# Plant Disease Classification Using TensorFlow Transfer Learning

**A Project Report**

***Submitted by***

## SAMYAK JHAVERI, B039

## SHREY KAMDAR, B044

***Under the Guidance of***

## Prof. Ameyaa Biwalkar

***in partial fulfillment for the award of the degree of***

## B.TECH

**COMPUTER ENGINEERING**

At



**MUKESH PATEL SCHOOL TECHNOLOGY MANAGEMENT AND ENGINEERING, NMIMS, MUMBAI**

**MARCH 2020**

## DECLARATION

I **Samyak Jhaveri**, **Shrey Kamdar** Roll No. **B039, B044** B.Tech (Computer Engineering), VI semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Students: \_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Names: **Samyak Jhaveri, Shrey Kamdar**

Roll Nos.: **B039, B044**

Place: Mumbai

Date:

## CERTIFICATE

This is to certify that the project entitled “**Plant Disease Classification using TensorFlow Transfer Learning**” is the bonafide work carried out by **Samyak Jhaveri (B039) and Shrey Kamdar (B044)** of B.Tech, MPSTME (NMIMS), Mumbai, during the VI semester of the academic year **2019-2020**, in partial fulfillment of the requirements for the Course Programming Language.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Ameyaa Biwalkar

Internal Mentor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner 1 Examiner 2

## Table of contents

**CHAPTER NO. TITLE PAGE NO.**

1. INTRODUCTION
2. SOFTWARES AND API USED WITH DESCRIPTION
3. METHODS IMPLEMENTED
4. SCREENSHOTS
5. CONCLUSION & FUTURE SCOPE
6. SOCIETAL APPLICATION

**ACKONWLEDGEMENTS**

We would like to thank Prof. Ameyaa Biwalkar for guiding us through the process of the development of this project and providing us with the outside-in perspective of what could be done in the project to test it, build-upon it and present the right data to convey the capabilities of the project.

**INTRODUCTION**

Early and accurate detection and diagnosis of plant disease are key factors in plant production and the reduction of both qualitative and quantitative losses in crop yield. Non-destructive, sensor-based methods support and expand upon and/or molecular approaches to plant disease assessment. The most relevant application of sensor-based analyses are precision agriculture and plant phenotyping.

Computer vision and machine-learning solutions offer great opportunities for the automatic recognition of sick plants by visual inspection of damaged leaves.

An automatic plant-disease detection system provides clear benefit in monitoring of large fields, as this is the only approach that provides a chance to discover diseases at an early stage. The computer vision core system inspects image flow from cameras, detects diseased leaves, and performs classification. The inspection results can be provided in various ways.

We’ve attempted to build a classifier system using Google’s TensorFlow API to train multiple model architectures on a readily available plant leaf dataset provided by Kaggle. Once trained, these model architectures are tested using previously unseen data (unseen by the training algorithm) to make classifications from the labelled data they were trained on. Parameters such as confidence and accuracy of classification are the ones that are optimized and used as rubrics of assessment.

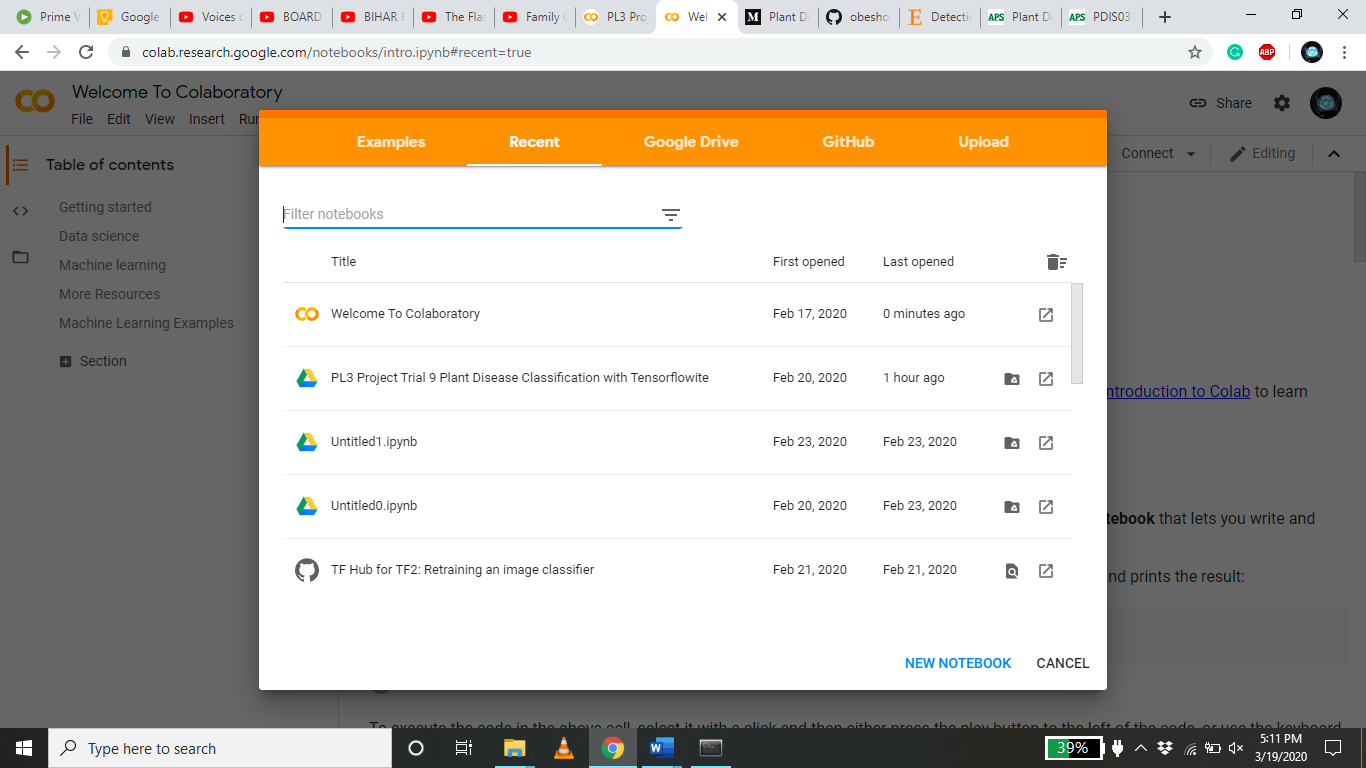
One of the most important characteristics of this project is that it uses a concept called transfer learning to train the models which makes the entire process much faster with very little loss in efficiency/accuracy.

**SOFTWARES AND APIs USED**

1. **Google Colab**

In the initial stages of the project, it was observed that even loading the dataset into the Jupyter Notebook was too resource-intensive in terms of time and computation requirements. Training the models architectures in the coming steps would prove to be quite impractical to execute on locally on the PC.

Google’s Colab platform allowed us to make use of Google’s computation power to load, train, test and evaluate the data and neural network model architectures online with the use of hardware optimized for machine learning - GPUs and TPUs.



1. **TensorFlow, TensorFlow Hub**

The TensorFlow API offers multiple levels of abstraction to choose from in building and training models. TensorFlow also gave us flexibility and control over the execution of the models in which we used high-level Keras APIs.

*Example:*

import tensorflow as tf  
mnist = tf.keras.datasets.mnist  
  
(x\_train, y\_train),(x\_test, y\_test) = mnist.load\_data()  
x\_train, x\_test = x\_train / 255.0, x\_test / 255.0  
  
model = tf.keras.models.Sequential([  
  tf.keras.layers.Flatten(input\_shape=(28, 28)),  
  tf.keras.layers.Dense(128, activation='relu'),  
  tf.keras.layers.Dropout(0.2),  
  tf.keras.layers.Dense(10, activation='softmax')  
])  
  
model.compile(optimizer='adam',  
              loss='sparse\_categorical\_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(x\_train, y\_train, epochs=5)  
model.evaluate(x\_test, y\_test)

TensorFlow Hub is a library for the publication, discovery, and consumption of reusable parts of machine learning models. A *module* is a self-contained piece of a TensorFlow graph, along with its weights and assets, that can be reused across different tasks in a process known as transfer learning. Transfer learning can:

* Train a model with a smaller dataset
* Improve generalization, and
* Speed up training.

*Example:*

  !pip install "tensorflow\_hub>=0.6.0"  
  !pip install "tensorflow>=2.0.0"  
  
  import tensorflow as tf  
  import tensorflow\_hub as hub  
  
  module\_url = "https://tfhub.dev/google/nnlm-en-dim128/2"  
  embed = hub.KerasLayer(module\_url)  
  embeddings = embed(["A long sentence.", "single-word",  
                      "http://example.com"])  
  print(embeddings.shape)  #(3,128)

**METHODS IMPLEMENTED**

**Gathering the Data**

The ‘PlantVillage Dataset’ is a labelled, public dataset of 54,305 images of diseased and healthy plants collected under control conditions. These images cover 14 species of crops, including: apple, blueberry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry and tomato. It contains images of 17 basic disease, 4 bacterial diseases, 2 diseases caused by mold, 2 viral diseases and 1 disease caused by mite. 12 crop species also have healthy leaf images that are not visible affected by disease.

This dataset was split into Training and Testing datasets.

**Label Mapping**

A supervised-learning model cannot make classifications of images without the help of labels assigned to those images. Hence it is necessary to load a mapping from category label to category name. This will give a dictionary mapping the integer encoded to the actual names of the plants and diseases.

**Transfer Learning**

Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem.

In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest.

Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error.

The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme. This may be useful when the first related problem has a lot more labeled data than the problem of interest and the similarity in the structure of the problem may be useful in both contexts. The objective is to take advantage of data from the first setting to extract information that may be useful when learning or even when directly making predictions in the second setting.

A range of high-performing models have been developed for image classification and demonstrated on the annual [ImageNet Large Scale Visual Recognition Challenge](http://www.image-net.org/challenges/LSVRC/), or ILSVRC.

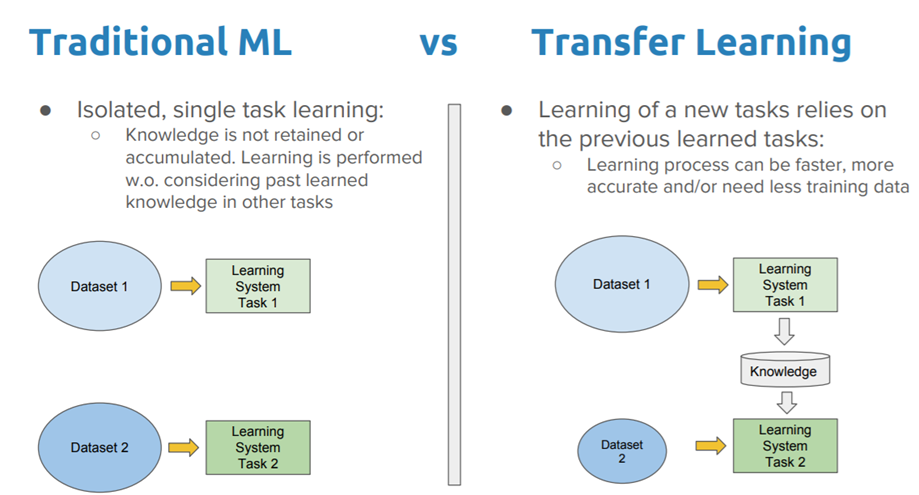
This challenge, often referred to simply as [ImageNet](http://image-net.org/), given the source of the image used in the competition, has resulted in a number of innovations in the architecture and training of convolutional neural networks. In addition, many of the models used in the competitions have been released under a permissive license.

These models can be used as the basis for transfer learning in computer vision applications.

This is desirable for a number of reasons, not least:

* **Useful Learned Features**: The models have learned how to detect generic features from photographs, given that they were trained on more than 1,000,000 images for 1,000 categories.
* **State-of-the-Art Performance**: The models achieved state of the art performance and remain effective on the specific image recognition task for which they were developed.
* **Easily Accessible**: The model weights are provided as free downloadable files and many libraries provide convenient APIs to download and use the models directly.

The model weights can be downloaded and used in the same model architecture using a range of different deep learning libraries, including Keras.



**Data Preprocessing**

The images were converted into ‘float32’ tensors and fed into the neural network, along with their labels. But before passing through the neural networks, the data must be normalized\* to make it more amenable to processing by the network. In this case, the pixel values have been normalized to the range between 0 and 1(from the original range of 0 to 255).

(\**Normalization* is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0. When the data has multiple attributes having different values on different scales, the models are likely to perform poorly. So, they are normalized to bring all the attributes on the same scale.)

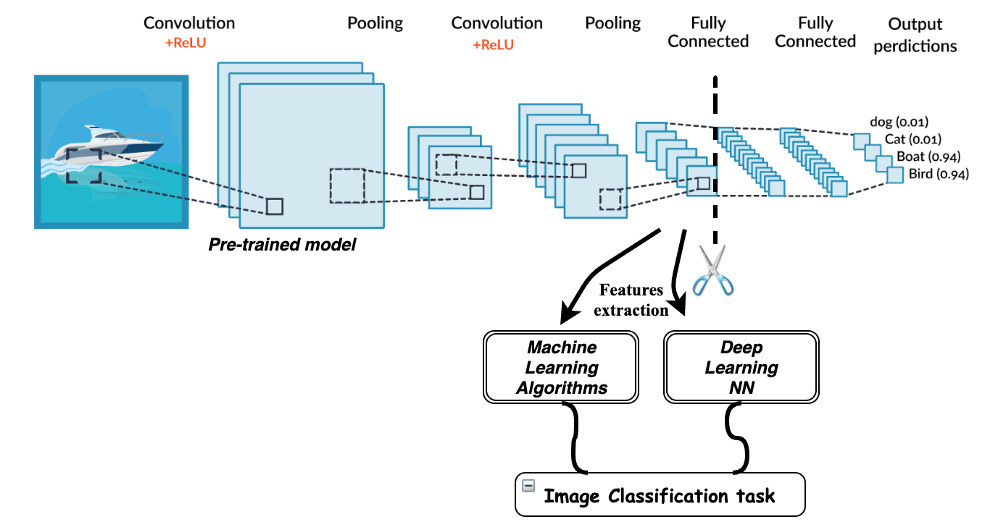
**Building the Model and Specifying Loss Function and Optimizer**

A linear classifier is put on top of the feature\_extractor with the TensorFlow Hub module.

The model is then compiled and the loss function and optimizers are specified. Here we have used ‘adam’ as the optimizer and ‘categorical crossentropy’ as the loss function, along with ‘accuracy’ as the metric of assessment of performance.

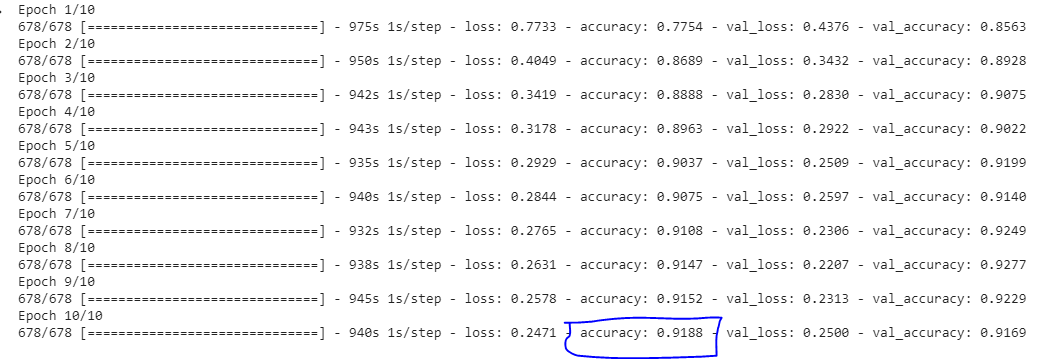
(*Loss function* – Algorithms learn by means of a loss function. It’s a method of evaluating how well specific algorithm models the given data. If predictions deviate too much from actual results, loss function would cough up a very large number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction.

*Optimizer* - Optimization algorithms are used to minimize, or maximize an **objective**function O**(x)**which is simply a mathematical function dependent on the Model’s internal **learnable** **parameters** which are used in computing the target values **(Y)** from the set of predictors(**X**) used in the model.)



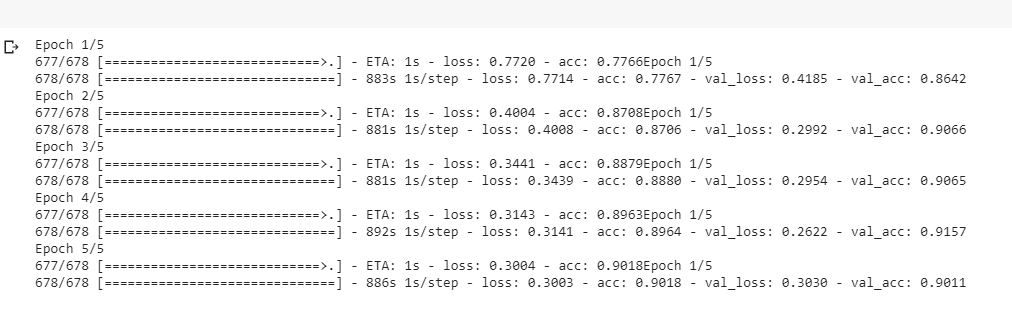
**Training the Model**

The model is trained on the testing part of the dataset and an accuracy of 92% after training for 10 epochs.

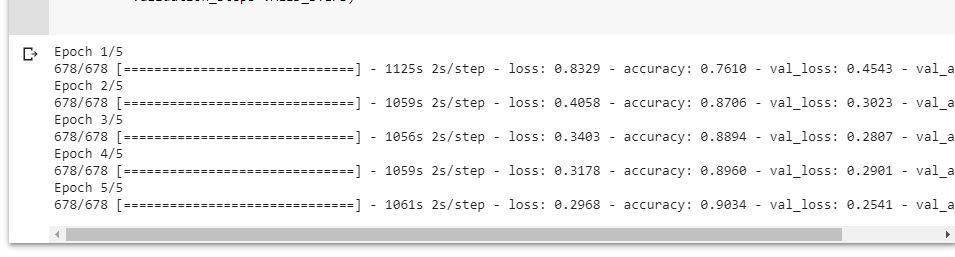


**Testing combinations of different activation functions in the neural-network layers**

1. ReLU and softmax (Original Combination, 5 EPOCHS)



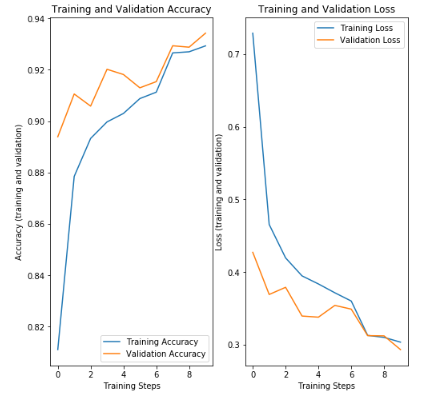
1. Softmax and softmax (5 EPOCHS)



It can be concluded that both the activation functions work nearly similar on a 5 EPOCH trial.

**Checking Performance and Testing the Trained Model with Random images**

The validation, accuracy and loss are plotted:

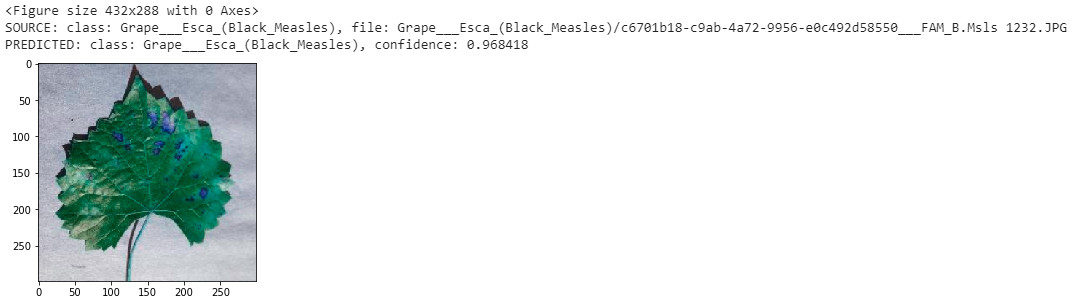


Here, it can be inferred that as the training steps progressed, both the training and texting accuracy climbed and the loss plummeted.

The results of the random test were as follows:







**CONCLUSION**

With the help of transfer learning and Google Colab’s GPUs, this project successfully achieves a high accuracy rate at classifying the plant species and diseases without compromising speed at which the models can be trained or tested.

**SOCIETAL APPLICATIONS**

This project highlights the usefulness of two main applications:

1. The use of computer vision and machine learning in detecting and diagnosing plant species and diseases that can make systems such as precision farming and plant phenotyping possible.
2. The versatility of transfer learning, especially in image-based classification models that can be easily maneuvered to work for a wide range of applications besides on vegetation.