

# Abstract

Plant Disease Prediction is an application which will detect and provide some remedial measures for diseases in the crop to the users. Initially the client can either click or upload the image of the diseased crop in the application. Once the plant disease is matched with the existing data, then the effective remedial measures such as what action should they take about the disease is provided. The image is processed for the effective remedial measures using the machine learning InceptionV3 Algorithm. In its current form, our application would be as a preliminary tool that could assess the users by providing some remedial measures like what type of fertilizers to use and the measures to be taken by comparing it with the datasets provided in the database. This comes with the simple Mobile application for handy and easy use of this service. The model is converted into a TFLITE model and then embedded into the Mobile Application.

# Table of Contents

<b>1. INTRODUCTION</b>	<b>3</b>
1.1 BACKGROUND	5
1.2 TRADITIONAL THRESHOLD SEGMENTATION METHODS	7
1.3 IMAGE RECOGNITION SYSTEM	8
1.4 OBJECTIVE OF THE PROJECTS	11
1.5 LIMITATION OF THE PROJECT	11
1.6 THESIS OVERVIEW	11
<b>2. LITERATURE REVIEW</b>	<b>12</b>
<b>3. METHODOLOGY AND SYSTEM BLOCK DIAGRAM</b>	<b>16</b>
3.1 COMPONENTS & PACKAGES	16
3.1.1 TENSORFLOW	17
3.1.2 KERAS	25
3.1.3 ANDROID STUDIO	31
3.1.4 KOTLIN	33
3.1.5 INCEPTION V3	37
3.2 SYSTEM DIAGRAM	40
3.2.1 USE CASE DIAGRAM	40
3.2.2 SYSTEM LEVEL DESIGN	41
3.3 METHODOLOGY	44
3.3.1 Importing the Libraries	44
3.3.2 Loading the data	45
3.3.3 Label mapping	46
3.3.4 Transfer Learning with TensorFlow hub	46
3.3.5 Data Preprocessing	47
3.3.6 Build the model	48
3.3.7 Specifying Loss Function and Optimizer	49
3.3.8 Training Model	50
3.3.9 Checking Performance	50
3.3.10 Random test	51
3.3.11 Convert model to TensorFlow Lite	52
3.3.12 Add TFLite model in our Android Project	53
3.3.13 Create classifier class to load our model and read the file with labels:	53
<b>4. RESULT &amp; DISCUSSION</b>	<b>59</b>
<b>5. CONCLUSION</b>	<b>69</b>
<b>6. REFERENCE</b>	<b>70</b>

# CHAPTER ONE

## 1. INTRODUCTION

Machine Learning behaves like a self-learning concept which will work without any interruption of a human. Now a day's self-driving cars, hand-writing recognition, Stock market are some of the examples of Machine Learning concepts. Machine learning will be able to predict the future based on the past or historical data. Computer programs are said to be learned from experience  $E$  with respect to some clause of task  $T$  and performance measure  $P$ , if its performance on  $T$  as measured by  $P$  improves with experience  $E$ . Machine learning broadly uses three major learning algorithms Supervised learning, Unsupervised learning, Reinforcement learning. Machine learning can be used in each and every routine task performed by human being. The research work deals with plant disease prediction with the help of machine learning.

A plant disease is a physiological abnormality. Once a plant suffers from any diseases it shows up certain symptoms. symptoms are the outward changes in the physical appearance that are gradually developed and can be witnessed by naked eyes. Illustrations of symptoms are wilt leaf spots, rots, cankers and many more. The visible effects of disease can broadly categorize in following types: -

**Wilting**, it is loss of turgor pressure in a plant leading to temporary or permanent drooping of leaves, shoots, or entire plants from lack of water or infection by different pathogens.

**Spot**, is a definite, localized, round to regular lesion, often with a border of a different color, characterized as to location (leaf spot, fruit spot) and color(brown spot, black spot).

**Powdery mildew**, is a fungal disease that affects a wide range of plants. Infected plants display white powdery spots on the leaves and stems. As the disease progresses,the spots get larger and denser.

**Galls**, these are abnormal growths that occur on leaves,twigs,or branches.They may be simple lumps or complicated structures, plain brown or brightly colored.

**Dryness**, after the normal aging process generally leaf's leaves dry and fall down from the tree,but at other times drying of leaves may be a symptom of fungal attacks.

In plant disease diagnosis, data provided is small and some of the values are missing that will require imputation of values. We will replace all the null values with -1.

The proposed research work applies the concept of ensemble learning, that is implemented through machine learning algorithms.After implementation the result is compareto get the model that has the highest accuracy.

India is an agricultural country. 70% of Indian economy depends on agriculture but leaf infection phenomena causes the loss of major crops resulting in economic loss. Leaf infection is the invasion of leaf tissues by disease causing agents such as bacteria, virus, fungus etc leading to degradation of the leaf as well as plant.

This can be characterized by spots on the leaves, dryness of leaves, color change in leaves and defoliation. The leaf infections may occur due to environmental condition changes such as huge rainfall, drastic changes in temperature or may be due to improper maintenance and some insects and pesticides. Once the disease causing organisms such as bacteria, virus etc, entered into the leaf tissue, they started multiplying and decreasing the strength of the leaf and degradation started.

For instance it is seen that the outbreak of diseases which leads to large scale death and famine. It is estimated that the outbreak of helminthosporium of rice in north eastern India in 1943 caused a heavy loss of food grains and death of a million people. In order to detect and diagnose the leaf infection/disease various research works have been carried out and various methods or algorithms have been proposed. For example grapefruit peel diseases were analyzed by color texture features analysis. The texture feature analysis is intern categorized into structural, statistical, model based and transform method

## 1.1 BACKGROUND

In India, There is a great variance of crops that farmers possess. The ancient and ionic access for detection and recognition of plant diseases is dependent on naked eye observation, which is a step by step method also gives less certainty. In India, Especially in provincial areas consulting experts found out that plant disease is expensive and time consuming due to availability of experts. A large number of teams of experts as well as continuous monitoring of experts are obligatory, which costs a huge amount when farms are

enormous. Also Unnecessary use of pesticides might be dangerous for natural resources such as water, soil, air, food chain etc.as well as it is expected that there needs to be less contamination of food products with pesticides. Automatic Detection of plant diseases is necessary to detect the symptoms of diseases in early stages.The Fundamental Identification Of the affected plant or cropisits leaves.One general disease is brown and yellow spots, early and late scorch, and other are fungal, viral and bacterial diseases.What is digital image processing?–DigitalImage processing is the technique which uses computer algorithms to create processes, communicate and display digital images.Isused for measuring damaged areas of disease, and to determine the difference in the color of the affected area. In this paper we use the following techniques for automatic plant disease detection after acquiring the image from the digital camera the image is pre-processed to increase or decrease the contrast accordingly and enhancement of the image to reduce noise.

The process of separating or grouping an image into different parts is called Image Segmentation. Imagesegmentation is the process of partitioning or grouping an image into different parts. Features are extracted before applying SVM algorithm for classification and detection,apparently there are different types of techniques such as Otsu segmentation, K means clustering,thresholding method,etc. These parts normally correspond to something that humans can easily separate and view as individual objects. Computers have no means of intelligently recognizing objects, and so many different methods have been developed in order to segment images. The segmentation process depends on different features

found in the image which can be colour information, boundaries or segment of an image.

The main step involved in the image processing is capturing digital high resolution images. The Images of the healthy and unhealthy regions are captured and stored for further experiments. For Image enhancement The Images are pre-processed and applied. Segmentation Process is processed by capturing the images of fruits or leaves and segmenting the image through InceptionV3 for training.

## 1.2 TRADITIONAL THRESHOLD SEGMENTATION METHODS

Threshold can never be ignored in image processing. Iter-ative Method, Otsu Method, and 2-Mode Method are the most common threshold segmentation methods. This section introduces these traditional methods. Iterative Method. The Iterative Method can calculate the threshold to a certain extent automatically. For the iterative process, the Iterative Method includes a prior knowledge concerning the image and noise statistics. And the optimal segmentation threshold can be found by continuously reduc-ing the grayscale mean [2].

**Otsu Method**, Given the split threshold of foreground and background  $T$ , the foreground image ratio  $w_0$ , the average gray scale  $u_0$ , background image ratio  $w_1$ , and average gray scale  $u_1$ , the total gray scale of the image is  $uT = w_0 \times u_0 + w_1 \times u_1$ .

When  $T$  makes the variance

$$\text{value } \sigma^2 = w_0 \times (u_0 - uT)^2 + w_1 \times (u_1 - uT)^2$$

maximum, it becomes the best segmentation threshold [3].

**Mode Method**, The image is often composed of normal foliage and diseased area, so the histogram of grayscale can be regarded as two normal distribution functions, which is shown in Figure 1. Select the trough, that is,  $T$  position in split the image into two parts, and the result

$$g(x, y) = \begin{cases} t_0 f(x, y) & \text{if } f(x, y) < T \\ t_1 f(x, y) & \text{if } f(x, y) \geq T \end{cases}$$

is the segmentation threshold, usually  $t_0=0$ (black),  $t_1=1$ (white) [4]

### 1.3 IMAGE RECOGNITION SYSTEM

In most practical situations, however, traditional methods are not able to be the most appropriate choice. It is unwise to ignore the difference between actual images and hypothesized data. Therefore, improved methods and a new recognition system based on multiple linear regression are created. This Section introduces the characteristics of this system.

**Improved Histogram Segmentation Method** (CalculateThreshold Automatically). Traditional 2-Mode Method needs to set the threshold manually. As the user has a huge task burden, it lacks identification efficiency.

This proposed segmentation method can automatically determine the threshold. It can greatly reduce the user's task burden and optimize the image segmentation process.

- (1) Perform median filtering operation on grayscale image in 5-by-5 neighborhood, smooth value with RobustLoess (quadratic fit) and specified span of moving average(17). These preprocessing operations are designed to fit histogram for segmentation.



- (2) Limit Selection Range(100–190of255pixels)and minimum peak interval (10 pixels), because there are some interferential fluctuations on histogram that need to be filtered out.
- (3) Pick the maximum height (between peak and trough)andlocatethismostsuitabletrough,whichistheadaptivethreshold $T$ .
- (4) Segment image according to the threshold $T$ [5]. Compared with other threshold segmentation methods,this improved histogram segmentation method is more accurate.

Extract Green matrix in RGB color image and calculate mean of this two-dimensional matrix. MeanSis average of green pixels of the segmented image, Mean represents green pixel mean of original image. The ratio is smaller, the more parts of normal foliage are excluded.

New histogram segmentation method works better in filtering out normal foliage com-paring with Iterative Method and Otsu Method, even indifferent severities. Both of the two methods remain a little more normal foliage parts after segmentation. This is mainly because the Iterative Method is greatly influenced by the overall grayscale of the image; it is weak in distinguishing the subtle pattern and will cause division errors. Although Otsu Method Is more stable and practical, it will have problems while handling gray statistic, especially when the ratio of the target lesion area to the background area is very small, so it cannot be used in this system either. In conclusion, the improved histogram segmentation method is more suitable for lesion segmentation in this system with great advantages, such as fast, efficient, and accurate.

**Disease Recognition System** Based on Multiple LinearRegression. In this system, a total of 11 features are extracted from three aspects: color, texture, and shape. In the color aspect, three characteristic values (Hue, Saturation, and Value) are extracted to represent the color features of the lesion [6]; the

intexture part, energy and homogeneity are selected by using the gray level cooccurrence matrix. Meanwhile, the statistical matrix is also used to collect four characteristics (smoothness, third-order moment, consistency, and entropy) [7]; for the shape, this system chooses the degree of rectification and density. All the extracted 11 characteristic parameters still remain stable no matter how the image changes in rotation, translation, and scale. Thus, they are representative and comprehensive.

The selected characteristics are regarded as 11 independent variables  $x_1, x_2, \dots, x_{11}$ , and the dependent variable  $y$  is the severity of the plant disease. Therefore, the regression model is

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_{11} x_{11} + \varepsilon$$

In this system, plant diseases are divided into four categories, including normal situations, minor disasters, medium disasters, and serious disasters. And a score system, from 0 to 100, is used to assess the severity of disease. The higher the score is, the more serious the disease is. 0–25 scores are normal, 25–50 scores are minor disasters, 50–75 scores are medium disasters, and 75–100 scores are serious disasters. Each situation has 10 images, the training library has a total of 40 images.

There are 11 selected eigenvalues of the 40 training images shown in Figure 4; the database has 40 columns, which are 40 images, and 11 lines, which are 11 characteristic parameters.

Put the 11 features into multiple linear regression models. The coefficients and confidence intervals are calculated via using the least squares estimation algorithm.

The 14th point is an anomaly and exceeds the credible range. Therefore, we remove this point, and then re linear regression of 39 normal points helps to obtain the correct coefficients and confidence intervals.

Finally, after eliminating the residua, the obtained multiple regression system model is evaluated and analyzed to test the accuracy of the recognition system.

## 1.4 OBJECTIVE OF THE PROJECTS

- To detect unhealthy regions of plant leaves.
- Classification of plant leaf diseases using texture features.
- Coding is used to analyze the leaf infection.
- To Give the remedy information to the farmer.
- To Make this service available in a Mobile App which can run on low Configuration devices.

## 1.5 LIMITATION OF THE PROJECT

Some of the limitation of this project is listed as follows:

- Low Image Quality.
- Accuracy can never be a cent percentage. Hence there may be a slight variation in results.
- Mobile Processing power can be a constraint for predicting the plant disease.
- Storage space in mobile devices to install the android application.
- This service is only developed for Android Application. Hence IOS users cannot make use of this service.

## 1.6 THESIS OVERVIEW

**Chapter one:** Gives about the brief introduction description to the Plant disease prediction and its implementation in Android Application and

answers the why we have done this project, what are the objectives, what the advantages we gained from the project and including its scopes.

**Chapter two:** Covers the extensive literature review of previous works on Plant Disease Prediction and different established standards and protocol and the platform of it how can it implemented

**Chapter three:** Methodology and system block diagram: in this chapter a brief description of the structure and interfacing of the Python Packages such as Tensorflow, Keras, and the algorithms Inception V3.

**Chapter four:** System design and Implementation: which covers the detail design and implementation of all parametric and the overall integrated system or the main projects simulation results of biometric authentication ATM machine system.

**Chapter five:** Conclusion and recommendation: this chapter concludes the entire work performed during the projects and point out some few recommendations on improving the performances and the efficiency

## CHAPTER TWO

### 2. LITERATURE REVIEW

Various techniques of image processing and pattern recognition have been developed for detection of diseases occurring plant leaves, stems, lesions etc. by the researchers. The Sooner disease appears on the leaf should be detected, identified and corresponding measures should be taken to avoid

loss. Hence a fast, accurate and less expensive system should be developed. The researchers have adopted various methods for detection and identification of disease accurately. One such system uses thresholding and back propagation network. Input is grape leaf image on which thresholding is performed to mask green pixels. Using K-means clustering segmented disease portion is obtained. Then ANN is used for classification

[1] S. S. Sannakki and V. S. Rajpurohit proposed a “Classification of Pomegranate Diseases Based on Back Propagation Neural Network” which mainly works on the method of Segment the affected area and color and texture are used as the features. Here they used a neural network classifier for the classification. The main advantage is it Converts to  $L^*a^*b$  to extract chromaticity layers of the image and Categorisation is found to be 97.30% accurate. The main disadvantage is that it is used only for the limited crops.

[2] P. R. Rothe and R. V. Kshirsagar introduced a "Cotton Leaf Disease Identification using Pattern Recognition Techniques" which Uses snake segmentation, here Hu's moments are used as a distinctive attribute. Active contour model used to limit the vitality inside the infection spot, BPNN classifier tackles the numerous class problems. The average classification is found to be 85.52%.

[3] Aakanksha Rastogi, Ritika Arora and Shanu Sharma, "Leaf Disease Detection and Grading using Computer Vision Technology & Fuzzy Logic". K-means clustering used to segment the defected area; GLCM is used for the extraction of texture features, Fuzzy logic is used for disease

grading. They used artificial neural network (ANN) as a classifier which mainly helps to check the severity of the diseased leaf.

[4] Godliver Owomugisha, John A. Quinn, Ernest Mwebaze and James Lwasa, proposed "Automated Vision-Based Diagnosis of Banana Bacterial Wilt Disease and Black Sigatoka Disease" Color histograms are extracted and transformed from RGB to HSV, RGB to L\*a\*b. Peak components are used to create max tree, five shape attributes are used and area under the curve analysis is used for classification. They used nearest neighbors, Decision tree, random forest, extremely randomized tree, Naïve bayes and SV classifier. In seven classifiers extremely, randomized trees yield a very high score, provide real time information and provide flexibility to the application.

[5] uan Tian, Chunjiang Zhao, Shenglian Lu and Xinyu Guo, "SVM-based Multiple Classifier System for Recognition of Wheat Leaf Diseases," Color features are represented in RGB to HIS, by using GLCM, seven invariant moment are taken as shape parameter. They used an SVM classifier which has MCS, used for detecting disease in wheat plants offline.

[6] The other method uses PCA and ANN. PCA is used to reduce the dimensions of the feature data. to reduce the no. of neurons input layer and to increase speed of NN.

[7] Sometimes thresholds cannot be fixed and objects in the spot image cannot be located. Hence Authors proposed LT RG-algorithm for segmentation of image.

[8] In cucumber leaf disease diagnosis, spectrum based algorithms are used.

[9] In the classification of rubber tree disease a device called spectrometer is used that measures the light intensity in the electromagnetic spectrum. For The analysis SPSS is used.

[10] In citrus canker disease detection uses a three level system. Global descriptor detects diseased lesion. To Identify disease from similar disease based regions zone based local descriptor is used In Last stage two level hierarchical detection structure identifies canker lesion.

[11] For identification of disease on plant and stems first segmentation is carried out using K-means clustering. Feature Extraction is done by CCM method. Identification Is done by using BPNN.

[12] With relevance to grapes, the fruit mostly suffer with tree types of diseases viz Powdery Mildew, Downy Mildew and Anthracnose. The Two diseases are considered Powdery Mildew and Downy Mildew.

[13] In 2011, an innovative approach was presented[1] to automatically grade the disease on plant leaves. According to that, plant pathologists mainly rely on naked eye prediction and a disease scoring scale to grade the disease. That leads to some problems associated with manual grading. This manual grading is not only time consuming but also not feasible. Hence an image processing-based approach to automatically grade the disease spread on plant leaves by employing Fuzzy Logic had been proposed. The results are proved to be accurate and satisfactory in contrast with manual diseases are inevitable in plants. The proposed methodology aims to model a promising disease grading system for plant leaves. The system was divided into the following steps: (1) Image acquisition (2) Image Pre-processing

(3) Color image segmentation (4) Calculating AT and AD (5) Disease grading by Fuzzy Logic.

[14] In 2014, an survey report was published[2], based on different classification techniques that could be used for plant leaf disease classification. A classification technique deals with classifying each pattern in one of the distinct classes. A classification is a technique where a leaf is classified based on its different morphological features. There are so many classification techniques such as k-Nearest Neighbor Classifier, Probabilistic Neural Network, Genetic Algorithm, Support Vector Machine, and Principal Component Analysis, Artificial neural network, Fuzzy logic. Selecting a classification method is always a difficult task because the quality of result can vary for different input data. Plant leaf disease classifications have wide applications in various fields such as in biological research, in Agriculture etc. The paper provides an overview of different classification techniques used for plant leaf disease classification. In 2012, an article was published[3] having a detailed description on definition of disease, types of diseases, symptoms and causes of most commonly observed plant diseases. One article was published by Michigan University[4] regarding the threats caused due to diseases. Various conditions for disease development had been discussed there. An overview of major disease-causing organisms and the effect of diseases caused by them was given



# CHAPTER THREE

## 3. METHODOLOGY AND SYSTEM BLOCK DIAGRAM

### 3.1 COMPONENTS & PACKAGES

The following packages are implemented in our project.

#### 3.1.1 TENSORFLOW

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017.[10] While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).[11] TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the

Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.[12]

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript.[13]

In Jan 2019, Google announced TensorFlow 2.0.[14] It became officially available in Sep 2019.[15]

In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.[16]

### **Tensor processing unit (TPU)**

In May 2016, Google announced its Tensor processing unit (TPU), an application-specific integrated circuit (a hardware chip) built specifically for machine learning and tailored for TensorFlow. TPU is a programmable AI accelerator designed to provide high throughput of low-precision arithmetic (e.g., 8-bit), and oriented toward using or running models rather than training them. Google announced they had been running TPUs inside their data centers for more than a year, and had found them to deliver an order of magnitude better-optimized performance per watt for machine learning.[17]

In May 2017, Google announced the second-generation, as well as the availability of the TPUs in Google Compute Engine.[18] The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs, provide up to 11.5 petaflops.

In May 2018, Google announced the third-generation TPUs delivering up to 420 teraflops of performance and 128 GB high bandwidth memory (HBM). Cloud TPU v3 Pods offer 100+ petaflops of performance and 32 TB HBM.[19]

In February 2018, Google announced that they were making TPUs available in beta on the Google Cloud Platform.[20]

## **TensorFlow execution model**

### **Graphs**

Machine learning can get complex quickly, and deep learning models can become large. For many model graphs, you need distributed training to be able to iterate within a reasonable time frame. And, you'll typically want the models you develop to deploy to multiple platforms.

With the current version of TensorFlow, you write code to build a computation graph, then execute it. The graph is a data structure that fully describes the computation you want to perform. This has lots of advantages:

- It's portable, as the graph can be executed immediately or saved to use later, and it can run on multiple platforms: CPUs, GPUs, TPUs, mobile, embedded. Also, it can be deployed to production without having to depend on any of the code that built the graph, only the runtime necessary to execute it.
- It's transformable and optimizable, as the graph can be transformed to produce a more optimal version for a given platform. Also, memory or compute optimizations can be performed and trade-offs made between

them. This is useful, for example, in supporting faster mobile inference after training on larger machines.

- Support for distributed execution

TensorFlow's high-level APIs, in conjunction with computation graphs, enable a rich and flexible development environment and powerful production capabilities in the same framework.

### **Eager execution**

An upcoming addition to TensorFlow is eager execution, an imperative style for writing TensorFlow. When you enable eager execution, you will be executing TensorFlow kernels immediately, rather than constructing graphs that will be executed later.

#### **Four major reasons:**

- You can inspect and debug intermediate values in your graph easily.
- You can use Python control flow within TensorFlow APIs—loops, conditionals, functions, closures, etc.
- Eager execution should make debugging more straightforward.
- Eager's "define-by-run" semantics will make building and training dynamic graphs easy.

Once you are satisfied with your TensorFlow code running eagerly, you can convert it to a graph automatically. This will make it easier to save, port, and distribute your graphs.

### **Performance and benchmarking**

TensorFlow has high standards around measurement and transparency. The team has developed a set of detailed benchmarks and has been very careful to

include all necessary details to reproduce. We've not yet run comparative benchmarks, but would welcome for others to publish comprehensive and reproducible benchmarks.

There's a section of the TensorFlow site with information specifically for performance-minded developers. Optimization can often be model-specific, but there are some general guidelines that can often make a big difference.

### **TensorFlow's open source models**

The TensorFlow team has open sourced a large number of models. You can find them in the tensorflow/models repo. For many of these, the released code includes not only the model graph, but also trained model weights. This means that you can try such models out of the box, and you can tune many of them further using a process called transfer learning.

Here are just a few of the recently released models (there are many more):

- The Object Detection API: It's still a core machine learning challenge to create accurate machine learning models capable of localizing and identifying multiple objects in a single image. The recently open sourced TensorFlow Object Detection API has produced state-of-the-art results (and placed first in the COCO detection challenge).
- tf-seq2seq: Google previously announced Google Neural Machine Translation (GNMT), a sequence-to-sequence (seq2seq) model that is now used in Google Translate production systems. tf-seq2seq is an open source seq2seq framework in TensorFlow that makes it easy to experiment with seq2seq models and achieve state-of-the-art results.
- ParseySaurus is a set of pretrained models that reflect an upgrade to SyntaxNet. The new models use a character-based input representation

and are much better at predicting the meaning of new words based both on their spelling and how they are used in context. They are much more accurate than their predecessors, particularly for languages where there can be dozens of forms for each word and many of these forms might never be observed during training, even in a very large corpus

- **Multistyle Pastiche Generator from the Magenta Project:** "Style transfer" is what's happening under the hood with those fun apps that apply the style of a painting to one of your photos. This Magenta model extends image style transfer by creating a single network that can perform more than one stylization of an image, optionally at the same time. (Try playing with the sliders for the dog images in this blog post.)

## **Transfer learning**

Many of the TensorFlow models include trained weights and examples that show how you can use them for transfer learning, e.g. to learn your own classifications. You typically do this by deriving information about your input data from the penultimate layer of a trained model—which encodes useful abstractions—then use that as input to train your own much smaller neural net to predict your own classes. Because of the power of the learned abstractions, the additional training typically does not require large data sets.

For example, you can use transfer learning with the Inception image classification model to train an image classifier that uses your specialized image data.

## **Using TensorFlow on mobile devices**

Mobile is a great use case for TensorFlow—mobile makes sense when there is a poor or missing network connection or where sending continuous data to a

server would be too expensive. But, once you've trained your model and you're ready to start using it, you don't want the on-device model footprint to be too big.

TensorFlow is working to help developers make lean mobile apps, both by continuing to reduce the code footprint and by supporting quantization.

(And although it's early days, see also Accelerated Linear Algebra [XLA], a domain-specific compiler for linear algebra that optimizes TensorFlow computations.)

One of the TensorFlow projects, MobileNet, is developing a set of computer vision models that are particularly designed to address the speed/accuracy trade-offs that need to be considered on mobile devices or in embedded applications. The MobileNet models can be found in the TensorFlow models repo as well.

One of the newer Android demos, TF Detect, uses a MobileNet model trained using the Tensorflow Object Detection API.

## **The TensorFlow ecosystem**

The TensorFlow ecosystem includes many tools and libraries to help you work more effectively. Here are a few.

### **TensorBoard**

TensorBoard is a suite of web applications for inspecting, visualizing, and understanding your TensorFlow runs and graphs. You can use TensorBoard to view your TensorFlow model graphs and zoom in on the details of graph subsections.

You can plot metrics like loss and accuracy during a training run; show histogram visualizations of how a tensor is changing over time; show additional data, like images; collect runtime metadata for a run, such as total memory usage and tensor shapes for nodes; and more.

TensorBoard works by reading TensorFlow files that contain summary information about the training process. You can generate these files when running TensorFlow jobs.

You can use TensorBoard to compare training runs, collect runtime stats, and generate histograms.

A particularly mesmerizing feature of TensorBoard is its embeddings visualizer. Embeddings are ubiquitous in machine learning, and in the context of TensorFlow, it's often natural to view tensors as points in space, so almost any TensorFlow model will give rise to various embeddings.

## **Datalab**

Jupyter notebooks are an easy way to interactively explore your data, define TensorFlow models, and kick off training runs. If you're using Google Cloud Platform tools and products as part of your workflow—maybe using Google Cloud Storage or BigQuery for your datasets, or Apache Beam for data preprocessing—then Google Cloud Datalab provides a Jupyter-based environment with all of these tools (and others like NumPy, pandas, scikit-learn, and Matplotlib), along with TensorFlow, preinstalled and bundled together. Datalab is open source, so if you want to further modify its notebook environment, it's easy to do.



## Facets

Machine learning's power comes from its ability to learn patterns from large amounts of data, so understanding your data can be critical to building a powerful machine learning system.

Facets is a recently released open source data visualization tool that helps you understand your machine learning datasets and get a sense of the shape and characteristics of each feature and see at a glance how the features interact with each other. For example, you can view your training and test datasets (as is done here with some Census data), compare the characteristics of each feature, and sort the features by "distribution distance."

### 3.1.2 KERAS

While deep neural networks are all the rage, the complexity of the major frameworks has been a barrier to their use for developers new to machine learning. There have been several proposals for improved and simplified high-level APIs for building neural network models, all of which tend to look similar from a distance but show differences on closer examination.

Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines.

Given that the TensorFlow project has adopted Keras as the high-level API for the upcoming TensorFlow 2.0 release, Keras looks to be a winner, if not necessarily the winner. In this article, we'll explore the principles and implementation of Keras, with an eye towards understanding why it's an improvement over low-level deep learning APIs.

Even in TensorFlow 1.12, the official Get Started with TensorFlow tutorial uses the high-level Keras API embedded in TensorFlow, `tf.keras`. By contrast, the TensorFlow Core API requires working with TensorFlow computational graphs, tensors, operations, and sessions, some of which can be hard to understand when you're just beginning to work with TensorFlow. There are some advantages to using the low-level TensorFlow Core API, mostly when debugging, but fortunately you can mix the high-level and low-level TensorFlow APIs as needed.

### **Keras principles**

Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training. Plus, Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

### **Keras back ends**

Keras proper does not do its own low-level operations, such as tensor products and convolutions; it relies on a back-end engine for that. Even though Keras supports multiple back-end engines, its primary (and default) back end is TensorFlow, and its primary supporter is Google. The Keras API comes packaged in TensorFlow as `tf.keras`, which as mentioned earlier will become the primary TensorFlow API as of TensorFlow 2.0.

To change back ends, simply edit your `$HOME/.keras/keras.json` file and specify a different back-end name, such as `theano` or `CNTK`. Alternatively, you can override the configured back end by defining the environment variable `KERAS_BACKEND`, either in your shell or in your Python code using the `os.environ["KERAS_BACKEND"]` property.

## Keras models

The Model is the core Keras data structure. There are two main types of models available in Keras: the Sequential model, and the Model class used with the functional API.

### Keras Sequential models

The Sequential model is a linear stack of layers, and the layers can be described very simply. Here is an example from the Keras documentation that uses `model.add()` to define two dense layers in a Sequential model:

```
import keras
from keras.models import Sequential
from keras.layers import Dense

#Create Sequential model with Dense layers, using the add
method
model = Sequential()
```

```
#Dense implements the operation:
#     output = activation(dot(input, kernel) + bias)
#Units are the dimensionality of the output space for the
layer,
#     which equals the number of hidden units
#Activation and loss functions may be specified by
strings or classes
model.add(Dense(units=64, activation='relu',
input_dim=100))
model.add(Dense(units=10, activation='softmax'))

#The compile method configures the model's learning
process
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

#The fit method does the training in batches
# x_train and y_train are Numpy arrays --just like in the
Scikit-Learn API.
model.fit(x_train, y_train, epochs=5, batch_size=32)

#The evaluate method calculates the losses and metrics
#     for the trained model
loss_and_metrics = model.evaluate(x_test, y_test,
batch_size=128)

#The predict method applies the trained model to inputs
#     to generate outputs
classes = model.predict(x_test, batch_size=128)
```

The comments in the code above are worth reading. It's also worth noting how little cruft there is in the actual code compared to, say, the low-level TensorFlow APIs. Each layer definition requires one line of code, the compilation (learning process definition) takes one line of code, and fitting (training), evaluating (calculating the losses and metrics), and predicting outputs from the trained model each take one line of code.

## Keras functional API

The Keras Sequential model is simple but limited in model topology. The Keras functional API is useful for creating complex models, such as multi-input/multi-output models, directed acyclic graphs (DAGs), and models with shared layers.

The functional API uses the same layers as the Sequential model but provides more flexibility in putting them together. In the functional API you define the layers first, and then create the Model, compile it, and fit (train) it. Evaluation and prediction are essentially the same as in a Sequential model, so have been omitted in the sample code below.

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)
```

```
# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels) # starts training
```

## Keras layers

In the previous examples we only used Dense layers. Keras has a wide selection of predefined layer types, and also supports writing your own layers.

Core layers include Dense (dot product plus bias), Activation (transfer function or neuron shape), Dropout (randomly set a fraction of input units to 0 at each training update to avoid overfitting), Lambda (wrap an arbitrary expression as a Layer object), and several others. Convolution layers (the use of a filter to create a feature map) run from 1D to 3D and include the most common variants, such as cropping and transposed convolution layers for each dimensionality. 2D convolution, which was inspired by the functionality of the visual cortex, is commonly used for image recognition.

Pooling (downscaling) layers run from 1D to 3D and include the most common variants, such as max and average pooling. Locally connected layers act like convolution layers, except that the weights are unshared. Recurrent layers include simple (fully connected recurrence), gated, LSTM, and others; these are useful for language processing, among other applications. Noise layers help to avoid overfitting.

## Keras datasets

Keras supplies seven of the common deep learning sample datasets via the `keras.datasets` class. That includes `cifar10` and `cifar100` small color images, IMDB movie reviews, Reuters newswire topics, MNIST handwritten digits, MNIST fashion images, and Boston housing prices.

### **Keras applications and examples**

Keras also supplies ten well-known models, called Keras Applications, pretrained against ImageNet: Xception, VGG16, VGG19, ResNet50, InceptionV3, InceptionResNetV2, MobileNet, DenseNet, NASNet, MobileNetV2TK. You can use these to predict the classification of images, extract features from them, and fine-tune the models on a different set of classes.

By the way, fine-tuning existing models is a good way to speed up training. For example, you can add layers as you wish, freeze the base layers to train the new layers, then unfreeze some of the base layers to fine-tune the training. You can freeze a layer by setting `layer.trainable = False`.

The Keras examples repository contains more than 40 sample models. They cover vision models, text and sequences, and generative models.

### **Deploying Keras**

Keras models can be deployed across a vast range of platforms, perhaps more than any other deep learning framework. That includes iOS, via CoreML (supported by Apple); Android, via the TensorFlow Android runtime; in a browser, via Keras.js and WebDNN; on Google Cloud, via TensorFlow-Serving; in a Python webapp back end; on the JVM, via DL4J model import; and on Raspberry Pi.

### 3.1.3 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. [10] It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.[11] The first stable build was released in December 2014, starting from version 1.0.[12]

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development.[13] Java is still supported, as is C++.[14]

#### **Gradle**

Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration. Gradle uses a directed acyclic graph ("DAG") to determine the order in which tasks can be run.

Gradle was designed for multi-project builds, which can grow to be quite large. It supports incremental builds by intelligently determining which parts of the build tree are up to date; any task dependent only on those parts does not need to be re-executed.



## **Graphical user interface builder**

A graphical user interface builder (or GUI builder), also known as GUI designer, is a software development tool that simplifies the creation of GUIs by allowing the designer to arrange graphical control elements (often called widgets) using a drag-and-drop WYSIWYG editor. Without a GUI builder, a GUI must be built by manually specifying each widget's parameters in source-code, with no visual feedback until the program is run.

User interfaces are commonly programmed using an event-driven architecture, so GUI builders also simplify creating event-driven code. This supporting code connects widgets with the outgoing and incoming events that trigger the functions providing the application logic.

Some graphical user interface builders, such as e.g. Glade Interface Designer, automatically generate all the source code for a graphical control element. Others, like Interface Builder, generate serialized object instances that are then loaded by the application.

### **3.1.4 KOTLIN**

Kotlin is a general purpose, free, open source, statically typed “pragmatic” programming language initially designed for the JVM (Java Virtual Machine) and Android that combines object-oriented and functional programming features. It is focused on interoperability, safety, clarity, and tooling support. Versions of Kotlin targeting JavaScript ES5.1 and native code (using LLVM) for a number of processors are in production as well.

Kotlin originated at JetBrains, the company behind IntelliJ IDEA, in 2010, and has been open source since 2012. The Kotlin team currently has more than 90 full-time members from JetBrains, and the Kotlin project on GitHub has more than 300 contributors. JetBrains uses Kotlin in many of its products including its flagship IntelliJ IDEA.

At first glance, Kotlin looks like a more concise and streamlined version of Java. Consider the screenshot above, where I have converted a Java code sample (at left) to Kotlin automatically. Notice that the mindless repetition inherent in instantiating Java variables has gone away. The Java idiom

You can see that functions are defined with the `fun` keyword, and that semicolons are now optional when newlines are present. The `val` keyword declares a read-only property or local variable. Similarly, the `var` keyword declares a mutable property or local variable.

Nevertheless, Kotlin is strongly typed. The `val` and `var` keywords can be used only when the type can be inferred. Otherwise you need to declare the type. Type inference seems to be improving with each release of Kotlin.

Have a look at the function declaration near the top of both panes. The return type in Java precedes the prototype, but in Kotlin it succeeds the prototype, demarcated with a colon as in Pascal.

It is not completely obvious from this example, but Kotlin has relaxed Java's requirement that functions be class members. In Kotlin, functions may be declared at top level in a file, locally inside other functions, as a member function inside a class or object, and as an extension function. Extension functions provide the C#-like ability to extend a class with new functionality

without having to inherit from the class or use any type of design pattern such as Decorator.

For Groovy fans, Kotlin implements builders; in fact, Kotlin builders can be type checked. Kotlin supports delegated properties, which can be used to implement lazy properties, observable properties, vetoable properties, and mapped properties.

Many asynchronous mechanisms available in other languages can be implemented as libraries using Kotlin coroutines. This includes `async/await` from C# and ECMAScript, channels and `select` from Go, and generators/`yield` from C# and Python.

### **Functional programming in Kotlin**

Allowing top-level functions is just the beginning of the functional programming story for Kotlin. The language also supports higher-order functions, anonymous functions, lambdas, inline functions, closures, tail recursion, and generics. In other words, Kotlin has all of the features and advantages of a functional language. For example, consider the following functional Kotlin idioms.

Even though Kotlin is a full-fledged functional programming language, it preserves most of the object-oriented nature of Java as an alternative programming style, which is very handy when converting existing Java code. Kotlin has classes with constructors, along with nested, inner, and anonymous inner classes, and it has interfaces like Java 8. Kotlin does not have a `new` keyword. To create a class instance, call the constructor just like a regular function. We saw that in the screenshot above.

Kotlin has single inheritance from a named superclass, and all Kotlin classes have a default superclass `Any`, which is not the same as the Java base class `java.lang.Object`. `Any` contains only three predefined member functions: `equals()`, `hashCode()`, and `toString()`.

Kotlin classes have to be marked with the `open` keyword in order to allow other classes to inherit from them; Java classes are kind of the opposite, as they are inheritable unless marked with the `final` keyword. To override a superclass method, the method itself must be marked `open`, and the subclass method must be marked `override`. This is all of a piece with Kotlin's philosophy of making things explicit rather than relying on defaults. In this particular case, I can see where Kotlin's way of explicitly marking base class members as open for inheritance and derived class members as overrides avoids several kinds of common Java errors.

### **Safety features in Kotlin**

Speaking of avoiding common errors, Kotlin was designed to eliminate the danger of null pointer references and streamline the handling of null values. It does this by making a null illegal for standard types, adding nullable types, and implementing shortcut notations to handle tests for null.

To avoid the verbose grammar normally needed for null testing, Kotlin introduces a safe call, written `?..`. For example, `b?.length` returns `b.length` if `b` is not null, and null otherwise. The type of this expression is `Int?`.

In other words, `b?.length` is a shortcut for `if (b != null) b.length else null`. This syntax chains nicely, eliminating quite a lot of prolix logic, especially when an object was populated from a series of database queries, any of which might have

failed. For instance, `bob?.department?.head?.name` would return the name of Bob's department head if Bob, the department, and the department head are all non-null.

The designers of Java thought this was a good idea, and it was a net win for toy programs, as long as the programmers implemented something sensible in the catch clause. All too often in large Java programs, however, you see code in which the mandatory catch clause contains nothing but a comment: `//todo: handle this`. This doesn't help anyone, and checked exceptions turned out to be a net loss for large programs.

## Kotlin coroutines

Coroutines in Kotlin are essentially lightweight threads. You start them with the `launch` coroutine builder in the context of some `CoroutineScope`. One of the most useful coroutine scopes is `runBlocking{}` , which applies to the scope of its code block.

## Kotlin for Android

Up until May 2017, the only officially supported programming languages for Android were Java and C++. Google announced official support for Kotlin on Android at Google I/O 2017, and starting with Android Studio 3.0 Kotlin is built into the Android development toolset. Kotlin can be added to earlier versions of Android Studio with a plug-in.

Kotlin compiles to the same byte code as Java, interoperates with Java classes in natural ways, and shares its tooling with Java. Because there is no overhead for calling back and forth between Kotlin and Java, adding Kotlin incrementally to an Android app currently in Java makes perfect sense. The few cases where the

interoperability between Kotlin and Java code lacks grace, such as Java set-only properties, are rarely encountered and easily fixed.

### 3.1.5 INCEPTION V3

Inception V3 by Google is the 3rd version in a series of Deep Learning Convolutional Architectures. Inception V3 was trained using a dataset of 1,000 classes (See the list of classes [here](#)) from the original ImageNet dataset which was trained with over 1 million training images, the Tensorflow version has 1,001 classes which is due to an additional "background" class not used in the original ImageNet. Inception V3 was trained for the ImageNet Large Visual Recognition Challenge where it was a first runner up.

### **Convolutional Neural Networks**

Convolutional neural networks are a type of deep learning neural network. These types of neural nets are widely used in computer vision and have pushed the capabilities of computer vision over the last few years, performing exceptionally better than older, more traditional neural networks; however, studies show that there are trade-offs related to training times and accuracy.

Transfer learning allows you to retrain the final layer of an existing model, resulting in a significant decrease in not only training time, but also the size of the dataset required. One of the most famous models that can be used for transfer learning is Inception V3. As mentioned above, this model was originally trained on over a million images from 1,000 classes on some very powerful machines. Being able to retrain the final layer means that you can maintain the knowledge that the model had learned during its original training

and apply it to your smaller dataset, resulting in highly accurate classifications without the need for extensive training and computational power.

### **TensorFlow -Slim image classification model library**

TF-Slim is a high-level API for TensorFlow\* that allows you to program, train and evaluate Convolutional Neural Networks. TF-Slim is a lightweight API so is well suited for lower powered devices.

#### **inception\_preprocessing.py**

The inception\_preprocessing file provides the tools required to preprocess both training and evaluation images allowing them to be used with Inception Networks.

#### **inception\_utils.py**

The inception\_utils class file utility code that is common across all Inception versions.

#### **inception\_v3.py**

The inception\_v3 file provides the code required to create an Inception V3 network.

In this file you will find the inception\_v3 function provided by TensorFlow, this function produces the exact Inception model from Rethinking the Inception Architecture for Computer Vision written by Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna.

### **Model Freezing**

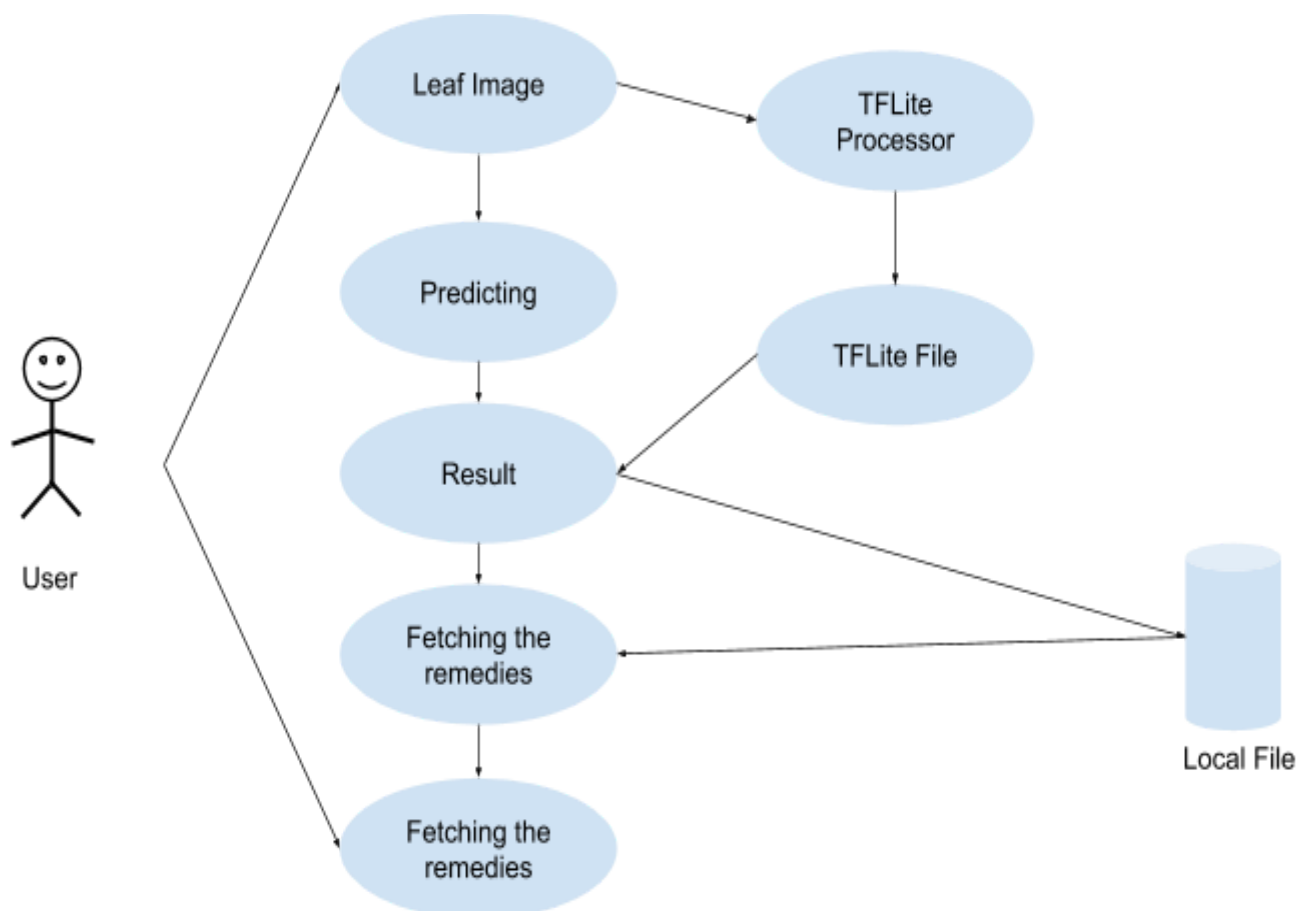
In projects that use the Intel® Neural Compute Stick (NCS/NCS2), it is required to freeze the model, a technique mostly used for deploying TensorFlow models

to mobile devices. Freezing a model basically removes unrequired/unused nodes such as training specific nodes etc. To find out more about model freezing, you can visit the [Preparing models for mobile deployment TensorFlow tutorial](#), to find the related project code you can check out the [NCS training program](#). The training program uses TF-Slim to produce a graph and uses `graph_util.convert_variables_to_constants` to create a TensorFlow GraphDef, saving it as a .pb file in the model directory.

## 3.2 SYSTEM DIAGRAM

This Java Application has been developed using the above mentioned packages. Let's see the Use Case Diagram of this Application.

### 3.2.1 USE CASE DIAGRAM

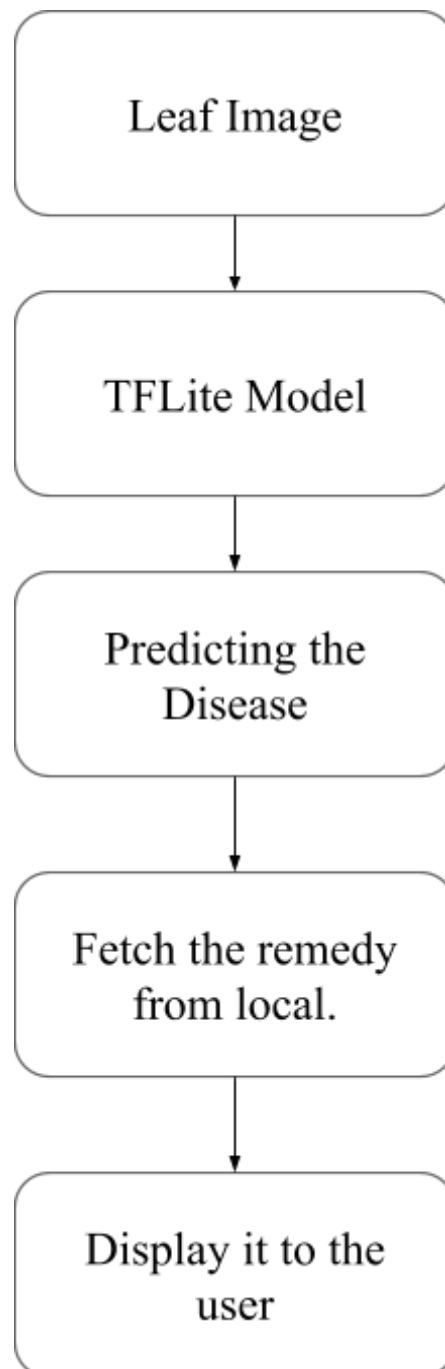




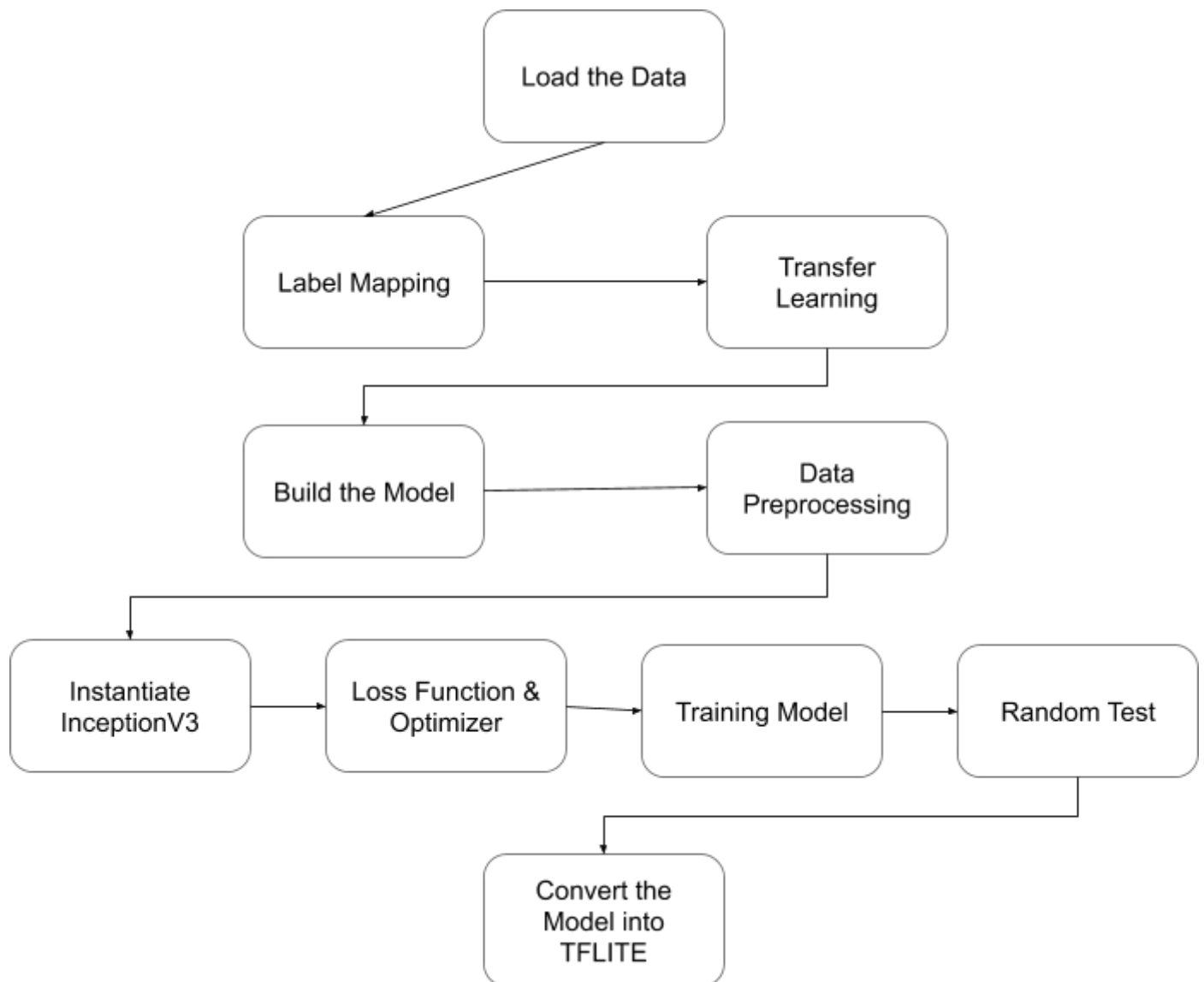
### 3.2.2 SYSTEM LEVEL DESIGN

There are 1 high level design and its broken down into two low level designs..

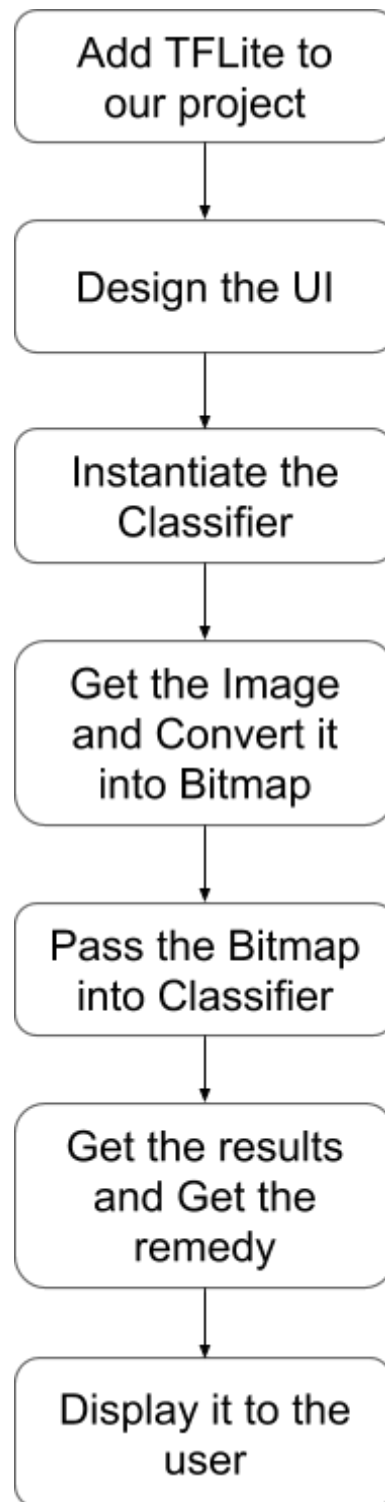
#### **High Level Design:**



## Low Level Design



## TFLite Model Building



### **Android App Implementation**

### 3.3 METHODOLOGY

The complete development of this project is explained below with the code being written.

The project is broken down into two steps:

- Building and creating a machine learning model using TensorFlow with Keras.
- Deploying the model to an Android application using TFLite.

#### Machine Learning model using Tensorflow with Keras

We designed algorithms and models to recognize species and diseases in the crop leaves by using Convolutional Neural Network. We use Colab for edit source code.

##### 3.3.1 Importing the Librairies

```
# Install nightly package for some functionalities that
aren't in alpha
!pip install tensorflow-gpu==2.0.0-beta1
# Install TF Hub for TF2
!pip install 'tensorflow-hub == 0.5'

from __future__ import absolute_import, division,
print_function, unicode_literals
import tensorflow as tf
import tensorflow_hub as hub

from tensorflow.keras.layers import Dense, Flatten,
Conv2D
from tensorflow.keras import Model
```

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
```

### 3.3.2 Loading the data

Download a public dataset of 54,305 images of diseased and healthy plant leaves collected under controlled conditions PlantVillage Dataset . The images cover 14 species of crops, including: apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry and tomato. It contains images of 17 basic diseases, 4 bacterial diseases, 2 diseases caused by mold (oomycete), 2 viral diseases and 1 disease caused by a mite. 12 crop species also have healthy leaf images that are not visibly affected by disease.

Create the training and validation directories:

```
#Load data
zip_file=tf.keras.utils.get_file(origin='https://storage.
googleapis.com/plantdata/PlantVillage.zip',
    fname='PlantVillage.zip', extract=True)
#Create the training and validation directories
data_dir = os.path.join(os.path.dirname(zip_file),
    'PlantVillage')
train_dir = os.path.join(data_dir, 'train')
validation_dir = os.path.join(data_dir, 'validation')
```

### 3.3.3 Label mapping

You'll also need to load in a mapping from category label to category name. This will give you a dictionary mapping the integer encoded categories to the actual names of the plants and diseases.

```
!wget
https://github.com/obeshor/Plant-Diseases-Detector/archiv
e/master.zip
!unzip master.zip;
import json
with
open('Plant-Diseases-Detector-master/categories.json',
'r') as f:
    cat_to_name = json.load(f)
    classes = list(cat_to_name.values())

print (classes)
```

### 3.3.4 Transfer Learning with TensorFlow hub

Select the Hub/TF2 module to use, you have a choice with inception v3 or Mobilenet

```
module_selection = ("inception_v3", 299, 2048) #@param
["(\"mobilenet_v2\", 224, 1280)", "(\"inception_v3\",
299, 2048)"] {type:"raw", allow-input: true}
handle_base, pixels, FV_SIZE = module_selection
MODULE_HANDLE = "https://tfhub.dev/google/tf2-
preview/{}/feature_vector/2".format(handle_base)
IMAGE_SIZE = (pixels, pixels)
BATCH_SIZE = 64 #@param {type:"integer"}
```

### 3.3.5 Data Preprocessing

Let's set up data generators that will read pictures in our source folders, convert them to `float32` tensors, and feed them (with their labels) to our network.

As you may already know, data that goes into neural networks should usually be normalized in some way to make it more amenable to processing by the network. In our case, we will preprocess our images by normalizing the pixel values to be in the `[0, 1]` range (originally all values are in the `[0, 255]` range). we'll need to make sure the input data is resized to 224x224 pixels or 299x299 pixels as required by the networks. You have the choice to implement image augmentation or not.

```
# Inputs are suitably resized for the selected module.
validation_datagen =
tf.keras.preprocessing.image.ImageDataGenerator(rescale=1
./255)
validation_generator =
validation_datagen.flow_from_directory(
    validation_dir,
    shuffle=False,
    seed=42,
    color_mode="rgb",
    class_mode="categorical",
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE)
do_data_augmentation = True #@param {type:"boolean"}
if do_data_augmentation:
    train_datagen =
tf.keras.preprocessing.image.ImageDataGenerator(
    rescale = 1./255,
```

```

        rotation_range=40,
        horizontal_flip=True,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        fill_mode='nearest' )
else:
    train_datagen = validation_datagen

train_generator = train_datagen.flow_from_directory(
    train_dir,
    subset="training",
    shuffle=True,
    seed=42,
    color_mode="rgb",
    class_mode="categorical",
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE)

```

### 3.3.6 Build the model

All it takes is to put a linear classifier on top of the feature\_extractor with the Hub module. For speed, we start out with a non-trainable feature\_extractor , but you can also enable fine-tuning for greater accuracy but that takes a lot of time to train the model.

```

feature_extractor = hub.KerasLayer(MODULE_HANDLE,

input_shape=IMAGE_SIZE+(3,),

output_shape=[FV_SIZE])
do_fine_tuning = False #@param {type:"boolean"}

```



```

if do_fine_tuning:
    feature_extractor.trainable = True
    # unfreeze some layers of base network for
    fine-tuning
    for layer in feature_extractor.layers[-30:]:
        layer.trainable = True

else:
    feature_extractor.trainable = False

model = tf.keras.Sequential([
    feature_extractor,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(rate=0.2),
    tf.keras.layers.Dense(train_generator.num_classes,
activation='softmax',
kernel_regularizer=tf.keras.regularizers.l2(0.0001))
])

```

### 3.3.7 Specifying Loss Function and Optimizer

```

| #Compile model specifying the optimizer learning rate
LEARNING_RATE = 0.001 #@param {type:"number"}
model.compile(
    optimizer=tf.keras.optimizers.Adam(lr=LEARNING_RATE),
    loss='categorical_crossentropy',
    metrics=['accuracy'])

```

### 3.3.8 Training Model

Train model using validation dataset for validate each steps. After 10 epochs, we get 94% for accuracy, you can improve this more than 99% using fine-tuning

```
EPOCHS=10 #@param {type:"integer"}
STEPS_EPOCHS =
train_generator.samples//train_generator.batch_size
VALID_STEPS=validation_generator.samples//validation_gene
rator.batch_size
history = model.fit_generator(
    train_generator,
    steps_per_epoch=STEPS_EPOCHS,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=VALID_STEPS)
```

### 3.3.9 Checking Performance

Plot training and validation, accuracy and loss

```
import matplotlib.pyplot as plt
import numpy as np
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(EPOCHS)
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
```

```

plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.show()

```

### 3.3.10 Random test

Random five sample images from validation dataset and predict:

```

# Import OpenCV
import cv2
# Utility
import itertools
import random
from collections import Counter
from glob import iglob
def load_image(filename):
    img = cv2.imread(os.path.join(data_dir,
validation_dir, filename))
    img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]))
)
    img = img /255

```

```

    return img
def predict(image):
    probabilities = model.predict(np.asarray([img]))[0]
    class_idx = np.argmax(probabilities)

    return {classes[class_idx]: probabilities[class_idx]}
for idx, filename in
enumerate(random.sample(validation_generator.filenames,
5)):
    print("SOURCE: class: %s, file: %s" %
(os.path.split(filename)[0], filename))

    img = load_image(filename)
    prediction = predict(img)
    print("PREDICTED: class: %s, confidence: %f" %
(list(prediction.keys())[0],
list(prediction.values())[0]))
    plt.imshow(img)
    plt.figure(idx)
    plt.show()

```

### 3.3.11 Convert model to TensorFlow Lite

```

# convert the model to TFLite
!mkdir "tflite_models"
TFLITE_MODEL = "tflite_models/plant_disease_model.tflite"

# Get the concrete function from the Keras model.
run_model = tf.function(lambda x : reloaded(x))

# Save the concrete function.

```

```

concrete_func = run_model.get_concrete_function(
    tf.TensorSpec(model.inputs[0].shape,
model.inputs[0].dtype)
)

# Convert the model to standard TensorFlow Lite model
converter =
tf.lite.TFLiteConverter.from_concrete_functions([concrete
_func])
converted_tflite_model = converter.convert()
open(TFLITE_MODEL, "wb").write(converted_tflite_model)

```

### 3.3.12 Add TFLite model in our Android Project

First — load the model in our Android project, we put `plant_disease_model.tflite` and `plant_labels.txt` into `assets/` directory. `plant_disease_model.tflite` is the result of our previous colab notebook. We need to add TFLite dependency to `app/build.gradle` file.

```

implementation 'org.tensorflow:tensorflow-lite:1.14.0'
aaptOptions {
    noCompress "tflite"
}

```

3.3.13 Create classifier class to load our model and read the file with labels:

```

class Classifier(assetManager: AssetManager, modelPath:
String, labelPath: String, inputSize: Int) {
    private var INTERPRETER: Interpreter

```

```

private var LABEL_LIST: List<String>
private val INPUT_SIZE: Int = inputSize
private val PIXEL_SIZE: Int = 3
private val IMAGE_MEAN = 0
private val IMAGE_STD = 255.0f
private val MAX_RESULTS = 3
private val THRESHOLD = 0.4f

...
init {
    INTERPRETER =
Interpreter(loadModelFile(assetManager, modelPath))
    LABEL_LIST = loadLabelList(assetManager,
labelPath)
}
private fun loadModelFile(assetManager:
AssetManager, modelPath: String): MappedByteBuffer {
    val fileDescriptor =
assetManager.openFd(modelPath)
    val inputStream =
FileInputStream(fileDescriptor.fileDescriptor)
    val fileChannel = inputStream.channel
    val startOffset = fileDescriptor.startOffset
    val declaredLength =
fileDescriptor.declaredLength
    return
fileChannel.map(FileChannel.MapMode.READ_ONLY,
startOffset, declaredLength)
}

private fun loadLabelList(assetManager: AssetManager,
labelPath: String): List<String> {
    return
assetManager.open(labelPath).bufferedReader().useLines {
it.toList() }
}

```

```

    }
    ...
}

```

Where Recognition is our humble result data class :

```

data class Recognition(
    var id: String = "",
    var title: String = "",
    var confidence: Float = 0F
) {
    override fun toString(): String {
        return "Title = $title, Confidence = $confidence)"
    }
}

```

When we have an instance of Interpreter, we need to convert the preprocessed bitmap into ByteBuffer then we create a method that will take an image as an argument and return a list of labels with assigned probabilities to them:

```

fun recognizeImage(bitmap: Bitmap):
List<Classifier.Recognition> {
    val scaledBitmap =
Bitmap.createScaledBitmap(bitmap, INPUT_SIZE, INPUT_SIZE,
false)
    val byteBuffer =
convertBitmapToByteBuffer(scaledBitmap)
    val result = Array(1) {
FloatArray(LABEL_LIST.size) }
    INTERPRETER.run(byteBuffer, result)
}

```

```

        return getSortedResult(result)
    }

```

Here's how we convert a bitmap into ByteBuffer:

```

private fun convertBitmapToByteBuffer(bitmap: Bitmap):
ByteBuffer {
    val byteBuffer = ByteBuffer.allocateDirect(4 *
INPUT_SIZE * INPUT_SIZE * PIXEL_SIZE)
    byteBuffer.order(ByteOrder.nativeOrder())
    val intValues = IntArray(INPUT_SIZE * INPUT_SIZE)

    bitmap.getPixels(intValues, 0, bitmap.width, 0,
0, bitmap.width, bitmap.height)
    var pixel = 0
    for (i in 0 until INPUT_SIZE) {
        for (j in 0 until INPUT_SIZE) {
            val `val` = intValues[pixel++]

            byteBuffer.putFloat((((`val`.shr(16) and
0xFF) - IMAGE_MEAN) / IMAGE_STD))
            byteBuffer.putFloat((((`val`.shr(8) and
0xFF) - IMAGE_MEAN) / IMAGE_STD))
            byteBuffer.putFloat((((`val` and 0xFF) -
IMAGE_MEAN) / IMAGE_STD))
        }
    }
    return byteBuffer
}

```

Here is most of the MainActivity code, we'll use in our app:



```

class MainActivity : AppCompatActivity() {
    private lateinit var mClassifier: Classifier
    private lateinit var mBitmap: Bitmap

    private val mCameraRequestCode = 0
    private val mGalleryRequestCode = 2

    private val mInputSize = 224
    private val mModelPath = "plant_disease_model.tflite"
    private val mLabelPath = "plant_labels.txt"
    private val mSamplePath = "soybean.JPG"

    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        requestedOrientation =
        ActivityInfo.SCREEN_ORIENTATION_PORTRAIT
        setContentView(R.layout.activity_main)
        mClassifier = Classifier(assets, mModelPath,
        mLabelPath, mInputSize)

        resources.assets.open(mSamplePath).use {
            mBitmap = BitmapFactory.decodeStream(it)
            mBitmap = Bitmap.createScaledBitmap(mBitmap,
        mInputSize, mInputSize, true)
            mPhotoImageView.setImageBitmap(mBitmap)
        }

        mCameraButton.setOnClickListener {
            val callCameraIntent =
        Intent(MediaStore.ACTION_IMAGE_CAPTURE)
            startActivityForResult(callCameraIntent,

```

```

mCameraRequestCode)
    }

    mGalleryButton.setOnClickListener {
        val callGalleryIntent =
Intent(Intent.ACTION_PICK)
        callGalleryIntent.type = "image/*"
        startActivityForResult(callGalleryIntent,
mGalleryRequestCode)
    }
    mDetectButton.setOnClickListener {
        val results =
mClassifier.recognizeImage(mBitmap).firstOrNull()
        mResultTextView.text= results?.title+"\n
Confidence:"+results?.confidence

    }
}
...
}

```

Where `scaleImage` method allows us to resize the image because our model expects the exact input shape (224x224 pixels), therefore we need to rescale a delivered bitmap to fit into these constraints:

```

fun scaleImage(bitmap: Bitmap?): Bitmap {
    val originalWidth = bitmap!!.width
    val originalHeight = bitmap.height
    val scaleWidth = mInputSize.toFloat() /
originalWidth
    val scaleHeight = mInputSize.toFloat() /
originalHeight
    val matrix = Matrix()

```

```

matrix.postScale(scaleWidth, scaleHeight)
return Bitmap.createBitmap(bitmap, 0, 0,
originalWidth, originalHeight, matrix, true)
}

```

## CHAPTER FOUR

### 4. RESULT & DISCUSSION

Activities Firefox Web Browser Fri Mar 27 5:59 PM Plant Diseases Detection with TF2 V2.ipynb - Collaboratory - Mozilla Firefox

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
- Export as saved model and convert to TFLite
- CONCLUSION
- Section

Importing the Libraries

```

[1] 1 # Install nightly package for some functionalities that aren't in alpha
2 !pip install tf-nightly-gpu-2.0-preview
3
4 # Install TF Hub for TF2
5 !pip install 'tensorflow-hub == 0.4'
6
ERROR: Could not find a version that satisfies the requirement tf-nightly-gpu-2.0-preview (from versions: none)
ERROR: No matching distribution found for tf-nightly-gpu-2.0-preview
Collecting tensorflow-hub==0.4
  Downloading https://files.pythonhosted.org/packages/10/5c/6f3698513cf1cd73ba5ea6a6c665d213adf9de59b34f362f27be8b126f/tensorflow_hub-0.4.0-py2.py3-none-any.whl (75KB)
Requirement already satisfied: protobuf>=3.4.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-hub==0.4) (3.10.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-hub==0.4) (1.12.0)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-hub==0.4) (1.18.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from tensorflow-hub==0.4) (46.0.0)
Installing collected packages: tensorflow-hub
  Found existing installation: tensorflow-hub 0.7.0
  Uninstalling tensorflow-hub-0.7.0:
    Successfully uninstalled tensorflow-hub-0.7.0
  Successfully installed tensorflow-hub-0.4.0

[2] 1 from __future__ import absolute_import, division, print_function, unicode_literals
2
3
4 import tensorflow as tf
5 #tf.logging.set_verbosity(tf.logging.ERROR)
6 #tf.enable_eager_execution()
7
8 import tensorflow_hub as hub
9 import os
10 from tensorflow.keras.layers import Dense, Flatten, Conv2D
11 from tensorflow.keras import Model
12 from tensorflow.keras.preprocessing.image import ImageDataGenerator
13 from tensorflow.keras.optimizers import Adam
14 from tensorflow.keras import layers
15 #from keras import optimizers
16
17
18

```

Activities Firefox Web Browser Fri Mar 27 5:59 PM

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V2.ipynb#scrollTo=IP...

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

+ Code + Text Copy to Drive

```
[2] 5 #tf.logging.set_verbosity(tf.logging.ERROR)
6 #tf.enable_eager_execution()
7
8 import tensorflow_hub as hub
9 import os
10 from tensorflow.keras.layers import Dense, Flatten, Conv2D
11 from tensorflow.keras import Model
12 from tensorflow.keras.preprocessing.image import ImageDataGenerator
13 from tensorflow.keras.optimizers import Adam
14 from tensorflow.keras import layers
15 #from keras import optimizers
16
17
18
```

The default version of TensorFlow in Colab will switch to TensorFlow 2.x on the 27th of March, 2020. We recommend you [upgrade now](#) or ensure your notebook will continue to use TensorFlow 1.x via the `!tensorflow_version 1.x` magic: [more info](#).

```
[3] 1 # verify TensorFlow version
2
3 print("Version: ", tf.__version__)
4 print("Eager mode: ", tf.executing_eagerly())
5 print("Hub version: ", hub.__version__)
6 print("GPU is", "available" if tf.test.is_gpu_available() else "NOT AVAILABLE")
7
```

Version: 1.15.2  
Eager mode: False  
Hub version: 0.4.0  
GPU is available

Load the data

We will download a public dataset of 54,305 images of diseased and healthy plant leaves collected under controlled conditions ([PlantVillage Dataset](#)). The images cover 14 species of crops, including: apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry and tomato. It contains images of 17 basic diseases, 4 bacterial diseases, 2 diseases caused by mold (comycete), 2 viral diseases and 1 disease caused by a mite. 12 crop species also have healthy leaf images that are not visibly affected by disease. Then store the downloaded zip file to the "/tmp/" directory.

We'll need to make sure the input data is resized to 224x224 or 229x229 pixels as required by the networks.

Activities Firefox Web Browser Fri Mar 27 6:00 PM

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V2.ipynb#scrollTo=IPe...

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

+ Code + Text Copy to Drive

Load the data

We will download a public dataset of 54,305 images of diseased and healthy plant leaves collected under controlled conditions ([PlantVillage Dataset](#)). The images cover 14 species of crops, including: apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry and tomato. It contains images of 17 basic diseases, 4 bacterial diseases, 2 diseases caused by mold (comycete), 2 viral diseases and 1 disease caused by a mite. 12 crop species also have healthy leaf images that are not visibly affected by disease. Then store the downloaded zip file to the "/tmp/" directory.

we'll need to make sure the input data is resized to 224x224 or 229x229 pixels as required by the networks.

```
1 zip_file = tf.keras.utils.get_file(origin='https://storage.googleapis.com/plantdata/PlantVillage.zip',
2   train_dir = os.path.join(data_dir, 'train')
3   fname='PlantVillage.zip', extract=True)
4
5 ... Downloading data from https://storage.googleapis.com/plantdata/PlantVillage.zip
55833952/856839884 [=====] - ETA: 5s
```

Prepare training and validation dataset

Create the training and validation directories

```
1 data_dir = os.path.join(os.path.dirname(zip_file), 'PlantVillage')
2 train_dir = os.path.join(data_dir, 'train')
3 validation_dir = os.path.join(data_dir, 'validation')
4
5 import time
6 import os
7 from os.path import exists
8
9 def count_dir(dir, counter=0):
10     'returns number of files in dir and subdirs'
11     for pack in os.walk(dir):
12         for f in pack[2]:
13             counter += 1
14     return dir + " : " + str(counter) + " files"
15
16 print('total images for training :', count(train_dir))
17 print('total images for validation :', count(validation_dir))
```

Activities Firefox Web Browser Fri Mar 27 6:01 PM

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V2.ipynb#scrollTo=IPe...

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

+ Code + Text Copy to Drive

```
[8] 1 !wget https://github.com/navayuvan-sb/Plant-Diseases-Detector/archive/master.zip
2 !unzip master.zip
3
4 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/drawable/ic_launcher_background.xml
5 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/layout/
6 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/layout/activity_main.xml
7 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-anydpi-v26/
8 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-anydpi-v26/ic_launcher.xml
9 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-anydpi-v26/ic_launcher_round.xml
10 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-hdpi/
11 extracting: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-hdpi/ic_launcher.png
12 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-hdpi/ic_launcher_foreground.png
13 extracting: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-hdpi/ic_launcher_round.png
14 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-mdpi/
15 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-mdpi/ic_launcher.png
16 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-mdpi/ic_launcher_foreground.png
17 extracting: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-mdpi/ic_launcher_round.png
18 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xhdpi/
19 extracting: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xhdpi/ic_launcher.png
20 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xhdpi/ic_launcher_foreground.png
21 extracting: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xhdpi/ic_launcher_round.png
22 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xxhdpi/
23 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xxhdpi/ic_launcher.png
24 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xxhdpi/ic_launcher_foreground.png
25 extracting: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/mipmap-xxhdpi/ic_launcher_round.png
26 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/values/
27 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/values/colors.xml
28 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/values/ic_launcher_background.xml
29 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/values/strings.xml
30 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/main/res/values/styles.xml
31 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/test/
32 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/test/java/
33 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/test/java/isoroma/
34 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/test/java/isoroma/com/
35 creating: Plant-Diseases-Detector-master/GreenDoctor/app/src/test/java/isoroma/com/plantdetector/
36 inflating: Plant-Diseases-Detector-master/GreenDoctor/app/src/test/java/isoroma/com/plantdetector/ExampleUnitTest.kt
37 inflating: Plant-Diseases-Detector-master/GreenDoctor/build.gradle
38 inflating: Plant-Diseases-Detector-master/GreenDoctor/gradle.properties
39 creating: Plant-Diseases-Detector-master/GreenDoctor/gradle/
40 creating: Plant-Diseases-Detector-master/GreenDoctor/gradle/wrapper/
41 inflating: Plant-Diseases-Detector-master/GreenDoctor/gradle/wrapper/gradle-wrapper.jar
42 inflating: Plant-Diseases-Detector-master/GreenDoctor/gradle/wrapper/gradle-wrapper.properties
43 inflating: Plant-Diseases-Detector-master/GreenDoctor/gradlew
```

Activities Firefox Web Browser

Fri Mar 27 6:01 PM

### Plant Diseases Detection with TF2 V2.ipynb - Collaboratory - Mozilla Firefox

Options

[https://colab.research.google.com/github/newayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\\_Diseases\\_Detection\\_with\\_TF2\\_V2.ipynb#scrollTo=Pej...](https://colab.research.google.com/github/newayuvan-sb/Plant-Diseases-Detector/blob/master/Plant_Diseases_Detection_with_TF2_V2.ipynb#scrollTo=Pej...)

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

+ Code + Text Copy to Drive

```
[9] 6 print(classes)

['Apple_Apple_scab', 'Apple_Black_rot', 'Apple_Cedar_apple_rust', 'Apple_healthy', 'Blueberry_healthy', 'Cherry_(including_sour)_Powdery_mildew', 'Cherry_(including...)']

[10] 1 print('Number of classes:', len(classes))

Number of classes: 38
```

Select the Hub/TF2 module to use

```
[11] 1 module_selection = ("inception_v3", 299, 2048) #@param [{"(\\\"mobilenet_v2\\\", 224, 1
2 handle_base, pixels, FV_SIZE = module_selection
3 MODULE_HANDLE = "https://tfhub.dev/google/tf2-preview/{}/feature_vector/2".format(h
4 IMAGE_SIZE = (pixels, pixels)
5 print("Using {} with input size {} and output dimension {}".format(
6 MODULE_HANDLE, IMAGE_SIZE, FV_SIZE))
7
8 BATCH_SIZE = 64 #@param [type:"integer"]

module_selection: ("inception_v3", 299, 2048)
BATCH_SIZE: 64
```

Using [https://tfhub.dev/google/tf2-preview/inception\\_v3/feature\\_vector/2](https://tfhub.dev/google/tf2-preview/inception_v3/feature_vector/2) with input size (299, 299) and output dimension 2048

Data Preprocessing

Lets set up data generators that will read pictures in our source folders, convert them to float32 tensors, and feed them (with their labels) to our network.

As you may already know, data that goes into neural networks should usually be normalized in some way to make it more amenable to processing by the network. (It is uncommon to feed raw pixels into a convnet.) In our case, we will preprocess our images by normalizing the pixel values to be in the [0, 1] range (originally all values are in the [0, 255] range).

```
[12] 1 # Inputs are suitably resized for the selected module. Dataset augmentation (i.e.,
2
3 validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255
4 validation_generator = validation_datagen.flow_from_directory(
5     validation_dir,
6     shuffle=False,
7     seed=42,
```

do\_data\_augmentation: ☒

The screenshot shows a Jupyter Notebook interface in a Firefox browser. The notebook is titled "Plant Diseases Detection with TF2 V2.ipynb". The left sidebar shows the notebook's structure, including sections like "Table of contents", "TensorFlow Lite End-to-End Android Application", "Machine Learning model using Tensorflow with Keras", "Importing the Libraries", "Load the data", "Prepare training and validation dataset", "Label mapping", "Select the Hub/TF2 module to use", "Data Preprocessing", "Build the model", "Specify Loss Function and Optimizer", "Train Model", "Check Performance", "Random test", "Export as saved model and convert to TFLite", and a "CONCLUSION" section. The main area displays a code cell with the following Python code:

```
[12]:
3 validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255
4 validation_generator = validation_datagen.flow_from_directory(
5     validation_dir,
6     shuffle=False,
7     seed=42,
8     color_mode="rgb",
9     class_mode="categorical",
10    target_size=IMAGE_SIZE,
11    batch_size=BATCH_SIZE)
12
13 do_data_augmentation = True #@param {type:'boolean'}
14 if do_data_augmentation:
15     train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
16         rescale = 1./255,
17         rotation_range=0,
18         horizontal_flip=True,
19         width_shift_range=0.2,
20         height_shift_range=0.2,
21         shear_range=0.2,
22         zoom_range=0.2,
23         fill_mode='nearest' )
24 else:
25     train_datagen = validation_datagen
26
27 train_generator = train_datagen.flow_from_directory(
28     train_dir,
29     subset="training",
30     shuffle=True,
31     seed=42,
32     color_mode="rgb",
33     class_mode="categorical",
34     target_size=IMAGE_SIZE,
35     batch_size=BATCH_SIZE)
36
```

Below the code cell, the output is displayed:

```
Found 10861 images belonging to 38 classes.
Found 43444 images belonging to 38 classes.
```



Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V2.ipynb#scrollTo=IP...

Plant Diseases Detection with TF2 V2.ipynb

Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

Code

```
[13] output_shape=[FV_SIZE])

WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_variable_ops.py:1781: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.i
Instructions for updating:
If using Keras pass '*' constraint arguments to layers.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resource_variable_ops.py:1781: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.i
Instructions for updating:
If using Keras pass '*' constraint arguments to layers.
```

do\_fine\_tuning: ☐

```
[14] 1 do_fine_tuning = False #@param {type:'boolean'}
2 if do_fine_tuning:
3     feature_extractor.trainable = True
4     # unfreeze some layers of base network for fine-tuning
5     for layer in base_model.layers[-38:]:
6         layer.trainable = True
7
8 else:
9     feature_extractor.trainable = False
10
```

```
[15] 1 print("Building model with", MODULE_HANDLE)
2 model = tf.keras.Sequential([
3     feature_extractor,
4     tf.keras.layers.Flatten(),
5     tf.keras.layers.Dense(512, activation='relu'),
6     tf.keras.layers.Dropout(rate=0.2),
7     tf.keras.layers.Dense(train_generator.num_classes, activation='softmax',
8                             kernel_regularizer=tf.keras.regularizers.l2(0.0001))
9 ])
10 #model.build((None,)+IMAGE_SIZE+(3,))
11
12 model.summary()
```

Building model with [https://tfhub.dev/google/tf2-preview/inception\\_v3/feature\\_vector/2](https://tfhub.dev/google/tf2-preview/inception_v3/feature_vector/2)

WARNING:tensorflow:Entity <bound method KerasLayer.call of <tensorflow.hub.keras\_layer.KerasLayer object at 0x7f4265e614ab>> could not be transformed and will be executed as-is.

WARNING:tensorflow:Entity <bound method KerasLayer.call of <tensorflow.hub.keras\_layer.KerasLayer object at 0x7f4265e614ab>> could not be transformed and will be executed as-is.

WARNING:tensorflow:Entity <bound method KerasLayer.call of <tensorflow.hub.keras\_layer.KerasLayer object at 0x7f4265e614ab>> could not be transformed and will be executed as-is. Please re

Model: "sequential"

Layer (type)	Output Shape	Param #
keras_layer (KerasLayer)	(None, 2048)	21802784
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 38)	19494
<b>Total params:</b>	<b>22,871,366</b>	
<b>Trainable params:</b>	<b>1,860,582</b>	
<b>Non-trainable params:</b>	<b>21,882,784</b>	

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V2.ipynb#scrollTo=IP...

Plant Diseases Detection with TF2 V2.ipynb

Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

Code

```
[15] 12 model.summary()
```

Building model with [https://tfhub.dev/google/tf2-preview/inception\\_v3/feature\\_vector/2](https://tfhub.dev/google/tf2-preview/inception_v3/feature_vector/2)

WARNING:tensorflow:Entity <bound method KerasLayer.call of <tensorflow.hub.keras\_layer.KerasLayer object at 0x7f4265e614ab>> could not be transformed and will be executed as-is.

WARNING:tensorflow:Entity <bound method KerasLayer.call of <tensorflow.hub.keras\_layer.KerasLayer object at 0x7f4265e614ab>> could not be transformed and will be executed as-is.

WARNING:tensorflow:Entity <bound method KerasLayer.call of <tensorflow.hub.keras\_layer.KerasLayer object at 0x7f4265e614ab>> could not be transformed and will be executed as-is. Please re

Model: "sequential"

Layer (type)	Output Shape	Param #
keras_layer (KerasLayer)	(None, 2048)	21802784
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 38)	19494
<b>Total params:</b>	<b>22,871,366</b>	
<b>Trainable params:</b>	<b>1,860,582</b>	
<b>Non-trainable params:</b>	<b>21,882,784</b>	

Specify Loss Function and Optimizer

```
[16] 1 #Compile model specifying the optimizer learning rate
2
3 LEARNING_RATE = 0.001 #@param {type:'number'}
4
5 model.compile(
6     optimizer=tf.keras.optimizers.Adam(lr=LEARNING_RATE),
7     loss='categorical_crossentropy',
8     metrics=['accuracy'])
9
10
```

LEARNING\_RATE: 0.001

Train Model

train model using validation dataset for validate each steps

Activities Firefox Web Browser Fri Mar 27 6:01 PM

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

Train Model

train model using validation dataset for validate each steps

```

1 EPOCHS=5 #@param {type:"integer"}
2
3
4 history = model.fit_generator(
5     train_generator,
6     steps_per_epoch=train_generator.samples//train_generator.batch_size,
7     epochs=EPOCHS,
8     validation_data=validation_generator,
9     validation_steps=validation_generator.samples//validation_generator.batch_size)

```

Epoch 1/5  
44/678 [>.....] - ETA: 12:37 - loss: 2.9380 - acc: 0.3363

Check Performance

Plot training and validation accuracy and loss

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 acc = history.history['accuracy']
5 val_acc = history.history['val_accuracy']
6
7 loss = history.history['loss']
8 val_loss = history.history['val_loss']
9
10 epochs_range = range(EPOCHS)
11

```

Activities Firefox Web Browser Fri Mar 27 6:01 PM

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
  - Export as saved model and convert to TFLite
- CONCLUSION
- Section

```

14 plt.plot(epochs_range, acc, label='Training Accuracy')
15 plt.plot(epochs_range, val_acc, label='Validation Accuracy')
16 plt.legend(loc='lower right')
17 plt.title('Training and Validation Accuracy')
18 plt.xlabel('Accuracy (training and validation)')
19 plt.ylabel('Training Steps')
20
21 plt.subplot(1, 2, 2)
22 plt.plot(epochs_range, loss, label='Training Loss')
23 plt.plot(epochs_range, val_loss, label='Validation Loss')
24 plt.legend(loc='upper right')
25 plt.title('Training and Validation Loss')
26 plt.xlabel('Loss (training and validation)')
27 plt.ylabel('Training Steps')
28 plt.show()

```



Activities Firefox Web Browser Fri Mar 27 6:01 PM

Plant Diseases Detection with TF2 V2.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V2.ipynb#scrollTo=Ipej

Plant Diseases Detection with TF2 V2.ipynb

File Edit View Insert Runtime Tools Help Unsaved changes since 5:59 PM

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Select the Hub/TF2 module to use
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
- Export as saved model and convert to TFLite
- CONCLUSION
- Section

Random sample images from validation dataset and predict

```

1 # Import OpenCV
2 import cv2
3
4 # Utility
5 import itertools
6 import random
7 from collections import Counter
8 from glob import glob
9
10
11 def load_image(filename):
12     img = cv2.imread(os.path.join(data_dir, validation_dir, filename))
13     img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]))
14     img = img / 255
15     return img
16
17
18 def predict(image):
19     probabilities = model.predict(np.asarray([img]))[0]
20     class_idx = np.argmax(probabilities)
21
22     return (classes[class_idx], probabilities[class_idx])
23
24
25 for idx, filename in enumerate(random.sample(validation_generator.filenames, 5)):
26     print("SOURCE: class: %s, file: %s" % (os.path.split(filename)[0], filename))
27
28     img = load_image(filename)
29     prediction = predict(img)
30     print("PREDICTED: class: %s, confidence: %f" % (list(prediction.keys())[0], list(prediction.values())[0]))
31     plt.imshow(img)
32     plt.figure(idx)
33     plt.show()
34
35 ...

```

Activities Firefox Web Browser Fri Mar 27 6:03 PM

Plant Diseases Detection with TF2 V4.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V4.ipynb#scrollTo=Q...

Plant Diseases Detection with TF2 V4.ipynb

File Edit View Insert Runtime Tools Help Last edited on March 10

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using Tensorflow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Setup Image shape and batch size
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test
- Export as saved model and convert to TFLite
- Convert Model to TFLite
- CONCLUSION
- Section

14 plt.plot(epochs\_range, acc, label='Training Accuracy')

15 plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

16 plt.legend(loc='lower right')

17 plt.title('Training and Validation Accuracy')

18 plt.xlabel('Accuracy (training and validation)')

19 plt.ylabel('Training Steps')

20

21 plt.subplot(1, 2, 2)

22 plt.plot(epochs\_range, loss, label='Training Loss')

23 plt.plot(epochs\_range, val\_loss, label='Validation Loss')

24 plt.legend(loc='upper right')

25 plt.title('Training and Validation Loss')

26 plt.xlabel('Loss (training and validation)')

27 plt.ylabel('Training Steps')

28 plt.show()

Accuracy (training and validation)

Training Steps

Loss (training and validation)

Training Steps

Activities Firefox Web Browser Fri Mar 27 6:03 PM

Plant Diseases Detection with TF2 V4.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V4.ipynb#scrollTo=C...

Plant Diseases Detection with TF2 V4.ipynb

File Edit View Insert Runtime Tools Help Last edited on March 10


Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Setup Image shape and batch size
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test**
  - Export as saved model and convert to TFLite
  - Convert Model to TFLite
  - CONCLUSION
- Section

```
[ ] 22
    23     return (classes[class_idx]: probabilities[class_idx])

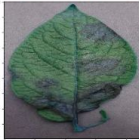
[ ] 1 for idx, filename in enumerate(random.sample(validation_generator filenames, 5)):
    2     print("SOURCE: class: %s, file: %s" % (os.path.split(filename)[0], filename))
    3
    4     img = load_image(filename)
    5     prediction = predict(img)
    6     print("PREDICTED: class: %s, confidence: %f" % (list(prediction.keys())[0], list(prediction.values())[0]))
    7     plt.imshow(img)
    8     plt.figure(idx)
    9     plt.show()
```

SOURCE: class: Tomato\_Leaf\_Mold, file: Tomato\_Leaf\_Mold/22cb45f2-6368-4e94-8deb-939d1f6b85ca\_Crnl\_L\_Mold 7084.JPG  
PREDICTED: class: Tomato\_Leaf\_Mold, confidence: 0.998060



<Figure size 432x288 with 0 Axes>

SOURCE: class: Potato\_Late\_blight, file: Potato\_Late\_blight/2d736aa6-79a6-42b0-9e92-d859bcd72824\_PS\_LB 5244.JPG  
PREDICTED: class: Potato\_Late\_blight, confidence: 0.999595



<Figure size 432x288 with 0 Axes>

Activities Firefox Web Browser Fri Mar 27 6:03 PM

Plant Diseases Detection with TF2 V4.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V4.ipynb#scrollTo=C...

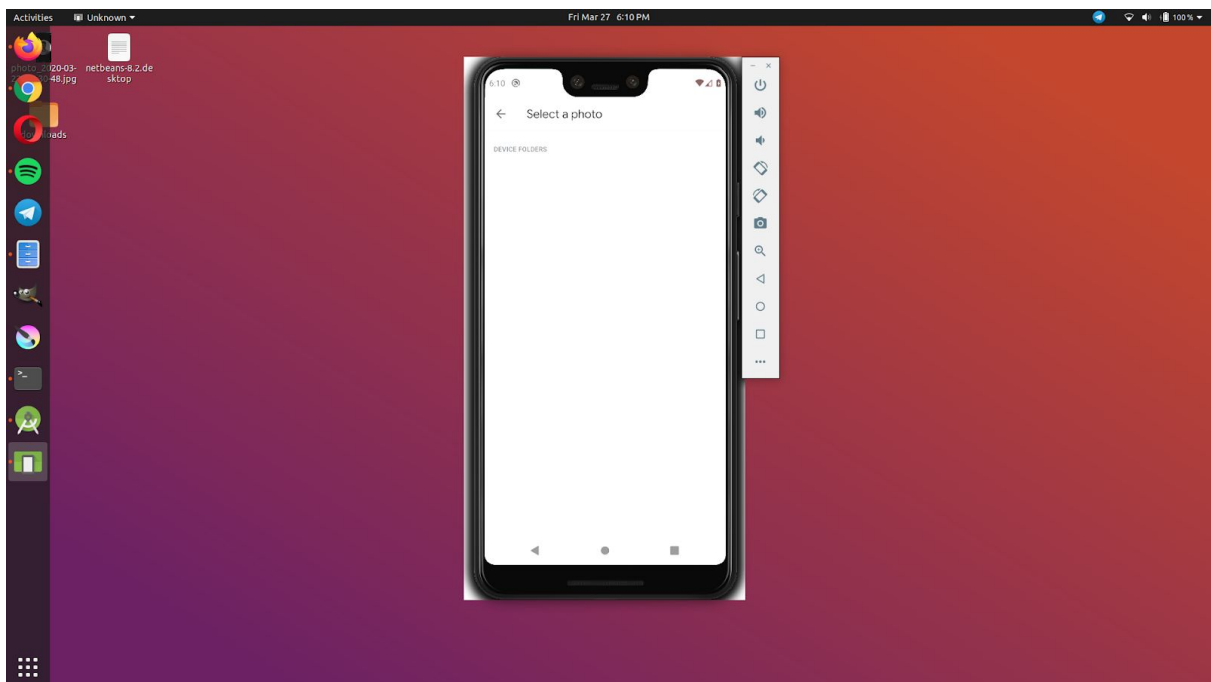
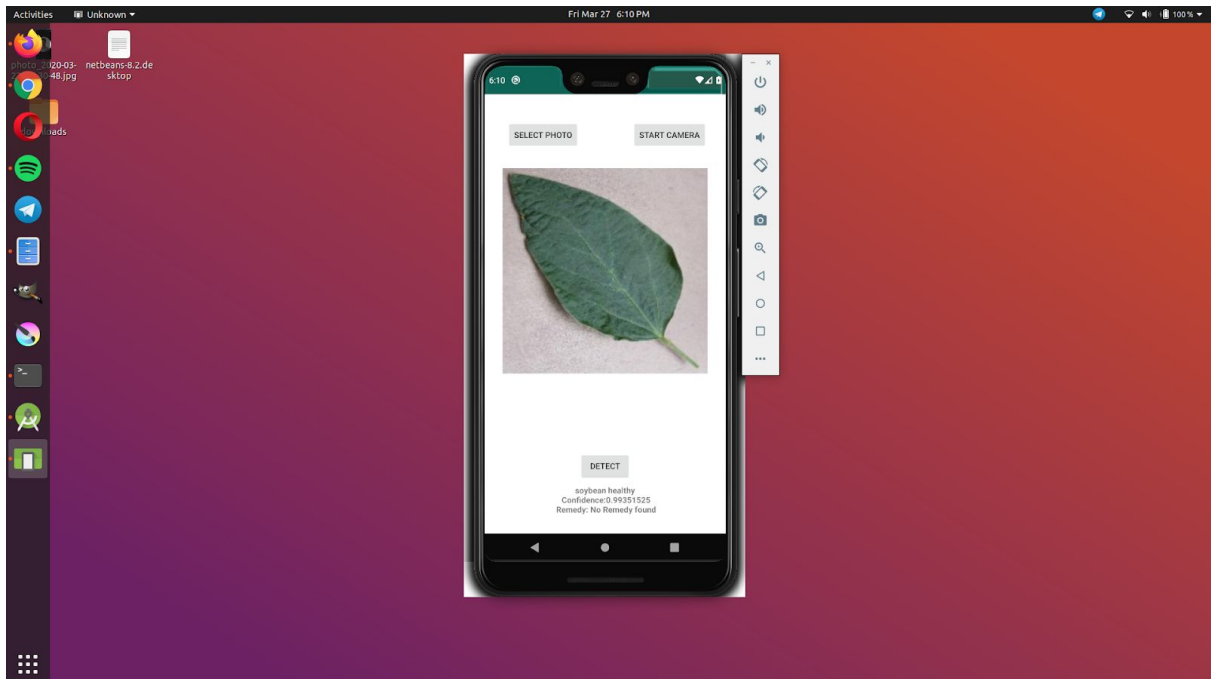
Plant Diseases Detection with TF2 V4.ipynb

File Edit View Insert Runtime Tools Help Last edited on March 10

Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Setup Image shape and batch size
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test**
  - Export as saved model and convert to TFLite
  - Convert Model to TFLite
  - CONCLUSION
- Section

```
[ ] 125
    126
    127
    128
    129
    130
    131
    132
    133
    134
    135
    136
    137
    138
    139
    140
    141
    142
    143
    144
    145
    146
    147
    148
    149
    150
    151
    152
    153
    154
    155
    156
    157
    158
    159
    160
    161
    162
    163
    164
    165
    166
    167
    168
    169
    170
    171
    172
    173
    174
    175
    176
    177
    178
    179
    180
    181
    182
    183
    184
    185
    186
    187
    188
    189
    190
    191
    192
    193
    194
    195
    196
    197
    198
    199
    200
    201
    202
    203
    204
    205
    206
    207
    208
    209
    210
    211
    212
    213
    214
    215
    216
    217
    218
    219
    220
    221
    222
    223
    224
    225
    226
    227
    228
    229
    230
    231
    232
    233
    234
    235
    236
    237
    238
    239
    240
    241
    242
    243
    244
    245
    246
    247
    248
    249
    250
    251
    252
    253
    254
    255
    256
    257
    258
    259
    260
    261
    262
    263
    264
    265
    266
    267
    268
    269
    270
    271
    272
    273
    274
    275
    276
    277
    278
    279
    280
    281
    282
    283
    284
    285
    286
    287
    288
    289
    290
    291
    292
    293
    294
    295
    296
    297
    298
    299
    300
    301
    302
    303
    304
    305
    306
    307
    308
    309
    310
    311
    312
    313
    314
    315
    316
    317
    318
    319
    320
    321
    322
    323
    324
    325
    326
    327
    328
    329
    330
    331
    332
    333
    334
    335
    336
    337
    338
    339
    340
    341
    342
    343
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368
    369
    370
    371
    372
    373
    374
    375
    376
    377
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811
    1812
    1813
    1814
    1815
    1816
    1817
    1818
    1819
    1820
    1821
    1822
    1823
    1824
    1825
    1826
    1827
    1828
    1829
    1830
    1831
    1832
    1833
    1834
    1835
    1836
    1837
    1838
    1839
    1840
    184
```



Activities Firefox Web Browser Fri Mar 27 6:03 PM

Plant Diseases Detection with TF2 V4.ipynb - Colaboratory - Mozilla Firefox

https://colab.research.google.com/github/navayuvan-sb/Plant-Diseases-Detector/blob/master/Plant\_Diseases\_Detection\_with\_TF2\_V4.ipynb#scrollTo=C...

Plant Diseases Detection with TF2 V4.ipynb

File Edit View Insert Runtime Tools Help Last edited on March 10


Table of contents

- TensorFlow Lite End-to-End Android Application
- Machine Learning model using TensorFlow with Keras
  - Importing the Libraries
  - Load the data
  - Prepare training and validation dataset
  - Label mapping
  - Setup Image shape and batch size
  - Data Preprocessing
  - Build the model
  - Specify Loss Function and Optimizer
  - Train Model
  - Check Performance
  - Random test**
  - Export as saved model and convert to TFLite
  - Convert Model to TFLite
  - CONCLUSION
- Section

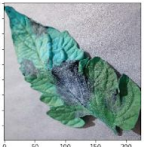
Code Text Copy to Drive

Connect Editing

SOURCE: class: Raspberry\_healthy, file: Raspberry\_healthy/f48dd477-0530-4619-a99a-03a51f053dfe\_Mary\_HL\_9167.JPG  
PREDICTED: class: Raspberry\_healthy, confidence: 0.829325

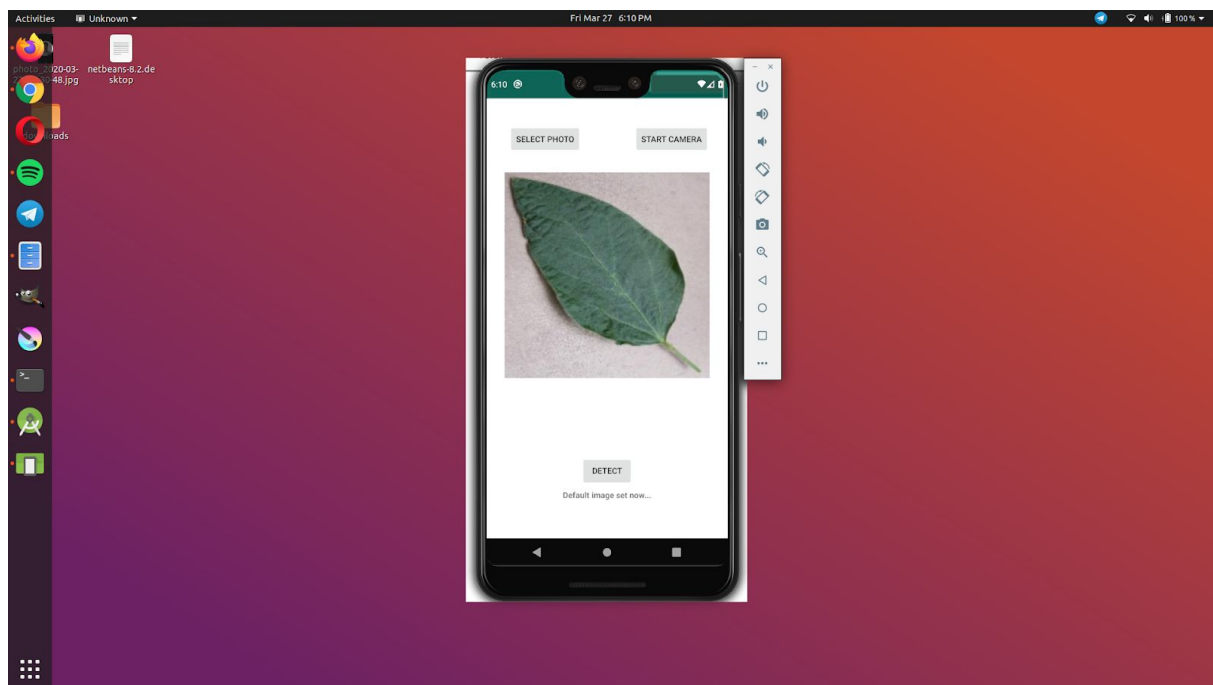


<Figure size 432x288 with 0 Axes>  
SOURCE: class: Tomato\_Late\_blight, file: Tomato\_Late\_blight/ed6a8fb9-9fa8-4d3d-a727-676ec58d147a\_RS\_Late.B\_7036.JPG  
PREDICTED: class: Tomato\_Late\_blight, confidence: 0.911693



Convert Model to TFLite

```
[ ] 1 # convert the model to TFLite
2 mkdir "tflite_models"
3 TFLITE_MODEL = "tflite_models/plant_disease_model.tflite"
4
5
6 # Get the concrete function from the Keras model.
7 run_model = tf.function(lambda x : reloaded(x))
```



# CHAPTER FIVE

## 5. CONCLUSION

The use of automated monitoring and management systems are gaining increasing demand with technological advancement. In the agricultural field loss of yield mainly occurs due to widespread disease. Mostly the detection and identification of the disease is noticed when the disease advances to severe stage. Therefore, causing the loss in terms of yield, time and money. The proposed system is capable of detecting the disease at the earlier stage as soon as it occurs on the leaf. Hence saving the loss and reducing the dependency on the expert to a certain extent is possible. It can provide the help for a person having less knowledge about the disease. Depending on these goals, we have to extract the features corresponding to the disease.

# CHAPTER SIX

## 6. REFERENCE

- [1].Ashqar, B. A., & AbuNaser S.S. (2018).Image-based tomato leaves diseases detection using deep learning.
- [2].Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband.
- [3].Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., & Hughes, D. P. (2017). Deep learning for image-based cassava disease detection. *Frontiers in plant science*, 8, 1852.
- [4].Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145, 311-318.
- [5].Ghaiwat, S. N., & Arora, P. (2014). Detection and classification of plant leaf diseases using image processing techniques: a review. *International Journal of Recent Advances in Engineering & Technology*, 2(3), 1-7.
- [6].Fang, Y., & Ramasamy, R. P. (2015). Current and prospective methods for plant disease detection. *Biosensors*, 5(3), 537-561.
- [7].Omrani, E., Khoshnevisan, B., Shamshirband, S., Saboohi, H., Anuar, N. B., & Nasir, M. H. N. M. (2014). Potential of radial basis function-based support vector regression for apple disease detection. *Measurement*, 55, 512-519.
- [8].Kawasaki, Y., Uga, H., Kagiwada, S., & Iyatomi, H. (2015, December). Basic study of automated diagnosis of viral plant diseases using convolutional

neural networks. In International Symposium on Visual Computing (pp. 638-645). Springer, Cham.

[9].Singh, V., & Misra, A. K. (2017). Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information processing in Agriculture*, 4(1), 41-49.

[10].Mahlein, A. K. (2016). Plant disease detection by imaging sensors—parallels and specific demands for precision agriculture and plant phenotyping. *Plant disease*, 100(2), 241-251.