like fresh leaves, flowers, roots, barks, fruits and extracts, gemstones, and minerals of shrubs and herbs around us. So, the knowledge regarding Ayurvedic plants passed down through generations should be preserved and protected.

978-1-7281-9105-8/20/$31.00 ©2020 IEEE

The World Health Organization (WHO) in 2009 states that 80% of the people worldwide still depend on Ayurvedic botanical drugs or medicine. The health-conscious today is searching for effective alternatives to the spiraling costs and side effects that result from the use of modern medicine [1]. Sri Lankans, in the last couple of millennia, have made use of the "user-friendly and traditional medicine – Ayurveda" which the top 75% of the island's inhabitants relies on due to its credit on natural and valuable medicinal plants, herbs and oils [1]. They have significant contributions towards human lives and play a leading role in the welfare and health of the global public.

Computer vision as well as image processing techniques provide promising results for automated classification of medicinal herbs. But identifying a medicinal plant with required essential herbal values is still one of the foremost challenging tasks [3] which play a key role in Ayurvedic medicine preparation. Additionally, accurate data sources, are important for many parties who are involved in the preparation of Ayurvedic medicines [3]. But absence of expert taxonomists is a major concern in this field. Even though herbal medicine lacks side effects when compared with synthetic drugs, treatment using an incorrectly recognized herb might lay claim to the patients' lives. Hence, a fully automated system that satisfies all the above-mentioned requirements regarding the local herbs is inevitable at this point in time.

Hence, the proposed solution is to develop a centralized platform (android mobile application) unique to herbal plants, which allows users to detect and classify a group of selected valuable Ayurvedic plant species accurately, based on the photograph of the plant part (leaf, root, fruit, etc.) as the basic functionality. Additionally, a dynamic abstractive summary description of the identified herb's medicinal properties, biological value, as well as reliable information sources about remedies and recipes for specific diseases associated with those plants are provided. The proposed function is a kind of real time information extraction, where users can get updated automatically. Therefore, anyone without prior knowledge also will be able to identify Ayurvedic plants hopefully and use them properly in their medications.

On a mobile device, the Ayurveda plant recognition has to be done with time and battery life in a critical manner, especially when it has to be done in a forest area. In the proposed system the whole identification process takes place on a mobile device and it does not require the internet connection. Therefore, this will be a great solution to identify Ayurveda plants in deep forest areas, where mobile network coverage is not available. The development strategy and methodology used in this approach will be able to be used and extended to identify any Ayurvedic herb furthermore.

The rest of the paper is organized as follows. Section II will discuss some related work to the system. Section III will be mainly focused on the methodology as well as the main functionalities of this system. The experimental results obtained, as well as the analysis and discussion of them will be elaborated in Section IV, and finally the research work will be concluded in Section V.

II. RELATED WORK

Novel technologies are emerging in the area of image processing frequently, especially in image segmentation. This research begins with image segmentation. Much research has been done under object detection and segmentation over several past decades. The research paper [6] presents automatic identification of fruits within complicated environmental factors such as lighting variability, branch, and leaf occlusion and tomato overlap. An enhanced tomato detection model called Tomato, based on YOLOv3, is proposed to solve these problems. A dense architecture is integrated into YOLOv3 to promote the reuse of features and to help learn a more compact and precise model. The research paper [7] presents object detection to spatially segregated bounding boxes and associated class probabilities as a regression problem. In one evaluation, a single neural network predicts bounding boxes and class probabilities straight from full images. Since the entire detection system is a single network, the performance of the detection can be optimized end to end directly. The architecture of the mode is very fast, and images are processed in real-time at 45 frames per second. And also processes a stunning 155 frames per second while still doubling the map function value rather than other object detectors.

Watershed algorithm is one of the most popular segmentation algorithms used in the processing of medical [8] and material science images [9]. This is based on the depiction of a gray picture as a topographical relief that is saturated with water, where watersheds separate water areas from various basins. The concept of texture gradient can be used [10] to establish an effective watershed segmentation strategy for common images in light of the boundaries of strength and texture. In addition, a new marker option calculation is performed to test the over-division problem. The research paper [11] introduces a new mathematical morphology-based watershed algorithm for cellular image. The watershed algorithm based on the marker enhancement segmentation approach incorporates the concept of morphological reconstruction during pretreatment. The approach can be more accurate in segmentation results compared to the original watershed method segmentation.

According to much research carried out regarding the problem of medicinal plant classification [13], there are existing applications which can identify plants with low prediction accuracies. In 2019 in the study [14], an approach has been proposed which depicts a combination of deep architectures together. Deep features have been extracted from the leaves using the fc6 layer of AlexNet and VGG16 models. After that, dimension reduction of deep features using the Principal Component Analysis (PCA) method has been applied efficiently and the best differentiated features were acquired. Finally, performances against classifications have been calculated using the K-Nearest Neighbor (KNN) algorithm. Flavia and Swedish, which are two popular plant leaf datasets, have been used for the testing of the system proposed. According to the experimental results, accuracy scores 99.42% and 99.64% were achieved for Flavia and Swedish leaf datasets, respectively.

A deep learning based Convolutional Neural Network (CNN) model has been proposed [3] to a system named AyurLeaf, in order to classify herbs using leaf features which included shape, size, color, texture etc. In addition, a standard dataset for medicinal plants has been proposed by this research work, which is habitually visible in various regions of Kerala. This dataset contains samples of leaves from 40 medicinal plants. A deep neural network inspired by Alexnet is utilized for the efficient feature extraction from the dataset. Finally, the classification is performed using Softmax and SVM classifiers. A classification accuracy of 96.76% has been achieved upon 5-cross validation, by their proposed model on the AyurLeaf dataset.

The paper [4] extends the state-of-the-art abstractive summarization architecture for multi-document summarization. This produces a comprehensive summary on a topic by combining the abstractive and extractive summarization approaches in a cascade. The state-of-the-art approach for abstractive summarization using the pointer-generator model is limited to single-document summarization. Summarization of multiple news articles on a topic can be handled one by one independently which results in multiple summaries for the same topic with possible redundancy. In order to avoid redundancy, the authors propose extractive summarization of the 6 multiple summaries as the second phase in the proposed cascade framework. The effectiveness of the framework is established using the ROUGE metric.

Another approach [5] is a multi-document text summarization using an unsupervised deep learning algorithm along with fuzzy logic. Feature matrix with seven features from the set of sample datasets from DUC2002(Document ID: AP880911-0016). The feature matrix is applied through the various levels of the RBM and finally, the efficient text summary is generated. The result indicates that this method generates efficient text summary when compared to previous methods based on evaluation metrics.

III. METHODOLOGY

In this research, the image of the leaf/digested root/fruit scanned using the application acts as the input for classification phase [Fig 1].

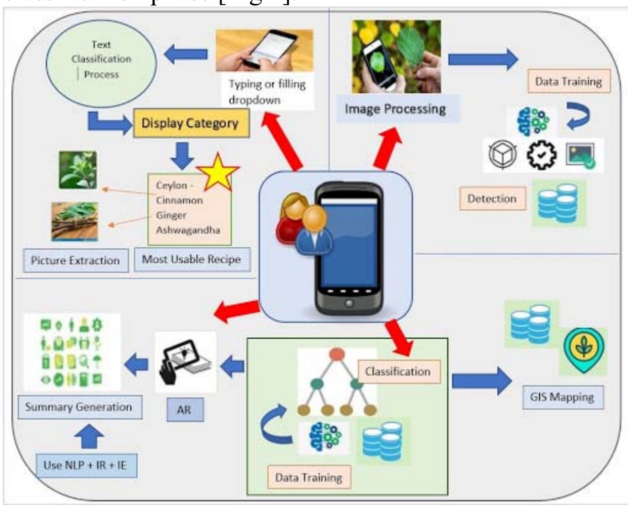

Fig 1: High Level Architecture Diagram of overall system

A. Dataset

A newly captured, prepared, and annotated Ayurvedic plant dataset from Sri Lanka was trained on the Convolutional Neural Network (CNN) from scratch. Among many herbal plants, 5 were chosen to analyze further in detail, and the images of the leaves/fruits/digested roots were collected from Ayurveda Research Institute at Navinna, alternative medicine websites and blogs related to Sri Lankan herbal plants and social media, creating a noisy dataset. A set of 1348 leaf images were retained, where 883 (48%), 235 (26%) and 230 (26%) images were used for training, validation, and testing purposes, respectively.

Summary dataset was created manually by extracting information on herbal plants from multiple websites dynamically. The top three websites from herbal plant search results were extracted, and the information was combined into a single document as the text column, while the manual summary column lied next to it in the dataset. Nearly, 2000 records were available in the manual dataset. It was split into training (90%) and validation (10%) datasets.

B. *Selecting the most accurate and highest performance segmentation technique*

Detecting the objects accurately was important because it helped the classification process to do a better job using a leaf, fruit, or root (any plant part). In addition, detecting objects from complex and noisy backgrounds was also to be done accurately. Therefore, much attention was given to detect the specific object accurately at the beginning. In order to achieve this objective, it was experimented with YOLO and marker-based watershed algorithms, and the most accurate one was selected based on the experimental results. YOLO applied the entire image into a single neural network, and then split the image into regions and projects boundary boxes and probabilities for each region [6]. But there were some limitations while applying the algorithm. It struggled to detect tiny objects that appeared in groups and it sometimes failed to generalize new or uncommon aspect ratios or configurations.

One of the most complicated operations of image processing was to distinguish touching objects in image. The watershed segmentation was applied here for this issue. Due to noise, direct application of the watershed segmentation caused over-segmentation. Markers were used as an approach to control over segmentation. The proposed algorithm was capable of segmenting the images with minimum limitations for under and over segmentation. This proposed marker-based watershed segmentation algorithm [Fig.2] provided several advantages such as it required low computation time, and provided closed contours, fast and simple.

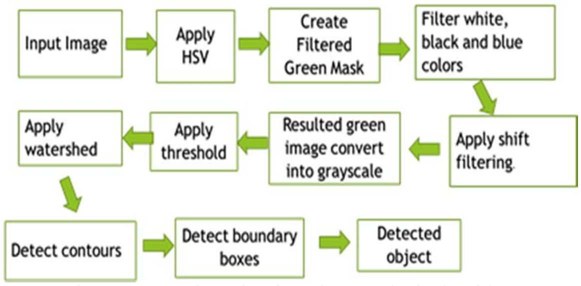

Fig 2: Proposed Marker-based Watershed Algorithm

C. *Classification of Ayurvedic plants using transfer learning based on deep CNN in an offline approach*

Main research areas of this component relied on the key pillars of deep learning, transfer learning based on deep CNN and data augmentation. Target was to analyze several deep CNN models with the acquired training and testing data from scratch, get the final testing accuracy from each in order to compare and achieve the model with the highest accuracy, and then to use it as the finalized model [Fig 3] in the herbal plant classification purpose in Arogya, based on images as the input from the mobile camera module.

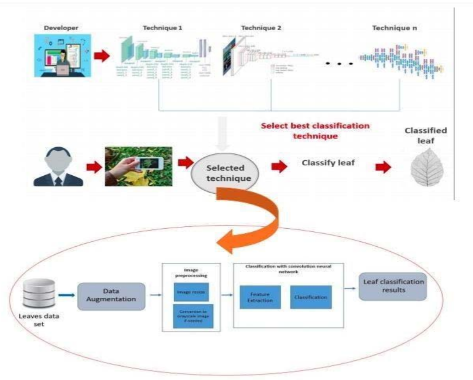

Fig 3: High Level Architecture Diagram for herb classification

After acquiring the images of leaves/fruit/digested root of the selected Ayurveda plants, they were labelled and annotated using “VGG Image Annotator” tool for multiclass classification, by forming 6 classes. After the preparation, transfer learning based on deep CNN was used on the prepared image set for processing them using TensorFlow, without data augmentation. However, training a model from scratch was too difficult, costly, and resulted in comparatively low accuracies. To overcome these challenges, transfer learning was applied on the Ayurvedic herb dataset. The proposed identification method was based on running a CNN.

After that, the dataset was retrained on the available convolutional neural network architectures, fine-tuned from pre-trained weights, and the best technique with the highest accuracy was selected. While the training dataset was augmented, testing and validation datasets were not augmented for higher accuracy purposes. Then, the selected re-trained model with highest accuracy was fine-tuned using data augmentation techniques on the labeled dataset and hyper-parameter tuning. It was re-trained with the dataset using Keras. Finally, the re-trained model was converted for deployment on mobile devices with Goggles' TensorFlow Lite.

D. Abstractive web page information summarization on Ayurvedic plants in an online approach

Main research area of this component was based on natural language processing techniques. Target was to extract dynamic information from multiple web pages and generate a summary on Ayurvedic plants [Fig 4].

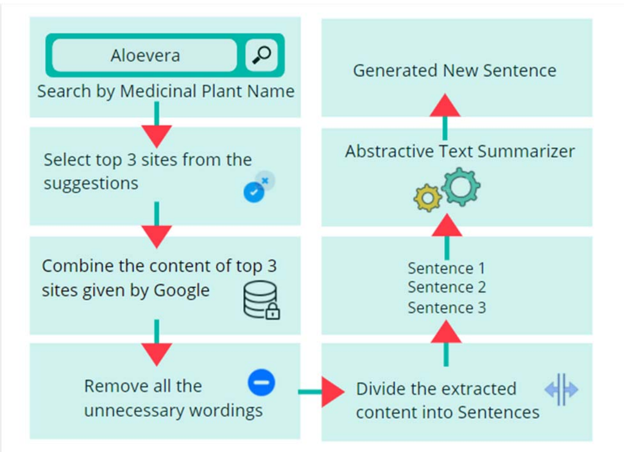

Fig 4: High Level Architecture Diagram for Multiple web pages Text Summarization

The process of text summarization is usually defined as generating an incisive and communicative synopsis while conserving the main content of the description and the overall outline. In general, for the process of text summarization, two main approaches are being executed called extractive summarization and abstractive summarization. As the methodology for this research, abstractive summarization was used. Instead of existed sentences, new sentences were generated which were totally differentiated from the original text with the most important sentences which were extracted from the original text.

A set of descriptions of medicinal plants which were extracted from highly ranked multiple websites was the input and the output was a short summary of sequence of words. Hence, this was modeled as a many to many Seq2Seq problem. Encoder and the decoder are the two major components of a Seq2Seq model. In this research component Long Short-Term Memory was used as the encoder and the decoder. The encoder-decoder was set in two phases as training phase and inference phase. As the first step encoder and the decoder were set in the training phase. The model was trained to predict the target sequence offset by one-time step. The whole input sequence was read by the LSTM model at each time step, one word was counted. Information at each time step was processed and related information present in the input was captured. Decoder also worked as a LSTM network. Entire target sequence was read word by word to predict the same sequence offset by the decoder. Decoder was capable of predicting the next word in the sequence given the previous word. At the inference phase after doing training, the model was evaluated on a new source sequence. Target sequence was an unknown one. Inference Architecture was to be set up to decode a test sequence. At the final stage, the user retrieved a summarized paragraph on the specific Ayurvedic plant.

IV. RESULTS AND DISCUSSION

Among YOLO and proposed marker-based watershed algorithms, proposed marker-based watershed algorithm was proved as the best detection algorithm for the parts of plants, even if the background was complex or noisy. Figures 5-7 indicate the results achieved from the finalized object detection model.

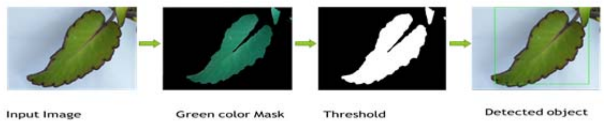

Fig 5: Leaf detection in simple background

Fig 6: Leaf detection in complex background

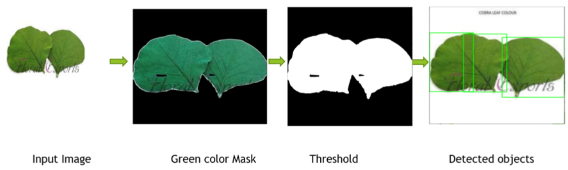

Fig 7: Detection of overlaps leaves

Then, the detected plant part was used for the classification purpose during the next phase. The finalized testing accuracy obtained from each pre-trained deep CNN architecture is illustrated below in Table I.

TABLE I. Comparison of accuracies for the selected CNN models in herbal plant classification

CNN Architecture	Final testing accuracy
InceptionV3	56.76%
MobileNetV2	63.58%
InceptionResNetV2	82.77%
Xception	86.01%
DenseNet121	89.12%
ResNet50	98.92%
VGG16	**99.53%**

According to the comparison, the pre-trained model VGG16 with a testing accuracy of 99.53% was chosen as the model with the highest accuracy and was fine-tuned with data augmentation techniques and with hyper-parameter tuning. To build the finalized CNN model with the highest accuracy, first the base model was created from the VGG-16 model, which is pre-trained on the ImageNet dataset. Then the features were extracted through freezing the convolutional base. Next a flatten layer was added, which did not affect the batch size. Then a dense layer (1x256) was added, with the activation function as 'relu' (adding of dense layers was decided by considering the number of classes). A dense layer for 6-class classification, with the activation function as 'softmax' was also added. Then the model was compiled before training and here, 'categorical_crossentropy' was used as the loss function, 'RMSprop' was used as the optimizer, (1e-4) was used as the learning rate and 'categorical accuracy' was used as metrics. Next training and testing datasets were loaded, where training dataset was augmented according to different augmentation techniques, while testing dataset was not augmented due to valid accuracy purposes. Finally, the dataset was trained for 100 epochs, with using the number of steps per epoch training as 100, and the number of steps per epoch testing as 50.

Figure 8 shows the finalized classification model summary of the selected VGG-16 pre-trained model, which achieved the highest accuracy.

```
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 3, 3, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1542
=================================================================
Total params: 15,896,134
Trainable params: 15,896,134
Non-trainable params: 0
```

Fig 8: Finalized classification model summary

Figure 9 and 10 depict the plot of accuracy on the training and validation datasets over training epochs and the plot of loss on the training and validation datasets over training epochs respectively, which were achieved from the finalized VGG-16 model.

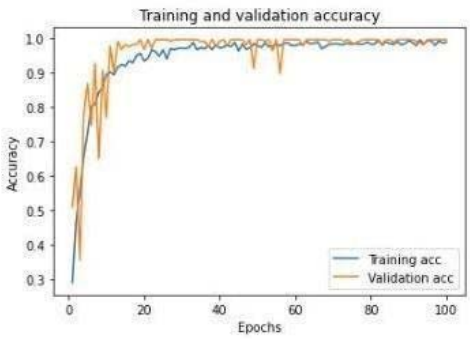

Fig 9: Finalized model performance accuracy graph

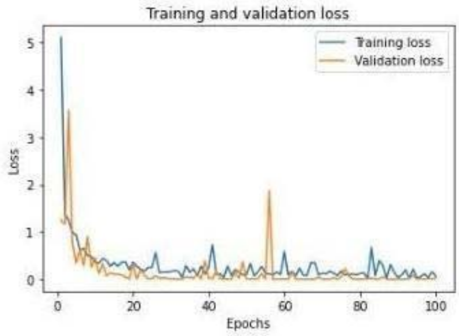

Fig 10: Finalized model performance loss graph

Table II indicates the classified and misclassified amounts of inputs against the plant class and the accuracy calculated for each class as a percentage using the finalized model.

TABLE II. Experimental classification accuracy class wise

Plant Class	**Classified plants**	**Misclassified plants**	**Accuracy**
Akkapana	40	0	100%
Cinnamon	36	4	90%
Katupila-leaf with fruit	36	4	90%
Kohomba	40	0	100%
Turmeric	38	2	95%
Turmeric - digested root	40	0	100%
Average			**95.83%**

Since the Seq2Seq LSTM model only works with short sequences, the encoder model was not capable of memorizing long input sentences. So as a solution for this problem, the Seq2Seq LSTM model with attention mechanism [Fig.11-12] was used with the testing accuracy 98.6%. While building the model, sparse categorical cross-entropy was used as the loss function since it was capable of converting integer sequence to a one-hot vector. This prevented any memory issue occurring while the process was going on. As the optimizer, RMSprop was used and as the metrics, categorical_accuracy was used. Early Stopping concept was used to stop training the neural network at the right time by monitoring a user-specified metric. Training was stopped once the validation loss started the increment process. The model was trained on a batch size of 32 and validated it on the holdout set which was 10% from the dataset. Total number of 40 epochs was used each with 51 steps. And the process was stopped at the 8th phase due to early stopping concept.

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 350)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 350, 100)     240100      input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 350, 300), ( 481200      embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 350, 300), ( 721200      lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 100)    21700       input_2[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 350, 300), ( 721200      lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 300),  481200      embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
attention_layer (AttentionLayer ((None, None, 300),  180300      lstm_2[0][0]
                                                                 lstm_3[0][0]
__________________________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 600)    0           lstm_3[0][0]
                                                                 attention_layer[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 217)    130417      concat_layer[0][0]
==================================================================================================
Total params: 2,977,317
Trainable params: 2,977,317
Non-trainable params: 0
__________________________________________________________________________________________________
```

Fig 11: Training phase building a three-stacked LSTM for the encoder

```
Layer (type)                 Output Shape               Param #
=================================================================
input_1 (InputLayer)         [(None, 350)]              0
_________________________________________________________________
embedding (Embedding)        (None, 350, 100)           240100
_________________________________________________________________
lstm (LSTM)                  [(None, 350, 300), (None, 481200
_________________________________________________________________
lstm_1 (LSTM)                [(None, 350, 300), (None, 721200
_________________________________________________________________
lstm_2 (LSTM)                [(None, 350, 300), (None, 721200
=================================================================
Total params: 2,163,700
Trainable params: 2,163,700
Non-trainable params: 0
```

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 100)    21700       input_2[0][0]
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 300)]        0
__________________________________________________________________________________________________
input_4 (InputLayer)            [(None, 300)]        0
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 300),  481200      embedding_1[1][0]
                                                                 input_3[0][0]
                                                                 input_4[0][0]
__________________________________________________________________________________________________
input_5 (InputLayer)            [(None, 350, 300)]   0
__________________________________________________________________________________________________
attention_layer (AttentionLayer ((None, None, 300),  180300      input_5[0][0]
                                                                 lstm_3[1][0]
__________________________________________________________________________________________________
concat (Concatenate)            (None, None, 600)    0           lstm_3[1][0]
                                                                 attention_layer[1][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 217)    130417      concat[0][0]
==================================================================================================
Total params: 813,617
Trainable params: 813,617
Non-trainable params: 0
```

Fig 12: Inference phase set up for the encoder and decoder

The following graphs [Fig 13-14] depict clear evidence to prove the experiment results.

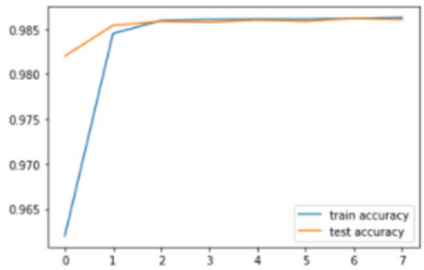

Fig13: Training and Test Accuracy over the time

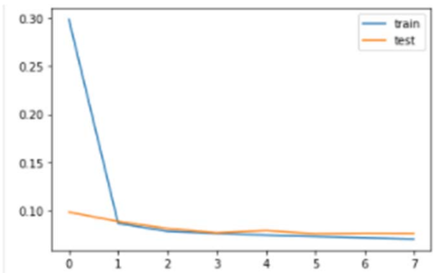

Fig14: Training and Test Value Loss over the time

V. CONCLUSION

The purpose of this research is to develop an application which is mobile-based and unique to herbal plants. This solution would be a great chance for those who are keen to learn and use Ayurveda medicine and plants, but who do not have prior knowledge about the specific domain. The application mainly creates awareness among common people about Ayurveda plants, their medicinal usage and value, and about their growth and availability throughout the country. This proposed system has been tested in various situations and it is capable of providing the most reliable and accurate output to the user. According to the main research components focused, it has been experimentally proved in this research that the most suitable algorithm for plant detection is Marker-based Watershed algorithm, and the most accurate CNN pre-trained model used for classification purpose is VGG-16, with accuracy of 99.53%. The results are highly promising, reaching over 99% accuracy using the VGG-16 model. Additionally, the Seq2Seq LSTM model with Attention mechanism which is a deep-learning model has been proved as the best model with optimum accuracy of 98.6% in abstractive summarization. This application is currently built only in English. Further, this can be applied in native languages such as Sinhala and Tamil. Since the system is designed only as a mobile application, it can be improved to a web application later with the same functionality and content.

REFERENCES

[1] Ayurveda - [Lakpura]
Available: https://lanka.com/about/interests/ayurveda/
Accessed: February 15, 2020. [Online]

[2] B. R. Pushpa, C. Anand, and P. Mithun Nambiar, "169Ayurvedic plant species recognition using statistical parameters on leaf images," *Int. J. Appl. Eng. Res.*, vol. 11, no. 7, pp. 5142–5147, 2016.

[3] M. R. Dileep and P. N. Pournami, "AyurLeaf: A Deep Learning Approach for Classification of Medicinal Plants," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2019-October, pp. 321–325, 2019, doi: 10.1109/TENCON.2019.8929394.

[4] A. K. Singh and M. Shashi, "Deep Learning Architecture for Multi-Document Summarization as a cascade of Abstractive and Extractive Summarization approaches," *Int. J. Comput. Sci. Eng.*, 2019, doi: 10.26438/ijcse/v7i3.950954.

[5] S. S. Lakshmi and M. U. Rani, "Multi-Document Text Summarization Using Deep Learning Algorithm with Fuzzy Logic," *SSRN Electron. J.*, 2018, doi: 10.2139/ssrn.3165331.

[6] G. Liu, J. C. Nouaze, P. L. T. Mbouembe, and J. H. Kim, "YOLO-tomato: A robust algorithm for tomato detection based on YOLOv3," *Sensors (Switzerland)*, vol. 20, no. 7, pp. 1–20, 2020, doi: 10.3390/s20072145.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[8] T. Atta-Fosu, W. Guo, D. Jeter, C. M. Mizutani, N. Stopczynski, and R. Sousa-Neves, "3D clumped cell segmentation using curvature based seeded watershed," *J. Imaging*, vol. 2, no. 4, 2016, doi: 10.3390/jimaging2040031.

[9] E. V. Myasnikov, "Hyperspectral image segmentation using dimensionality reduction and classical segmentation approaches," *Comput. Opt.*, vol. 41, no. 4, pp. 564–572, 2017, doi: 10.18287/2412-6179-2017-41-4-564-572.

[10] R. Hill, C. N. Canagarajah, and D. R. Bull, "Watershed Transform," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1618–1633, 2003.

[11] W. Bin Chen and X. Zhang, "A new watershed algorithm for cellular image segmentation based on mathematical morphology," *2010 Int. Conf. Mach. Vis. Human-Machine Interface, MVHI 2010*, pp. 653–656, 2010, doi: 10.1109/MVHI.2010.55.

[12] J. W. Tan, S. W. Chang, S. Abdul-Kareem, H. J. Yap, and K. T. Yong, "Deep Learning for Plant Species Classification Using Leaf Vein Morphometric," *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 17, no. 1, pp. 82–90, 2020, doi: 10.1109/TCBB.2018.2848653.

[13] B. S. Anami, S. S. Nandyal, and A. Govardhan, "A Combined Color, Texture and Edge Features Based Approach for Identification and Classification of Indian Medicinal Plants," *Int. J. Comput. Appl.*, vol. 6, no. 12, pp. 45–51, 2010, doi: 10.5120/1122-1471.

[14] B. Muhendisligi, B. Oniversitesi, and B. Muhendisligi, "Combination ofDeep Features and KNN Algorithm for Classification ofLeaf-Based Plant Species."