**Ayurvedic leaf Image classification and Translation of leaf information using Natural Language Processing**

# **Abstract**

The identification and classification of ayurvedic plants could be a quick and useful tool for academics in a variety of fields, including pharmacology, botany, agriculture, forestry, and Ayurveda medicine. Even ordinary folks who are interested in the identification and knowing the uses of such plants will find this very useful. However, not everyone is a botanical specialist and incorrect identification can cause more harm, thus we must look for a relatively error-free way for the layman to use the system in identifying and interpreting the leaf information in his/her preferred language. The proposed system uses image processing and Natural Language Processing (NLP) to achieve this. The goal of the system is to provide a simple automated approach for correctly identifying leaves and providing the information in the user's language of choice.

# **Keywords** CNN, Deep Learning, Transfer learning, NLP, Text Translation

# **1 Introduction**

Various plants with medical benefits can be found all over the planet. Research in this field could contribute to the emergence of novel drugs and also the treatment of existing disorders. This has the potential to significantly improve patient care and public health. There are many elements of plants in nature that can be useful, such as the bark, roots, leaves, flowers, and much more, but some of them pose a serious threat to human life. These plants may appear to be harmless, but they may carry some of the most lethal poisons known. The process of identifying these plants with human eyes can be difficult, time-consuming, as well as error-prone. As a result, an automated system for swiftly and precisely identifying the leaves is required. The proposed solution entails the development of an Android application that allows users to identify plant species using images of the plant's leaves taken with a smartphone, obtain the plant data, and then have it translated into the local language.

# **2 Review of Literature**

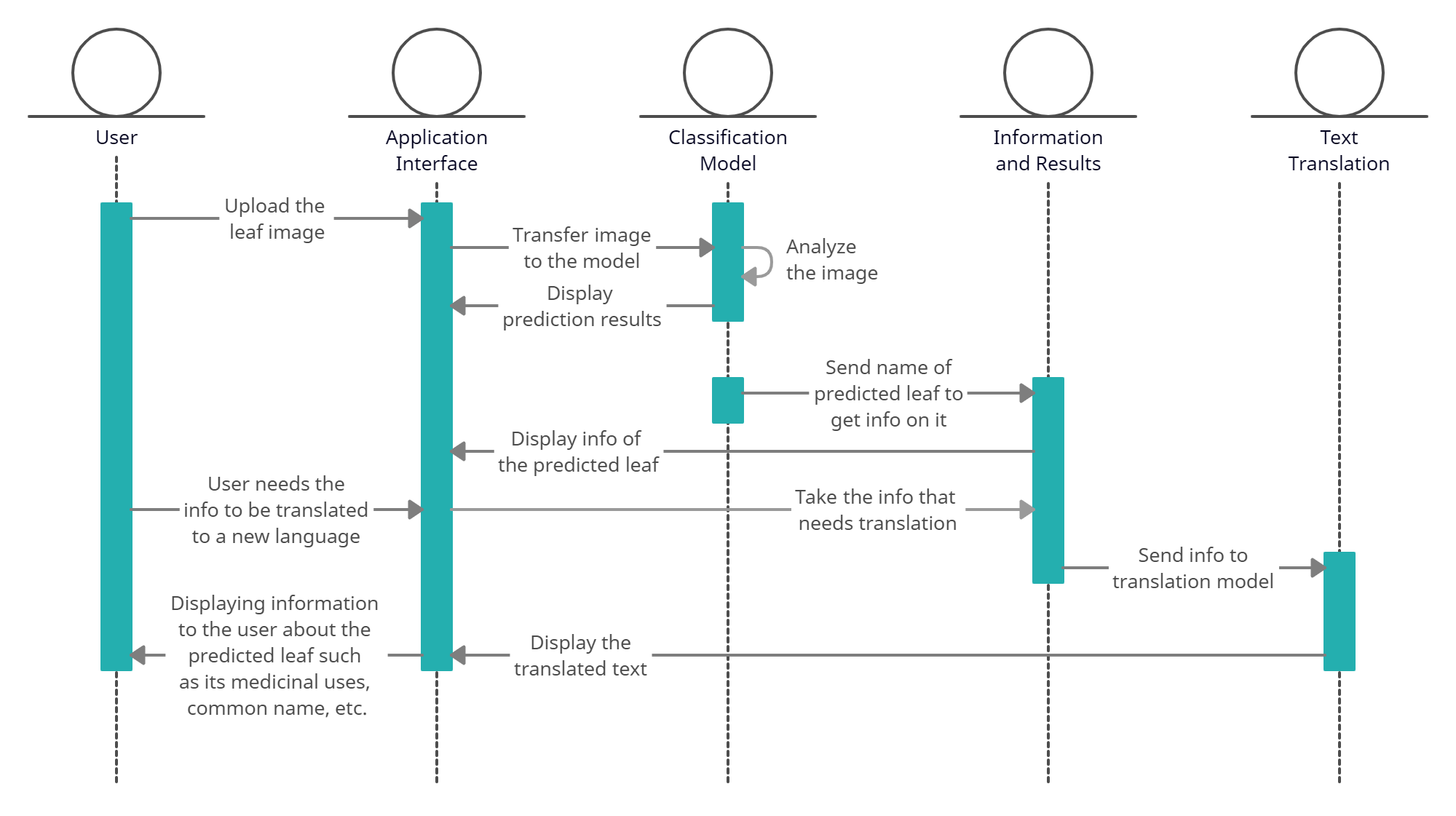
Convolutional Neural Networks (CNN) are a widely used and effective method for handling image classification problems. Various techniques can be used to solve image classification problems. A common technique to solve this problem is with the help of CNN. A custom CNN architecture was used to classify whether a patient has lung disease or not. This classification is both automatic and effective, demonstrating the importance of image processing techniques in the medical field [1]. An image classification technique SVM is implemented over a dataset of over 3000 images, the accuracy achieved is 82% whereas CNN achieved an accuracy of 93.57% [2[]](https://www.researchgate.net/profile/Ipseeta-Nanda-2/publication/342685399_Image_Classification_using_SVM_and_CNN/links/618f335ed7d1af224be324ba/Image-Classification-using-SVM-and-CNN.pdf). The use of CNN was tested on various standard datasets like the remote sensing data of aerial images (UC Merced Land Use Dataset) and scene images from the SUN database. The study showed graphs that are plotted as MSE against the number of training epochs. The evaluation of these results shows that the CNN algorithm gave a fairly acceptable classification accuracy over the tested datasets [3]. When the error rate of different variations in the convolutional neural networks was compared, it was found that among other models, the CNN model demonstrates high performance. A simple convolutional neural network imposes less computational cost thus making it suitable to work on low-end devices [4]. Running complex CNN models on low-end edge devices is a problem due to the limited computation capability. TensorFlow Lite (Tflite) is an inference compiler that integrates most of the compiler optimization techniques that have been proposed for edge computing. TFLite performs better with lightweight deep learning (DL) models compared to deep CNN models like ResNet50. TFLite is not optimized for desktop-based GPUs, most of the computations were being executed on the CPU causing over a 95% drop in the throughput. TFLite showed a decrease in model size by 75% and a decrease of approximately 35% in power consumption, making it suitable for low-end devices [5]. Different DL models were deployed on smart edge devices. A quantitative analysis of these five models on the basis of the performance of the proposed classifier in terms of accuracy (precision, recall, F1-score), flash and RAM occupation, average inference time, and energy consumption were compared. It was demonstrated that the quantization techniques outperformed the compression method in every metric we measured, by careful selection and tuning of the parameters [6]. The TensorFlow Lite Model Maker library is a high-level library that simplifies the process of training a TFLite model using a custom dataset. It makes use of transfer learning to reduce the amount of data needed for training and shorten the training time. EfficientNet-lite is a set of image classification models that are suitable for mobile/IoT devices [7]. The authors of this paper have proposed a model for text translation that follows a neural machine translation approach. The traditional models that followed the encoder-decoder approach encoded a sentence into a fixed-length variable vector which usually resulted in a bottleneck problem whenever longer sentences were passed to it. The proposed model encodes the sentence into a variable-length vector. By following this method on an English-to-French translation, their model outperformed the traditional models. The main components in this model are encoder, attention layer and the decoder [8]. The authors have presented a dataset called Samanantar dataset, which is the largest available parallel corpora for Indic languages. The model IndicTrans can translate to 11 Indic languages[9]. The authors of this paper have proposed a metric evaluation system for machine translation tasks. It is called the Bilingual Evaluation Understudy (BLEU) score. It tells us how good a candidate sentence is when compared to one or more reference sentences. It compares the n-grams of the candidate sentences to the reference sentences and counts the number of matches. The score is given from the range of 0 to 1 [10].

The authors have developed a mobile application to detect COVID-19 using X-ray images of the patient’s lungs. The model used here is a CNN model which is then converted to a TFLite model [11].

# **3 Proposed System:**

An Android application was developed in this research project that allows the user to upload an image taken with the device's camera. Using image processing techniques, the uploaded image is classified into its leaf species using image processing algorithms. The user can then click on the classified leaf species to learn more about it, such as its origin and location, as well as its medicinal applications. We have also developed a translation system so the user can get the information in different languages. The system developed uses attention mechanism to convert a source text in English to its target language in Hindi. The dataset of English-to-Hindi translation was created by us which consists of 28501 sentences. Since the user will have an option to convert the text into more than one language therefore we will be using IndicTrans for the rest of the languages.

The complete proposed system's design is depicted in figure 1.



**Fig. 1** Proposed system

## **3.1 Image classification using Convolutional Neural Networks**

CNN is short for Convolutional Neural Networks, which is a class of ANN or artificial neural networks. The defining characteristic of CNN is that it first undergoes feature extraction before the classification process is begun. CNN is mainly used in computer vision, more specifically image classification. However, it has applications in OCR and handwriting verification, social media tagging, object detection for automated systems, and image analysis in the healthcare sector among others. Convolutional is a mathematical function that calculates an integral value for the overlap of a function as it is shifted over another function. In this case, convolutional is performed on the image using a filter, which consequently creates a feature map, indicating where in the image the desired feature is located. This process is performed multiple times with different filters depending on the number of features you want to be extracted before classification. As a hyperparameter, you can specify the number of filters and their size. The values of the filters are figured out by the network through learning and backpropagation. Here an activation function such as ReLU is used to introduce non-linearity. The ReLU function makes all negative values 0, thus speeding up the computation and training processes. However, this feature map has too many cells which in turn greatly increases the computation power required. To combat this, pooling is used to reduce the dimensions and computation needed. Additionally, pooling reduces overfitting as the parameters considered are fewer. Pooling also makes the model tolerant to variation and distortion as it ignores the noise and focuses more on the main details. In convolution, as not every node is connected to every other node like in dense neural networks, the computation is quicker. When used in combination with pooling, feature extraction can be done, regardless of where in the image the desired features are located. When a filter parameter is learned, it can be applied over the entire image. After this, the classification process begins which is a regular dense neural network that uses various combinations of the extracted features to determine which class the input image fits into best. One of the drawbacks of CNN is it doesn't account for the rotation and scaling of images. This can be fixed by data augmentation methods to generate rotated and scaled versions of existing images from the dataset, and use these for training the model.

### **3.1.1 CNN implementation:**

We built a simple two-layer neural network. This neural network has two convolutional layers, two max-pooling layers, and a dense layer of 30 neurons for the 30 species of leaves in the dataset. SoftMax was chosen as our activation function since it is popularly used for multiclass classification problems. When this function is applied to each neuron in the dense layer, the probability that the image belongs to a specific class is determined. The sum of the probabilities of all the neurons present in the output layer will always be equal to one.

## **3.2 Image classification using a transfer learning model:**

Transfer learning is a deep learning technique in which a pre-trained model is used to solve a similar but new problem. The idea is to freeze the layers of the pre-trained model and simply modify the last layer or last few layers, such that we solve the new problem. Doing this ensures that the weights of the pre-trained model's layers remain unchanged, as these have usually been trained on a large dataset over a long period of time. When this modified model is trained for the new dataset, you get a feature vector. One of the modifications is editing the softmax layer to classify the vector into the desired number of classes as opposed to the original number. For example, if training a model from scratch takes 30 epochs to achieve a specific level of accuracy, transfer learning can achieve a higher level of accuracy in just 5 epochs. We save hours of CPU and GPU computing time and energy as a result of this. This is why transfer learning is so prevalent in computer vision and natural language processing. Before attempting to develop a model from the ground up, transfer learning should be tried.

### **3.2.1 Transfer Learning Implementation**

For transfer learning, we have used an existing pre-trained model called EffiecientNet-lite 0. They are a set of mobile/IoT-friendly image classification models. Transfer learning was done using the TFLite model maker library which simplifies the process of training a TensorFlow Lite model using a custom dataset. It uses transfer learning to reduce the amount of training data required and shorten the training time. The dataset used was split into 80% train and 20% test data

## **3.3 Using a JSON file to retrieve leaf information**

JSON stands for JavaScript Object Notation. JSON files are used to send and receive data between computers. In our case, we only use it to get data. This is done using the Volley library in Android, which is an HTTP library developed by Google with the aim of simplifying and optimizing networking for Android apps. For information regarding the plants in our app, we created a JSON file that stores the different plants in an array, each having the properties such as scientific name, common name, family, description, uses, benefits, ayurvedic information, origin and location. This file is stored online on Github. When A leaf has been successfully classified, the app must now display information corresponding to the identified plant. For this, we must first create a request queue in which we add request objects. Following its addition, it passes via the cache thread, where Volley determines if the request can be completed using existing cache files. If it can, it is parsed and returned immediately. If it cannot be done, it gets added to the network queue where it completes the HTTP GET request, parses the received data and also stores it in cache, and finally does a POST method on the parsed data. Now that the data is received we find the plant we want and print all of its properties for the user to read.

## **3.4 Translating Leaf Information from English-to-Hindi**

In the preprocessing step, the following operations were done for every sentence before it was sent for training.

1. Convert text to lower case.
2. Remove Apostrophe.
3. Add Spaces around special characters.
4. Add 'start\_' and '\_end' tags at the beginning and at the end of the sentence respectively.

For e.g. :

start\_ the size and shape of the leaves are 3 . 8 to 11 . 5 centimeters. \_end

start\_ पत्तियों का आकार और आकार ३ . ८ से ११ . ५ सेंटीमीटर है। \_end

We then created a tokenizer and applied the tokenizer on the source and target sentences. For training, 20% of the data was used for testing the model and 80% for training purposes. Next, we create the Sequence to Sequence model with Bahadanu's Attention Mechanism. It consists of encoder, attention layer, and decoder. Finally, the model was trained on 20 epochs.

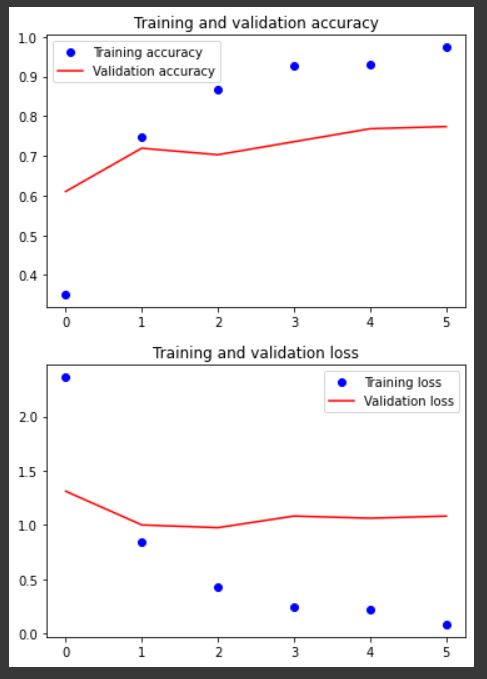
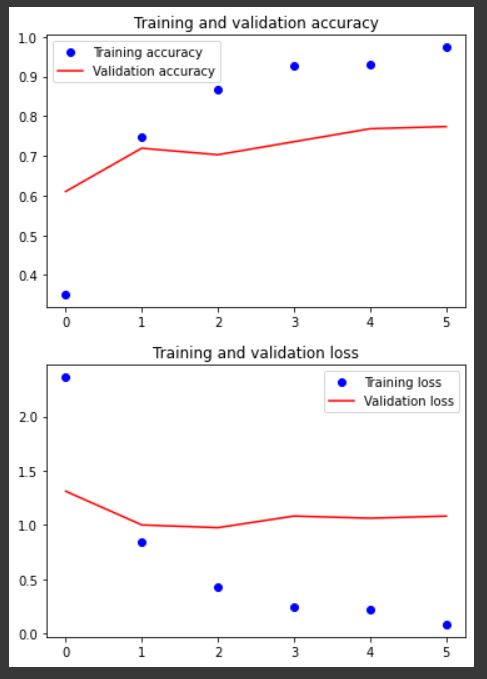
## **3.5 Translating Leaf Information to 11 languages using Indictrans**

To support the translation of information to more than one language, we have made use of the IndicTrans model. Our system uses the English to Indic model (v0.3) which supports 11 Indic languages. We will be using all 11 languages i.e Hindi, Bengali, Gujarati, Kannada, Malayalam, Marathi, Oriya, Punjabi, Tamil, and Telugu.

# **4 Implementation and experimental results**

## **4.1 CNN Results**

The images in the dataset are 1200x1600 pixels in size, and as you can expect, loading and training such large files would require quite a lot of RAM, which we did not have [12]. As a result, we had to preprocess the images. In the preprocessing stage, the photos were downsized to 200x200 pixels. The model was compiled using the ‘adam’ optimizer and uses ‘categorical cross entropy’ as its loss function. With the patience parameter set at 3, early stopping was also employed to monitor the validity loss. With early stopping our validation loss reaches its minimum after only 7 epochs and it has achieved an accuracy of 77% and a loss of 1% over the validation data. In the graph of training and validation accuracy shown in Figure 2, the X-axis represents the number of epochs and the Y-axis represents the accuracy in percentage. We can observe that we are overfitting, it may be due to the dataset being small. A larger dataset could solve this issue. Our training accuracy (in blue) approaches 100%, whereas our validation accuracy (in red) hovers around 77%.

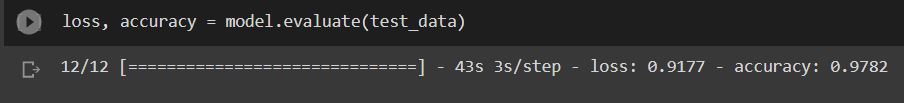
 

**(a) (b)**

**Fig. 2** Graphical representation of training and validation accuracy (a) and loss (b) using CNN algorithm

## **4.2 Transfer Learning results**

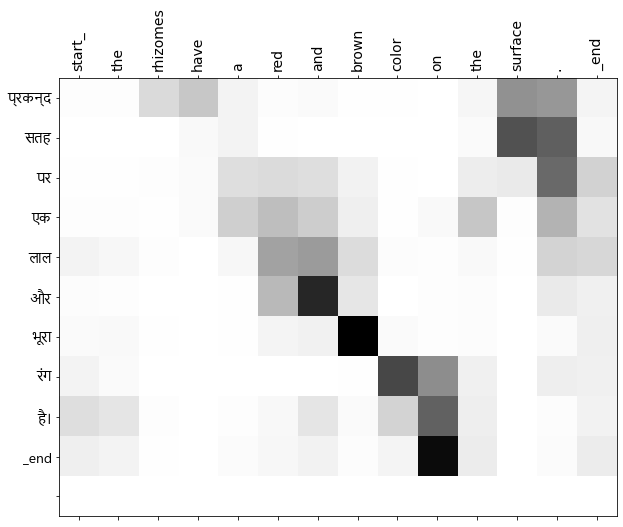
The dataset was split into 80% train and 20% test. Since we are using transfer learning, we do not need the enormous ram requirements for loading our images, hence there was no need for the resizing of images. The model was trained over 10 epochs with a batch size of 45 and took 133 seconds to complete. It was able to obtain a 97% accuracy and a 0.9% loss over the validation data as shown in Figure 3.



**Fig. 3** Evaluation of the transfer learning model over the test dataset

## **4.3 NLP results**

Figure 4 shows the attention plot of the translated sentence.



**Fig. 4** Attention plot of the translated sentence using our model

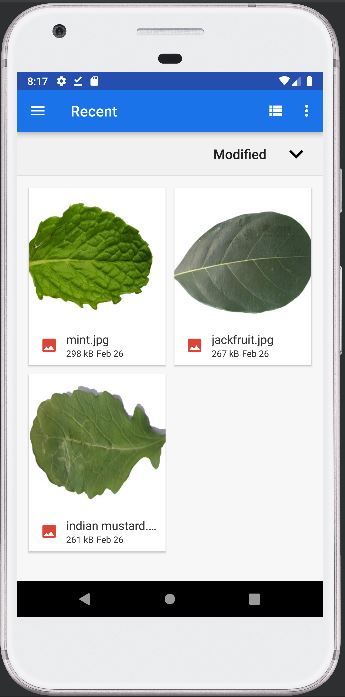
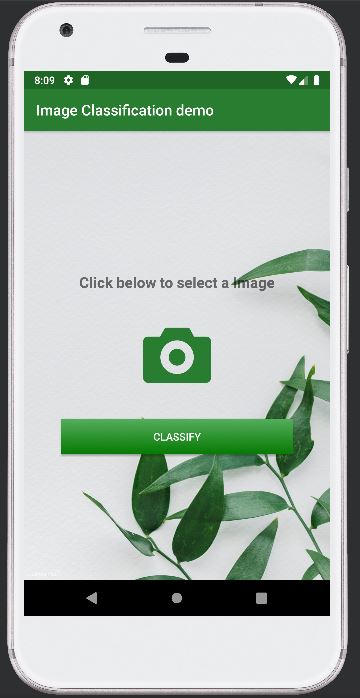
We can see how the word ‘surface’ was given higher attention for translated text ‘सतह’. Similarly, ‘brown’ was given higher attention to predict ‘भूरा’, ‘color’ to ‘रंग’ and ‘and’ to ‘और’.

Using the Bilingual Evaluation Understudy (BLEU) score, we evaluated the translated sentences. For example, the following sentence: *The tree grows white and fragrant flowers* has a score of 0.55 when compared to the translation from IndicTrans.

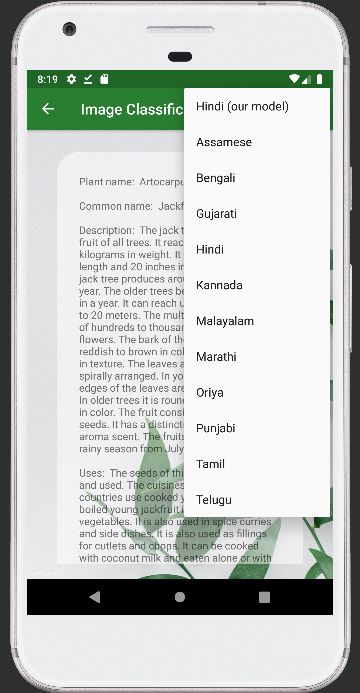
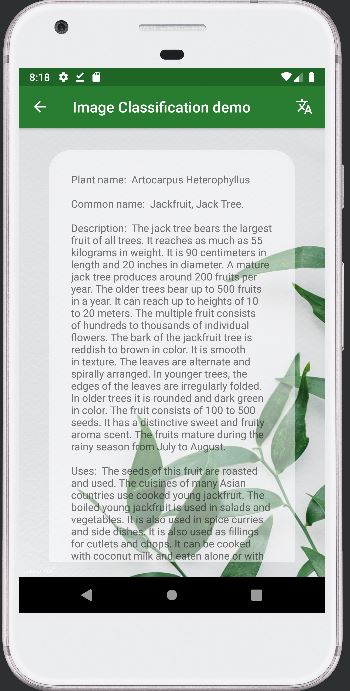
## **4.4 Android App**

We have successfully built a Java application in Android Studio. Our NLP model was hosted locally. The IndicTrans model and the JSON data file were both hosted online.

**(a) (b) (c)**



**(d) (e) (f)**



**(g)**



**Fig. 5** Screenshots of the app with leaf classification (**a-c**) and translation using NLP (**d - g)**

To use the application we select a leaf image from our device and upload it to the app. After the leaf image is uploaded we click the "Classify" button for the system to predict the type of leaf species. The system detects and displays the name of the species the leaf belongs to. By clicking on the name of the leaf we get information about the leaf in English language. To display information in a regional language, a translate option is also available which provides translation in 11 languages. Using the application is easy as the design is simple and easy to navigate.

# **5 Conclusion**

In this research, we have built an image classification and a text translation model. We followed two different approaches for building an image classifier i.e. a CNN model and Transfer Learning. Using CNN we achieved 77% accuracy and 97% accuracy with transfer learning. These results show that our system can classify the 30 leaf species with good accuracy. The text translation model was built using the Sequence to Sequence model with Bahadanu's Attention Mechanism. The translated sentences are evaluated using the BLEU metric and show satisfactory results. We also used the IndicTrans model to translate the sentences into 11 more languages. Finally, we developed an Android App which combines all these features into one application.

# **References**

1. Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014, December). Medical image classification with convolutional neural network. In *2014 13th international conference on control automation robotics & vision (ICARCV)* (pp. 844-848). IEEE.
2. Chaganti, S. Y., Nanda, I., Pandi, K. R., Prudhvith, T. G., & Kumar, N. (2020, March). Image Classification using SVM and CNN. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1-5). IEEE.
3. Jaswal, D., Vishvanathan, S., & Kp, S. (2014). Image classification using convolutional neural networks. *International Journal of Scientific and Engineering Research*, *5*(6), 1661-1668.
4. Guo, T., Dong, J., Li, H., & Gao, Y. (2017, March). Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)* (pp. 721-724). IEEE.
5. Verma, G., Gupta, Y., Malik, A. M., & Chapman, B. (2021, June). Performance evaluation of deep learning compilers for edge inference. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 858-865). IEEE.
6. De Vita, F., Nocera, G., Bruneo, D., Tomaselli, V., Giacalone, D., & Das, S. K. (2020, September). Quantitative analysis of deep leaf: A plant disease detector on the smart edge. In *2020 IEEE International Conference on Smart Computing (SMARTCOMP)* (pp. 49-56). IEEE.
7. Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.
8. Bahdanau, D.; Cho, K. & Bengio, Y. (2014), 'Neural machine translation by jointly learning to align and translate', arXiv preprint arXiv:1409.0473
9. Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, Mitesh Shantadevi Khapra. 2022. Samanantar: The Largest Publicly Available Parallel Corpora Collection for 11 Indic Languages. Transactions of the ACL.
10. Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: a Method for Automatic Evaluation of Machine Translation](https://aclanthology.org/P02-1040). In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
11. Bushra, K. F., Ahamed, M., & Ahmad, M. (2021). Automated detection of COVID-19 from X-ray images using CNN and Android mobile. Research on Biomedical Engineering, 37(3), 545-552.
12. S, Roopashree; J, Anitha (2020), “Medicinal Leaf Dataset”, Mendeley Data, V1, doi: 10.17632/nnytj2v3n5.1