National College of Ireland

Automated leaf disease detection and treatment recommendation using Transfer Learning

MSc Research Project
Data Analytics

Sachin Malpe
Student ID: x17164044

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sachin Malpe
Student ID:	x17164044
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	12/08/2019
Project Title:	Automated leaf disease detection and treatment recommendation using Transfer Learning
Word Count:	8084
Page Count:	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automated leaf disease detection and treatment recommendation using Transfer Learning

Sachin Malpe
x17164044

Abstract

Current automated leaf disease detection models do not generalize well on unseen data and require enormous amounts of time, computational resources and data to build. On the other hand, not much has been explored on how to provide treatment recommendations after disease detection. Deep learning (DL) methods applied using transfer learning tends to drastically reduce the amount of data, time and resources needed to build this type of model which generalizes well on new data. Hence, this study utilizes pre-trained DL architectures such as VGG16, VGG19 and SqueezeNet as feature extractors along with Extreme Gradient Boosting (XGBoost) and Support Vector Machines (SVM) as the classifier to detect 27 distinct classes of (plant, disease) combination which also includes healthy plants. Out of all the hybrid models, VGG16+SVM performed the best with the highest f1 score of 96.16%, with one of the fastest overall computational time recorded using only CPU's. This proves the viability of this approach, which can be further researched to improve this model. Treatment recommendation is provided after detection of disease by building a rest API providing the disease and treatment details in a single JSON object.

1 Introduction

The health of the plants is important for any agricultural society. It determines the quality and quantity of crop production. Annual losses in crops in the United States due to fungal diseases amounts to more than $200 billion, and around $600 million are spent only in the usage of fungicides (González-Fernández et al.; 2010). Similarly, as mentioned in (Rangaswami and Mahadevan; 1998), United States lost around $4 billion due to plant diseases in the year 1980. Thus, early detection of diseases in plants became extremely crucial to reduce the ecological, economic and biological losses (Pantazi et al.; 2019; Dhingra et al.; 2018). Due to this, a lot of research was conducted to detect diseases using DNA/RNA (Eun et al.; 2002), thermography (Chaerle et al.; 2006; Mahlein; 2016), techniques involving fluorescence imaging (Kuckenberg et al.; 2009) and many biological studies. However, manual identification of these diseases is expensive and can only be done by a person who is highly skilled and experienced in this field (Pantazi et al.; 2019). Moreover, these methods were highly inefficient, laborious and even time-consuming (Dhingra et al.; 2018). As a result, automated identification of plant diseases became a necessity to minimize the cost of their identification and improve the accuracy of the method (Dhingra et al.; 2018).

Recent advancements in computational speeds have led to rapid developments in the field of image processing using machine learning techniques (Ferentinos; 2018). Hence, classifiers such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naive Bayes (NB) and Random Forests (RF) (Bandi et al.; 2013; Gavhale et al.; 2014) have been utilized for detecting and classifying diseases in plants. With neural networks giving higher performance when compared with traditional classifiers, especially in the field of computer vision (Pantazi et al.; 2019; Ferentinos; 2018), many researchers have started using deep learning for plant disease detection successfully.

In (Mohanty et al.; 2016), Convolutional Neural Networks (CNN) and transfer learning are used providing state of the art results of 99.35% for disease detection in plants for 38 classes (combination of plants and diseases). However, as the authors themselves mentioned that this method took enormous amounts of time to train on multiple high performing Graphical Processing Units (GPU), this model is not an efficient one. Also, as argued by (Ferentinos; 2018), the dataset used by (Mohanty et al.; 2016), are only from experimental setup and when real-world images were used for this model, the accuracy dropped significantly. In (Ferentinos; 2018), transfer learning is used where a model which was trained on some different dataset is utilized for a different dataset to improve the training times and performance of the model. Models used were AlexNet (99.06%), AlexNet@WTBn (99.49%), GoogleNet (97.27%), Overfeat(98.96%) and VGG (99.53%). Even though these two studies have exorbitantly high accuracy's, they do not generalize well on unseen data. This means that when an image different from their own dataset is used, the performance of the model drops. Due to this, a new method is required, which needs to be efficient with an ability to generalize well for unseen data. Hence, this led this study to employ a hybrid model which uses a pre-trained model to extract features from the images and use Extreme Gradient Boosting (XGBoost) as a classifier.

The main aim of this research is to build an efficient model which can be developed with minimum configuration, which generalizes well with high performance. Thus, transfer learning seems ideal for this case as it drastically reduces the training time (Ferentinos; 2018) to extract features and patterns from the image. This hybrid approach of transfer learning combined with XGBoost, to the best of our knowledge has not yet been utilized for the detection of diseases in plants. The second aim of the research is also the future work of (Ferentinos; 2018), which states that, after predicting the type of disease, a recommendation of a treatment plan for that particular disease should also be provided so that the disease can be treated effectively and on time. This aim is achieved by building a rest API combining treatment-related data and the model predictions on the server-side. The structure of the paper is as follows Section 2 encompasses all the successful literature's that conducted study in this domain, Section 3 showcases the methodology used, Section 5 discusses how the study was implemented, Section 6 analyzes as discusses the results and Section 7 provides conclusion and future scope of the study.

2 Related Work

Identification or recognition of diseases in plants manually is expensive, laborious, time-consuming and requires skilled professionals. As a result, much research has taken place in finding automated methods which can be reliable, accurate and cost-effective. Due to the recent advancements in machine learning various methods have been put forth to tackle this problem providing exceptional results as mentioned below:

2.1 Plant disease detection using traditional machine learning

The study (Chung et al.; 2016), used Support Vector Machine (SVM) to build a classifier to detect healthy or diseased seedling in rice. They use images of seedling taken at three weeks to distinguish the images. After optimizing the parameters, the model was able to classify infected seedling with 87.9% accuracy. Due to this, the authors claimed that this automated approach is far more superior than manual detection as it was quicker and accurate. Similarly, in (Shrivastava et al.; 2017), the authors presented another technique to reduce manual detection of diseases by employing SVM, K-Nearest Neighbors (KNN) and Probabilistic Neural Networks (PNN) to detect six types of diseases in soybean plants. The authors use several different color and texture features and emphasize the importance of these features in machine learning models. Acknowledging the importance of these features, the study (Camargo and Smith; 2009) mentions that texture related features becomes essential when the affected areas of the diseased plant do not follow any color pattern or shape. They also use SVM to classify the images similar to (Chung et al.; 2016) and produce good results.

In the study of (Liu et al.; 2016), the authors use SVM classifier to detect aphids in wheat-related images. They strongly mention that features such as color, texture, density, weather and location had an immense impact on the detection rate of aphids in wheat. Similarly, in (Akhtar et al.; 2013), images are segmented from the background, then a different type of handcrafted features are extracted from the images which are later used in different machine learning models such as KNN, Naïve Bayes (NB), Decision Trees (DT) and SVM. SVM performed the best among these classifiers. From these traditional machine learning approaches, it is seen that SVM performs the best for plant disease classification task, but one major drawback is that these models require hand-engineered features as they are unable to learn different features in images on their own. Due to this, deep learning models are used extensively for computer vision challenges (Pantazi et al.; 2019) as they are able to extract and learn features automatically and generally provide much higher accuracy than traditional machine learning classifiers for image classification.

2.2 Plant disease detection using Deep Learning

Owing to the massive success of deep learning architecture of disease detection tasks in plants, (Amara et al.; 2017) developed a CNN model based on LeNet architecture to classify banana leaf diseases. The total number of images in the dataset was 3700 resized to the size of 60x60. Both colors as well grayscale version was used to train the model where color model performed better as diseases have a certain type of colors or patterns which CNN extracts easily. The model performed very nicely, giving 92-99% accuracy based on different splits in data. However, due to the small size of the data, it can be assumed that the model may not generalize well as the model was trained from scratch with only 3700 images without any augmentation techniques. Similarly, (Francis and Deisy; 2019) uses around 3663 images to classify apple and tomato plant diseases. However, since in this study author used methods to control overfitting such as adding dropout layer and including image augmentation techniques in the pre-processing steps reached an overall accuracy of 87%.

As mentioned by (Arsenovic et al.; 2019), due to limitations in acquiring a real-world dataset of plant diseases, more and more researches have used transfer learning to eliminate the need of having a vast dataset to achieve good results. However, this study uses a Generative Adversarial Network (GAN) to generate new synthetic images

to overcome this challenge. This approach yielded good results and performed better, even in the hold-out test set. However, GAN requires a lot of GPU processing, and if researchers have limitation in the resources that are available to them, this approach cannot be used.

2.3 Plant disease detection using Transfer Learning

The most successful and also considered to be state of the art in plant disease detection is the work of (Mohanty et al.; 2016). They use pre-trained Convolutional Neural Network (CNN) architectures and train only the last few layers of these models on their plant disease dataset to build a model. The dataset has been converted into three types i.e., color, grayscale and segmented, and the model is trained on each one of them. It was found that the model trained on the colored version performed the best. This finding helps the current research as the main objective of the research is to find the optimal solution without requiring any intense computations. Although the model proposed by this research gives excellent overall accuracy of 99.53%, but as they themselves mentioned, and also as pointed out by (Barbedo; 2018; Arsenovic et al.; 2019), the model when used for a dataset different than the one it was trained on, the results dropped almost 31%, which can be due to the fact that no image augmentation techniques or usage of dropout layers to control overfitting was implemented. Extending the work done by (Mohanty et al.; 2016), (Ferentinos; 2018) included real-world images of diseases in plants to the same dataset in an attempt to improve the generalisability of the model. The authors argued that images taken using a controlled environment were the reason that (Mohanty et al.; 2016) achieved such high accuracy and that when images from the real world are used, the model does not perform well. In addition to that, multiple architectures such as AlexNet, AlexNet@WTBn, GoogleNet, OverFeat, and VGG was used to find the best architecture for this problem. No image augmentation techniques have been used. Although VGG performed the best with 99.53% accuracy, the time taken per epoch was 7034 (sec/epoch) with 67 epochs. This shows that an enormous amount of time has been spent training the model on high performing GPU's, which brings us to the main aim of this research, which is to build an optimal model which reduces this training time, but still capable of providing acceptable results. The study also mentions a future possibility of recommending treatments for the predicted diseases.

In the study conducted by (Rangarajan et al.; 2018), transfer learning is used to identify six types of diseases in tomato plants namely, Late blight, Leaf mold, Two-spotted spider mite attack, Target spot, Tomato mosaic virus, and Tomato yellow leaf curl disease. The architectures used were VGG16 (97.29%) and AlexNet (97.49%), of which AlexNet performed slightly better than VGG16. No methods to prevent overfitting were implemented apart from dropout layers which are already present in the pre-trained models. The study also intended to find the effects of different hyper-parameters on the accuracy and found that learning rate has a much greater impact on the accuracy than the epochs (n=10) and batch size (n=32). Thus, in the current research, epochs have been set to 10 for all the models as mentioned by (Rangarajan et al.; 2018).

In (Fuentes et al.; 2017), the study used deep meta architectures as well as deep feature extractors to identify diseases and pests in plants in real-time. The deep meta-architectures such as Faster-R-CNN, Region-based fully convolutional network (R-FCN) and Single Shot Multibox Detector (SSD) are used in combination with pre-trained deep learning architectures such as VGG16, ResNet50, ResNet101, ResNet152 and ResNeXt50.

For pre-processing the images are annotated with bounding box manually, then data augmentation techniques are used to prevent overfitting similar to the current research. Here as well, similar to (Ferentinos; 2018), VGG16 performed better than the rest of the model. Hence, VGG16 is utilized as one of the pre-trained models in the current study as well, as it performed best in many of the literatures.

In the comparative study conducted by (Too et al.; 2019), pre-trained architectures VGG16, ResNet50, ResNet101, ResNet152, InceptionV4 and DenseNet121 are compared on the plant village dataset. Out of which DenseNet performed the best with 99.75% accuracy. Batch Normalisation is used in this study to normalize the output of each layer to be ingested as an input by the next layer which reduces overfitting of data and allows higher learning rate which as mentioned by the author, which reduces the overall training time significantly. Hence, in the currently proposed study, batch normalization is used after each layer for the model to achieve the aim of building an efficient model with low computational requirement.

In (Ramcharan et al.; 2017), used an approach using pre-trained CNN models re-training the last few layers on their own dataset and using Support Vector Machine (SVM) as a classification layer. InceptionV3 architecture with SVM gave an accuracy in the range of 90-95%. They also mentioned that, when using transfer learning, even if the size of the dataset is small, there is no significant impact on the performance of the model. Thus, in the current study, image augmentation techniques are used only to transform or edit the existing images as it is not necessary to have a large dataset when using transfer learning. This also helps in reducing the overall processing requirements allowing us to achieve the main aim of the research.

2.4 Image classification using XGBoost

(Zhong et al.; 2019) uses deep learning models to monitor the growth of the plants and identify different crop types. According to the authors, CNN is used as the feature extractors to eliminate the process of generating manually crafted features which reduces the processing time and standardizes feature extraction. The CNN model built by (Zhong et al.; 2019) is developed using convolution 1D, as they found that convolution 1D layers are computationally efficient when it comes to identifying temporal patterns in vegetation images. Since in the currently proposed research, there will not be any temporal patterns in the images of leaves, convolution 2D is used, which reduces the dimension to 2 from 3 to make it computationally efficient. It is found that the best accuracy was given by the CNN model with 85.54%

In the comparative study conducted by (Georganos et al.; 2018), they built a classifier using extreme gradient boosting (XGBoost), to understand the growth of population in urban areas. Other machine learning classifier such as RF and SVM were also used to understand the performance of XGBoost over other classifiers. It was found that XGBoost clearly outperformed RF and SVM. The authors also mention in the future work to combine deep learning method to understand if there is any improvement in the performance of the model, which is exactly what the current research is attempting to perform.

The study (Mudgal et al.; 2017) uses XGBoost as the classifier to detect brain tumors from magnetic resonance images (MRI). They use K-Means clustering with mean shift segmentation for pre-processing, which basically reduces the noise in an image for better feature detection. Morphological enhancements like Marker-Controlled Watershed

Transform are also performed to highlight tumor affected region specifically and lastly, for feature extraction, Gray-Level Co-Occurrence Matrix (GLCM) is used. Their model, however, produced 100% accuracy, which may be due to the size of the dataset, which was around 50. But, nevertheless, as mentioned by them that the implementation of XGBoost is growing in different domains due to the performance provided by it is comparatively quicker than the rest.

In (Jiang et al.; 2018), the study classifies images taken from the satellite into a specific category of terrain. The aim of the project is to build an automatic scene label system to build a comprehensive labeled dataset for future usage. This approach used for the classification task provided great results, reaching a max accuracy of 95%. They found that the performance of the CNN model built from scratch with XGBoost performed exceptionally well in terms of accuracy and training time. Thus, in the currently proposed study pre-trained CNN model along with XGBoostand SVM is going to be used and the performance of the CNN model built from scratch will also be compared for the classification of diseases in plants and verify the results that this study achieved.

In (Gao et al.; 2017), the authors present a novel classification technique based on XGBoost. They use a pre-trained CNN model and fine-tune the model on their dataset which is then passed to the XGBoost classifier based on weighted column subsampling. This model achieves an accuracy of 92.1% which is considered as state of the art for PASCAL VOC 2012 dataset. The authors state that when features are extracted using CNN model there is a possibility of features being redundant and hence, XGBoost is the ideal candidate for this type of scenario as it builds multiple trees sequentially based on the strengths of the previous trees.

2.5 Summary

Study	**Models**	**Highlights**
(Chung et al.; 2016)	SVM	Successfully automated disease detection and provided its benefits over manual.
(Camargo and Smith; 2009; Liu et al.; 2016; Shrivastava et al.; 2017)	SVM, KNN, PNN	Hand crafted several features and emphasized its importance for traditional machine learning classifiers. SVM was the best perform classifier.
(Amara et al.; 2017)	LeNet architecture	Colored version of images performed better than grayscale
(Francis and Deisy; 2019)	4-Layer CNN	Achieved 87% without engineering any features. Used several techniques to control overftting
(Arsenovic et al.; 2019)	AlexNet + YOLO classifier	Used novel General Adversarial Networks (GAN) for effective image augmentation creating artificial images for the model to generalise well on unseen data
(Mohanty et al.; 2016)	AlexNet, GoogleNet	The first state of the art study using transfer learning achieving 99.35% accuracy on plant village dataset

(Ferentinos; 2018)	AlexNet, AlexNet@WTBn, GoogleNet, Overfeat, VGGNet	VGG model gave the best performance 99.53%, extending the study of state of the art model.
(Rangarajan et al.; 2018)	VGG16, AlexNet	VGG16 performed the best with overall score of 97.29%.
(Rangarajan et al.; 2018)	VGG16, ResNet50, ResNet101, ResNet152, ResNeXT50	Here as well VGG16 outperformed all the other models.
(Too et al.; 2019)	VGG16, ResNet50, ResNet101, ResNet152, InceptionV4, DenseNet121	Batch Normalisation is used to reduce overfitting and increase training efficiency of CNN model
(Ramcharan et al.; 2017)	InceptionV3, SVM	Fine tuned InceptionV3 model along with SVM giving excellent scores with low computational time
(Georganos et al.; 2018)	RF, SVM, XGBoost	XGBoost outperformed all of the traditional classifiers.
(Mudgal et al.; 2017)	XGBoost	Emphasized the usage of CNN models for feature extraction, as it automatically learns the important features without manual processing
(Jiang et al.; 2018)	CNN and XGBoost	Fine tuned or retrained CNN model along with XGBoost
(Gao et al.; 2017)	CNN and XGBoost	Similar to (Jiang et al.; 2018)

As seen from the related works, many issues and challenges were identified. The lack of real-world images of plants with labels does not allow the state of the art models to generalize well. The amount of time and resources required to train and build the model is enormous, and only those researches and studies which has access to these resources can try and implement this project. Transfer learning has two approaches, where one involves adding additional layers or retraining the layers of the pre-trained model; the other one is to use the model without training only for feature extraction. This may help in reducing overfitting as the model will never learn features of the dataset used by this study and thereby generalize well. Thus, in this research, we try to undertake the latter approach, which is known to reduce the computational resources (Simonyan and Zisserman; 2014), using architectures SqueezeNet, VGG16, and VGG19. Thus, the current study aims to strike a balance between optimal performance with a low computational requirement, which also needs to generalize well on unseen data.

Also, extending the future work of (Ferentinos; 2018), this study also aims to build a rest API to provide treatment details for the predicted disease.

3 Methodology

The research methodology followed in this study was given by (Fayyad et al.; 1996) known as Knowledge Discovery Database. It follows certain processes which allows a research to reach its goal in a systematic way. It consists of the following processes,

- Data Selection

- Data Pre-Processing
- Data Transformation
- Data Mining
- Interpretation/Evaluation

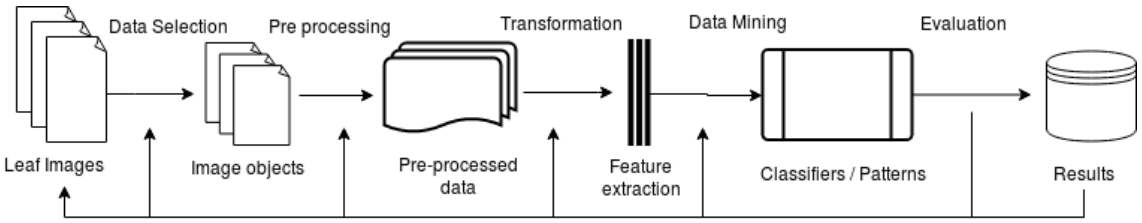

Figure 1: Research methodology followed with reference to KDD

These processes are explained in detail below in reference with the current research.

3.1 Data Selection

As mentioned by (Arsenovic et al.; 2019), acquiring datasets which contain images of plants in the real world and not from controlled laboratory environment is difficult and expensive, this study reused the dataset used by the state of the art literature (Mohanty et al.; 2016) which contains images of 14 different plant species affected by 26 diseases. In total, the dataset had 38 class labels where each class label is a plant-disease combination. The dataset was already transformed into three parts by (Mohanty et al.; 2016) into color, segmented and grayscale. In this study, only the color version is used as the study (Ferentinos; 2018; Mohanty et al.; 2016) found excellent results using the color version of the images. This dataset[1] was made available publicly on GitHub by the authors (Mohanty et al.; 2016). This dataset was, however, first created by the study (Hughes et al.; 2015).

For completing the second aim of the project which is to recommend treatment for the predicted disease, a second dataset was created by using the data web scraped from the Planet Natural[2] website. The authenticity and the validity of the data were checked by finding literature's (Goswami et al.; 2018; Chouhan et al.; 2018; Evans and Percy; 2014), which have used data from the website for research purposes. Treatment-related to each disease present in the first dataset was extracted using python library BeatifulSoup4[3] from the website, where the treatment products and treatment steps were extracted and stored in a CSV file. It must be noted that not all diseases were found in this particular site, and efforts were made to find trustworthy information for the remaining diseases, but no satisfactory data was found.

[1]https://github.com/spMohanty/PlantVillage-Dataset
[2]https://www.planetnatural.com/pest-problem-solver/plant-disease
[3]https://pypi.org/project/beautifulsoup4/

3.2 Data Pre-Processing

The repository of images was created in such a way that each folder represented a label/class of images with the syntax *<plant_name>_<disease_name>*. Hence, the first task was to remove all the folders of images which only had images of either healthy or only diseased plants. Hence, the following classes *('Blueberry_healthy, Orange_Haunglongbing_(Citrus_green ing), Raspberry_healthy, Squash_PowderyMildew, Soyabean_healthy')* were removed from the repository as the model requires at least two folders of the same plant. The second step involved the removal of those folders of images for which there was no matching data available from Planet Natural website. Thus another six folders were removed from the repository bringing our final classification classes to 27.

Then the source repository was split into two separate folders of train and test with stratified 80/20 ratio. The 80/20 ratio was selected as it gave the best results in the study (Ferentinos; 2018; Mohanty et al.; 2016). Thus around 26379 images were added to the train folder, and 6608 images were moved to val(test) folder, which is more than enough since our approach is to use transfer learning. Only after the root folder is pre-processed, cleaned and split, the data is transferred to the cloud instance so as to reduce the overhead of transferring data to the cloud. The data were loaded into the models using ImageDataGenerator class from Keras library, which automatically reads all the images from the root directory to the child directories and labels them accordingly. It also transforms the images, as explained in the following section. One hot encoding is performed as the Keras fit method requires labels to be in that form.

3.3 Data Transformation

A number of data transformations are utilized to ensure that the model does not overfit on the training data and that it is able to provide good validation accuracy on the test set. The steps followed were utilized after careful research is done in section 2, where image augmentation, batch normalization, and dropout layers were used to mitigate this issue. The Image augmentation techniques were applied as mentioned in (Chollet; 2016),

- rescale: It normalizes the Red Green Blue (RGB) values, which is always in the range between 0-255, to values between 0-1.
- zoom_range: It randomly zooms inside the image. The images of the leaf benefits from this method as the areas affected by the disease becomes much bigger and makes it easier for the model to learn features related to only affected parts.
- width_shift_range: Shifts the image on the horizontal axis.
- height_shift_range: Shifts the image on the vertical axis.
- rotation_range: Rotates the entire image based on the degree of angle specified.
- horizontal_flip: Flips the image horizontally.
- fill_mode: It is the method of filling in newly created pixels after applying the above transformations

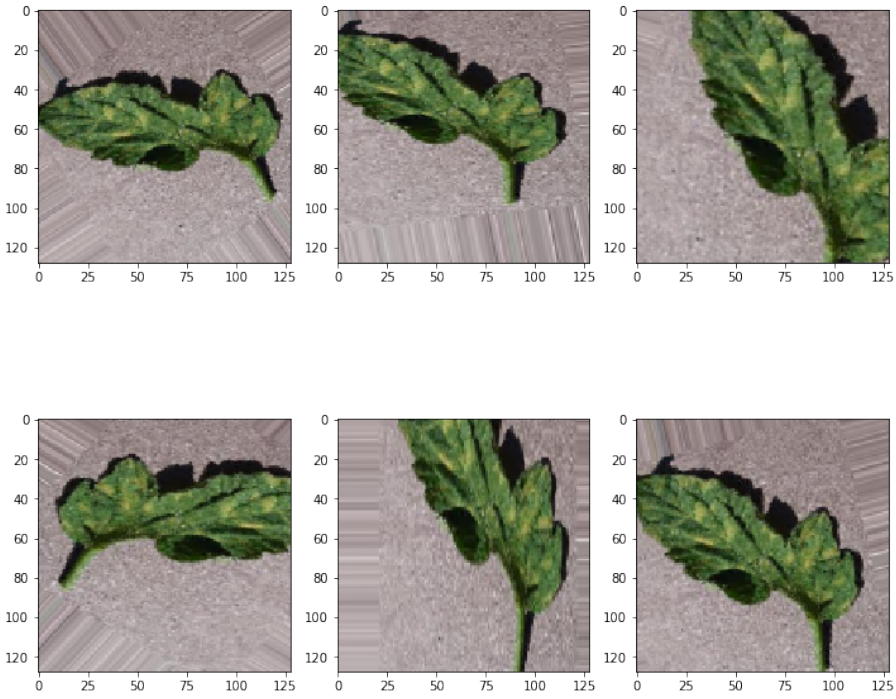

Figure 2: Example of transformation performed on training data of size 150x150 pixels

However, it must be noted that these transformations were only applied to the training set and not to the testing set so that the model is able to generalize well on unseen data which is a major cause for concern in (Mohanty et al.; 2016; Ferentinos; 2018) as mentioned by (Arsenovic et al.; 2019). The only transformation applied to the testing set was the normalization.

3.4 Data Mining

Several models were built to find the optimal balance between good validation accuracy and model performance. For this, two approaches were selected, which involved building a CNN model from scratch and building a hybrid model which uses a pre-trained CNN model as feature extractor and uses SVM and XGBoost as the classifiers. For the first approach, five different CNN models were built with layer 2, 3, 4, 5, and 6 to find the model which performs the best for plant disease detection. The model with the highest validation accuracy was selected, and that model was used to conduct hyper-parameter optimization. For the second approach, three pre-trained models were used to extract features from the images. They were SqueezeNet, VGG16, and VGG19 as these models performed the best with low computational time based on the literature. The classifiers used for these models are SVM and XGBoost. Based on the literature's SVM performed the best among other traditional machine learning classifiers and XGBoost, which is yet to be implemented in this domain is utilized to understand if there is any benefit over other methods. 10-fold cross-validation is also performed to find the optimal set of parameters found using Randomized Search. The prediction is merged with the second dataset to provide insights into treatment plans based on the classification

3.5 Interpretation/Evaluation

The primary metrics considered for measuring the performance in this study are accuracy, precision, recall, and f1 score. Evaluations are conducted on five different CNN model with different layers built from scratch and three different pre-trained CNN model executed each one with SVM and XGBoost. The metrics are calculated as explained in (Cruz et al.; 2019; Loresco et al.; 2018)

- **Accuracy:** It is calculated as the number of correct predictions made over the total number of predictions.
- **Precision:** It is calculated as the number of correct predictions over a number of correct predictions and a number of predictions which the model assumed to be correct but was not. In short, it is the number of true positives divided by a number of true positives and a number of false positives. Since this is a multiclass classification of 27 classes, the precision score is calculated for each of the classes and a mean value extracted as the overall precision score of the model.
- **Recall:** It is calculated as the number of correct predictions over a number of correct predictions and predictions which the model assumed to be something else but was not. In short, it is the number of true positives divided by a number of true positives and a number of false negatives. Since this is a multiclass classification of 27 classes, recall score is calculated for each of the classes, and a mean value is extracted as the overall recall score of the model.
- **F1 score:** It is also known as the mean of precision and recall. All the models are majorly evaluated using the f1 score as the most important metric as the state of the art literature's (Mohanty et al.; 2016; Ferentinos; 2018) used this metric for evaluation.

4 Design Specification

Since this study has used two approaches, this section explains in detail how each model is built and the architecture of each model. A normal CNN model consists of different types of layers which allows the model to learn and extract features relevant to those classes. The layers that are used in the model are explained below as mentioned by (Yang et al.; 2017),

- **Convolution layer:** This is the layer where n a number of filters are applied to extract features based on the given size of the kernel.
- **BatchNormalisation:** Batch normalization is used to normalize the output of one convolution going as an input to another convolution. This results in efficient training and helps in reducing overfitting.
- **Max Pooling:** Max pooling layer is used to reduce the dimensionality of the feature map by selecting the maximum value of a particular region based on the kernel size.
- **GlobalAveragePooling2D:** Reduces the dimension to 1 for the Dense layers by taking the average value of the feature maps.

- **Dense (Fully Connected):** These are the layers that are usually placed before the classification layer. It aggregates the information which is extracted from the previous layers feature maps.
- **Dropout layer:** This layer is used to reduce overfitting (Krizhevsky et al.; 2012) by dropping the specified percentage of features from the model.
- **Softmax layer:** This is the final dense layer, also known as the classification layer.

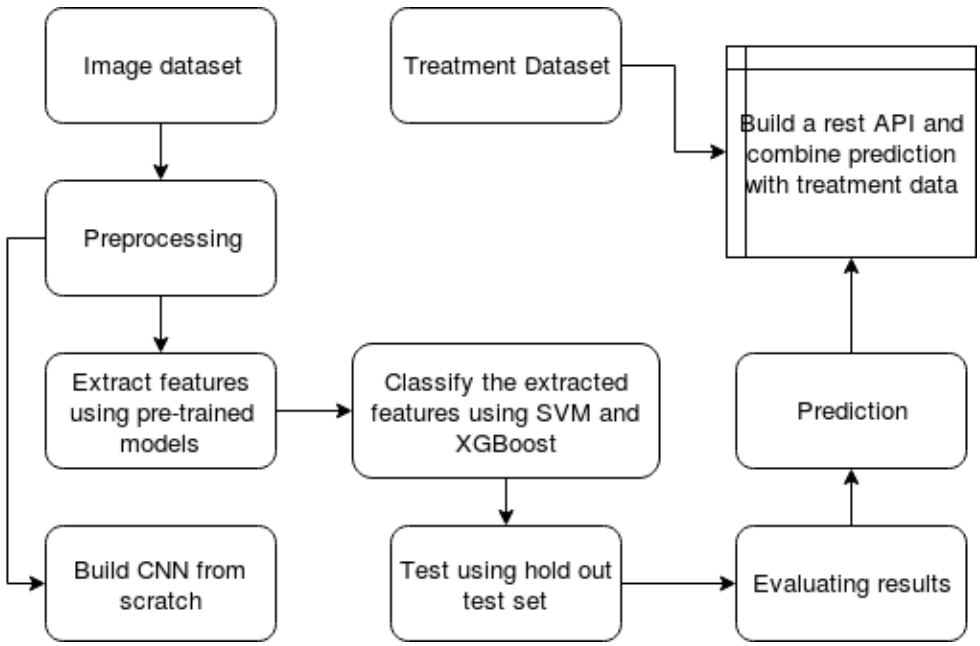

Figure 3: Research study architecture

4.1 Convolutional Neural Network model built from scratch:

The model built from scratch follows a combination of architecture VGG16, VGG19 (Simonyan and Zisserman; 2014) and SqueezeNet (Iandola et al.; 2016). This is because VGG architectures are one of the best performing models for plant disease detection based on the literature's and whereas SqueezeNet gives AlexNet level accuracy with 50 times faster processing and low memory consumption according to (Iandola et al.; 2016; Brahimi et al.; 2018). Thus, the architecture consists of 4 convolutional layers with different filter size 3x3, 4 max poling layers each of size 2x2, 4 batch normalization layers after each convolutional layer, a global pooling average layer to reduce the dimension to 1, 2 fully connected layer with 2 drop out layers to reduce overfitting and the last classification layer with 27 neurons representing the 27 classes of plant diseases.

4.2 Transfer learning using pre-trained model:

Transfer learning is used to transfer the knowledge gained by learning from data of other domain, into a different domain. As Andrew NG himself said, *"Transfer learning is the next driver for ML success"*, this method is employed in the architecture of this study to efficiently use the knowledge extracted by models on ImageNet database (1.6 million

images) containing images from different domains and has around 1000 classes. Thus, we use the best performing models that were used on this dataset as it is known to provide excellent performance minus the computational time.

4.2.1 SqueezeNet:

The SqueezeNet architecture developed by (Iandola et al.; 2016) literally squeezes the input feature maps to reduce the total number of parameters that are computed while training the model. This makes this model fast, and as mentioned by the authors gives performance similar to (Krizhevsky et al.; 2012). The architecture consists of 1 convolutional layer, 3 max-pooling layers, and 8 fire layers. It contains no dense layers which help in faster processing, 1 global average pool layer and lastly the classification layer with 1000 classes. Due to this squeezed architecture, the resulting model takes 50 times fewer parameters and is 510 time smaller than the next best small architecture i.e., AlexNet.

4.2.2 VGG16 and VGG19:

This architecture, developed by (Simonyan and Zisserman; 2014), contains a considerable increase in the number of convolutional layers when compared with AlexNet and SqueezeNet. However, as mentioned in the related works section 2 VGGNet architectures perform really well for plant disease detection using images. The architecture is made up of 13 convolutional layers, followed by 5 max-pooling layers. The number of filters for each convolutional layers varies from 64-512 based on how deep that particular layer is. The final layers of the model are precisely similar to AlexNet with 2 dense layers of 4096 neurons and last dense layer of 1000 neurons.

5 Implementation

The section provides detail on how the entire implementation of the project was done for creating an efficient plant disease detection model. It also discusses how the treatment plan was suggested based on the classification.

5.1 Environment

. The project was implemented using Python 3.5 using Jupyter notebook as the main integrated development environment (IDE). Python language was selected as there is a lot of support from an active community for image classification using Tensorflow with Keras. The study was initially started on a local Linux system but was eventually moved to Google Cloud Platform (GCP)[4] due to a shortage of resources. A Deep Learning VM image[5] instance was created on the cloud, which is optimized for deep learning and has all the required packages pre-installed.

[4]https://cloud.google.com/

[5]https://cloud.google.com/deep-learning-vm/docs/cloud-marketplace

5.2 Data Handling

The data for the treatment plan was web scraped from the Planet Natural website[6] as explained in section 3.1 and saved as Fungicide_Recommendation.csv file. The second dataset using which the CNN model is built was downloaded and pre-processed, as mentioned in section 3.2. The main issue in plant disease detection using deep learning according to (Arsenovic et al.; 2019) is that, although the state of the art study's (Mohanty et al.; 2016; Ferentinos; 2018) give excellent scores, the model performs poor on a different dataset, which is due to poor handling of model overfitting. Thus to handle this, image augmentation is performed using ImageDataGenerator class from Keras library on only the training set; hence, the model learns features using only morphed images and does not overfit.

5.3 Architecture

All the required packages were pre-installed on the cloud instance to implement TensorFlow using Keras. Any extra packages that were needed such as pickle, joblib to save the objects, xgboost, flask, etc. were installed using pip command. For the first of the two approaches as explained in section 4 5 different CNN models were created with a different number of layers and default configuration to find how many layered CNN model is the most efficient for plant disease detection. Out of all the model, starting from 2 layers till 6, 4 layer CNN model performed the best in this study with good training time. The selected model was then used to perform hyper-parameter tuning using Random Search. Parameters space used were

- batch_size: 32, 64, 96 (had to keep lower than 100 due to memory error)
- epochs: 10, 15
- dropout_rate: 0.3, 0.5, 0.7
- optimizer: SGD, Adam, Adamax, Nadam

The model was then subjected to 10-fold cross validation to ensure that the model is not overfitting and the variance is low.

The second approach of transfer learning is to use pre-trained CNN model for a different task to reduce the need for huge amounts of data and the processing time. Three models were selected, namely, SqueezeNet, VGG16, and VGG19. SqueezeNet was selected as it provides great performance with low computational time (Iandola et al.; 2016) and VGG models were selected as these models performed very well in the majority of the studies. According to (Simonyan and Zisserman; 2014) and also as mentioned in section 2, It is not necessary to fine-tune this model to achieve good performance, we can directly use this model to extract features and use traditional classifiers to boost the performance and reduce computational costs. Thus, we use these models only to extract features without re-training them with our dataset, which helps in preventing the main issue of overfitting as the model never learns features specific to this dataset. The extracted features are then fed to SVM and XGBoost to evaluate the performance. The classifiers are then optimized using random search where the parameters space used were,

- **SVM:** (Mantovani et al.; 2015)

[6]https://www.planetnatural.com/pest-problem-solver/plant-disease

- Cost (C): values between 2.0-10.0
- Gamma : values between 0.1-1.0
- kernel : RBF and Linear

- **XGBoost:** (Nishio et al.; 2018)
  - max_depth: values between 3-10
  - n _estimators: values between 1-100
  - learning_rate : 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.50, 0.60
  - eta: values between 1-100
  - subsample : values between 0.5-1.0
  - min_child_weight : values between 1-6
  - gamma: 0.0, 0.1, 0.2, 0.3, 0.4

Before hyper-parameter tuning, dimensionality reduction has to be performed as the number of features that VGG16 and VGG19 extracts are 4096 and SqueezeNet extracts 1000. So Principal Component Analysis (PCA) is used before tuning the parameters as it will lead to memory error, or it would take days to try all different combinations of parameters. Features that account for 95% variance are selected for VGG16 and VGG19, which reduces the features from 4096 to 828 and 761, respectively. For SqueezeNet, features that account for 99% variance are selected, which brings down the features to 269. After this random search of parameters is performed. The best parameters are selected to build the final model for evaluation. 10-fold cross-validation is performed for each of the model similar to the CNN model to ensure no overfitting has taken place and the variance of the model is low.

5.4 Building the rest API for treatment recommendation

The dataset web scraped and stored in Fungicide_Recommendation.csv file is used to merge the predicted disease with the appropriate treatment. For this, we use the python library flask, which allows the building of a simple rest API with few lines of code. The method to build the rest API was followed from Keras official blog[7]. A separate python notebook is created for this task. The best performing model from the earlier task is saved using joblib library. A POST request is sent to the server along with the image. On the server-side, the image is pre-processed, and features are extracted using pre-trained model and classified with the best performing saved model. Then the disease is searched in the CSV file, and the matching row is sent back a JSON object.

6 Evaluation

This section provides a comprehensive analysis of the achieved results related to the aim of the research. All of the models and architectures that have been implemented for this study has been selected from the best performing models or parameters, to try and solve some of the issues that this domain faces. Two of the most common machine learning classifiers that are selected for this study are SVM and XGBoost.

[7]https://blog.keras.io/building-a-simple-keras-deep-learning-rest-api.html

Support Vector Machine (SVM) was employed since image classification is a difficult task and any traditional classifier will find it challenging to achieve good results, which have been proved in literature review where CNN, SVM, and XGBoost outperformed all other classifiers. SVM classifies points by finding the hyperplane with the biggest margins between two points. If it is not possible to perfectly separate the two points, SVM finds the margin with a minimum intersection between two points (Chung et al.; 2016; Liu et al.; 2016). This is highly beneficial for image classification.

Extreme Gradient Boosting (XGBoost) was employed for this task as it is known to outperform most of the state of the art in various domains, and since it is yet to be utilized successfully for plant disease detection, this study utilizes this model. XGBoost is an extension of gradient boosted decision trees. It groups together weak classifiers to make one strong classifier. Unlike Random forests where each tree is built parallelly, XGBoost creates trees step by step by optimizing previously built trees (Mudgal et al.; 2017)

Models	**F1-score**	**Kappa**	**Recall**	**Precision**	**Accuracy**
VGG16-SVM	**96.16**%	**96.35**%	**95.7**%	**96.7**%	**96.5**%
VGG16 - XGBoost	89.7%	90.2%	88.7%	91.05%	90.8%
VGG19 - SVM	95.07%	95.5%	94.6%	95.6%	95.8%
VGG19 - XGBoost	89.5%	89.8%	88.6%	90.6%	90.4%
SqueezeNet - SVM	94.9%	95.13%	94.9%	94.8%	95.4%
SqueezeNet - XGBoost	82.9%	84.2%	80.7%	86.4%	85.27%
4-Layer CNN	96.86%	NA	97.05%	98.03%	92.23%
(Mohanty et al.; 2016)	**99.34**%	NA	NA	NA	**99.35**%
(Ferentinos; 2018)	**99.53**%	NA	NA	NA	NA

6.1 SqueezeNet with SVM and XGBoost

SqueezeNet architecture is selected as it is computationally less expensive, and as demonstrated by the authors, (Iandola et al.; 2016) provides excellent results for image classification tasks. It also consists of less number of parameters that need to be trained, which saves a lot of time while training or extracting features.

From the figure, it is clear that SqueezeNet architecture is very efficient and takes the least amount of time out of all the model to extract features from the images. It takes around 226 seconds to extract features from the entire dataset of train and test. Also, the prediction performance given by SVM for SqueezeNet is exceptional around 94.9% f1 score, which means that when the model is predicting that a particular image belongs to a class, it is 94.9% sure. However, the performance expected from XGBoost given its algorithm and proven performance in other domain was below par. It achieved the f1 score of 89.7% with a training time of 3251 seconds which is too high than SVM with a training time of 71 seconds

6.2 VGG16 with SVM and XGBoost

VGG16 architecture was selected as it has a moderate amount of convolutional layers which does not affect computational requirement too much, and it was the best performing architecture in the majority of the literatures. From the figure, we can see that it takes around 4515 seconds, which is significantly lower than state of the art (Ferentinos; 2018) which took 7034 seconds per epochs for 67 epochs. However, the f1 score given by VGG16+SVM, as mentioned in the above table, is the highest score for this study. When compared with the state of the art, the f1 score is just a little behind, but it is significantly ahead in terms of time required by the model to complete execution of the model. This result falls in line with the first aim of the research, which was to build an efficient model with limited resources, but which was capable enough to provide high performance. Similar to the earlier performance with SqueezeNet, XGBoost performs lower than VGG16+SVM score with 89.7% f1 score but still good enough. Below figure 4 shows how well the model VGG16+SVM classifies each class. It is clear that the model does a pretty good job in the classification of all the classes.

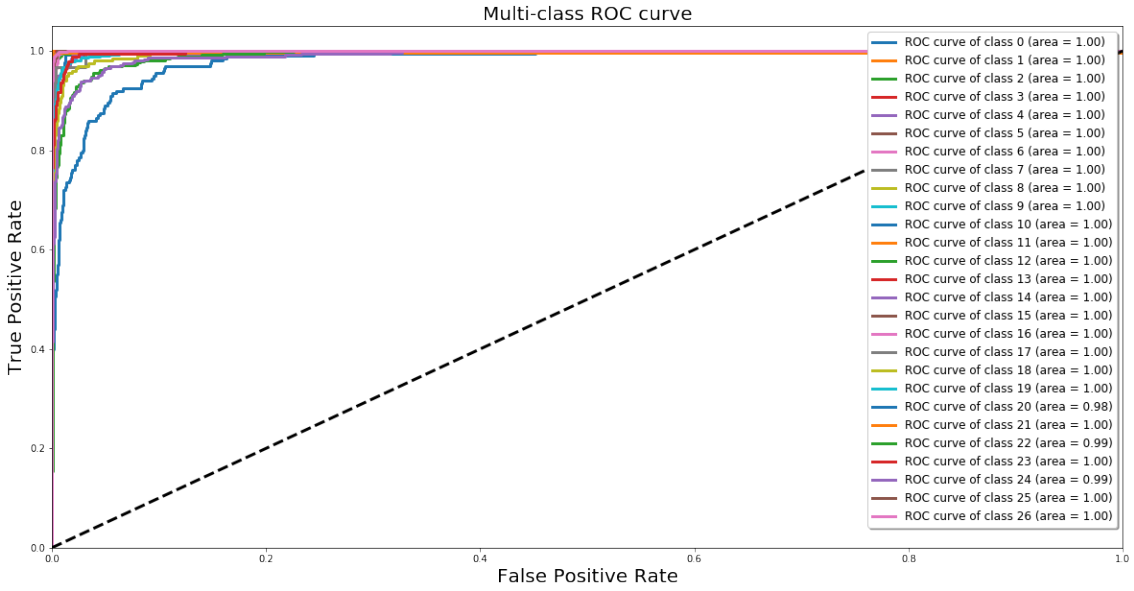

Figure 4: ROC curve for multi-class evaluation

6.3 VGG19 with SVM and XGBoost

VGG19 architecture was selected, as it is expected to give better performance than VGG16 in terms of prediction capability. But, in this study VGG19 performed slightly lower than VGG16 for both SVM and XGBoost. F1 score using SVM as classification layer was 95% whereas XGBoost provided an F1 score of 89%. However, the time taken by VGG19+SVM is still much faster when compared with the state of the art. These performances by these models shows that, this approach of transfer learning can be used when resources and time is of the essence to quickly and efficiently build a model.

6.4 CNN model with 4-Layer

This model was built and trained from scratch with 4 convolutional layers with filter sizes of 32, 32, 64, and 128. The model was executed for 10 epochs giving an f1 score of 96.8%;

however, the validation accuracy was around 92%. The model was subjected to 10-fold cross-validation to check the variance in the model and to test how the model performs for unseen data. Although there was no additional data from a different dataset to verify this, 10-cross-validation helps us analyze this variance in results. If the variance is high, it means that the model fails to generalize well. The model took around 8426 seconds to complete training for 10 epochs, which shows that training a CNN model from scratch requires a lot of processing time. As SVM is clearly the best performing model for this image classification study, we tried to implement one model with SVM using this 4-Layer architecture as the feature extractor. However, the result was meager, around 76% f1 score.

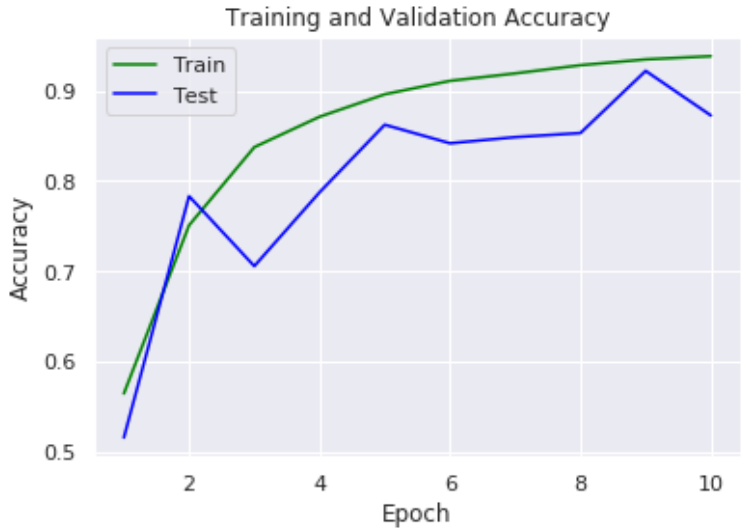

Figure 5: Training evaluation of CNN model for 10 epochs

6.5 Computational time taken by each model to build

As only the models built with deep learning using SVM as the classifier performed the fastest, an analysis was performed on the training time required for each model to prove the efficiency of the proposed approach against state of the art. Time units considered for this analysis is measured in seconds. As seen from the below table when it comes to time taken to build the model, Squeeze outperforms all the model which aligns with the claims made by the authors about the speed of their architecture (Iandola et al.; 2016). However, the state of the art model even though uses much more data, still consumes too much time to build a model that too on powerful GPU's. This shows that the proposed approach is a viable solution and can be explored further to reach its full potential.

Models	**Feature Extraction (sec)**	**Training (sec)**	**No. of Images**
VGG16 - SVM	4444	71	32,987
VGG19 - SVM	5741	68	32,987
SqueezeNet - SVM	**226**	**90**	**32,987**
4-Layer CNN	NA	8426	32,987
(Ferentinos; 2018)	NA	**471,278**	87,848

6.6 Discussion

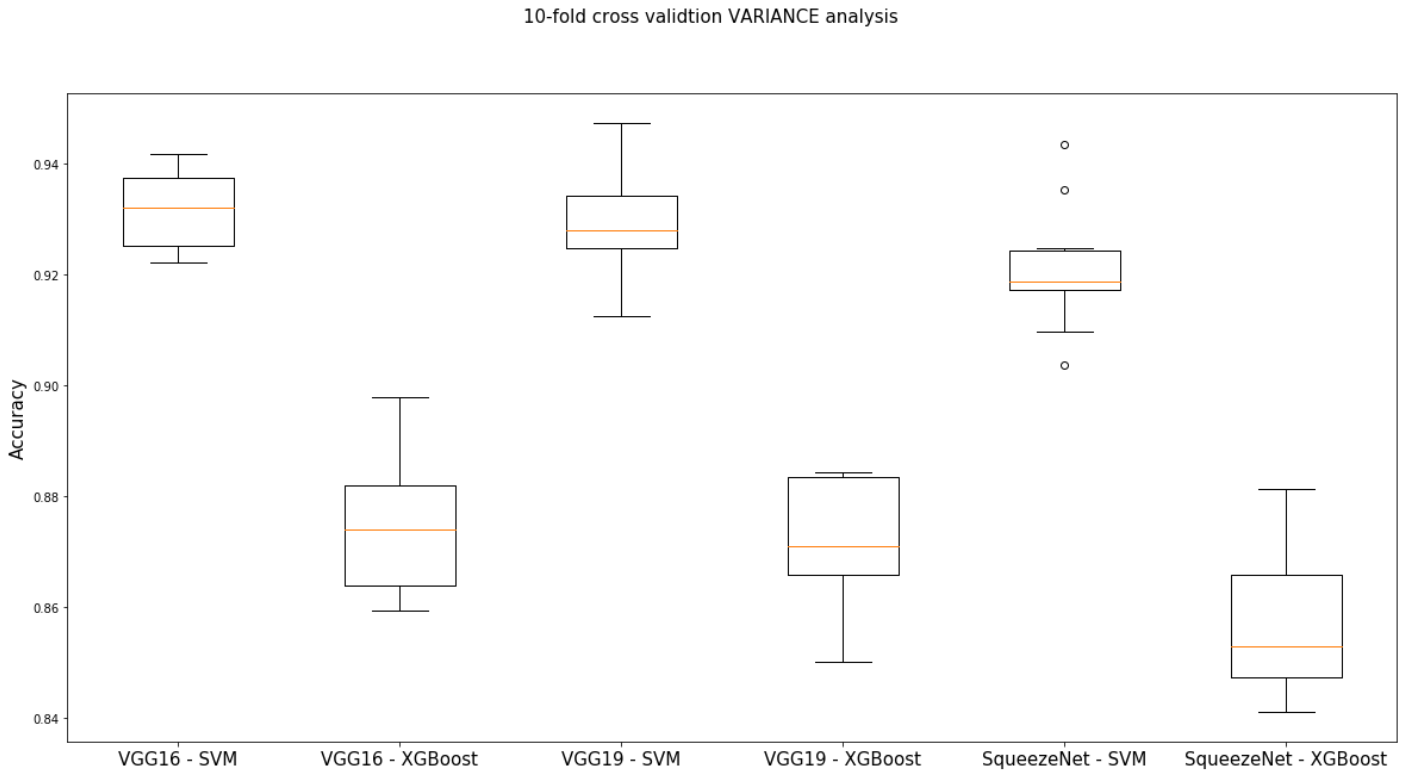

Figure 6: Evaluation of variance in model scores after 10-fold cross validation

The main aim of the research was to build a high performing model with adequate training time and speed using pre-trained deep learning architectures along with XGBoost. However, the models which utilized XGBoost performed comparably lower than the models which used SVM as its classification layer. Nonetheless, the aim to build a fast and efficient model was achieved with VGG16+SVM which gave the highest F1 score and fastest overall training time which includes feature extraction time from the deep learning architectures and the training time taken by the classifiers in seconds. When compared with the state of the art (Ferentinos; 2018) where it took 7034 seconds per epochs with 67 epochs, the model built in this study is hugely efficient with a slight decrease in accuracy (3%) which can be accepted as a successful research.

While trying to optimize the model for both SVM and XGBoost, PCA was used to reduce the dimensionality due to memory constraint. Features that explained 95% variance in case of VGGnet and 99% in case of SqueezeNet were retained. This method, which was used before hyper-parameter optimization might have been the reason why after optimization, the scores of all the model reduced even slightly. In the future, a machine with high memory can be utilized for performing a random search of parameters with all of the features without any reduction. If there were no memory constraints, the model was supposed to perform even better with greater accuracy.

All the model were subjected to 10-fold cross-validation to test the variance in results when presented with different unseen data set. The variance found was very low for VGG16+SVM model as shown in figure 6 which proves that the model will perform better than the state of the art studies for unseen data, where their model's accuracy

dropped almost 31% when tested with unseen data (Ferentinos; 2018). However, more data with real-world images are also needed to improve the robustness of the model as the model cannot be considered entirely robust just by performing 10-fold cross-validation.

7 Conclusion and Future Work

In this study, deep learning architectures are implemented by means of transfer learning using famous pre-trained architectures (VGG16 and VGG19) (Simonyan and Zisserman; 2014) and (SqueezeNet) (Iandola et al.; 2016). Architectures for this study are carefully selected based on their performance and computational requirement. These deep learning models were used only as a feature extractors as then it does not require retraining (saving us the computational time) and used with two famous classification algorithm which are successful in this domain i.e., SVM and XGBoost. The models were then trained on 32,987 images of plants with diseases using which the model VGG16+SVM attained an f1-score of 96.16% with the lowest computational cost. Thus proving the efficacy of the model, as this approach can be used by other researchers without needing substantial computational resources to conduct their research with high performance. Ultimately this model is then included in the rest API that is built-in combination with treatment data from the second dataset for showing that this model can be used by any developer for any kind of application as it gives response as a JSON object.

Even though the model was successful in achieving the aim of this research, it is not void of limitations. The testing of the model was conducted on unseen hold out test set, but it belonged to the same dataset, which is a simple step across most of the machine learning projects. This means that the methodology followed to create the dataset initially was same for all the images. Even though to minimize the effect of this, 10-fold cross-validation was performed to check if there is any variance for the proposed approach, there should have been additional data which was part of some other database. However, the cross-validation scores were checked to see if the variance was high, and it was found to be very low around 0.05%, which can be considered as extremely good. The second limitation of the model is that it is trained using only one leaf of the plant, which is either affected with the disease or not. This makes it highly important that when this model is used, only single leaf images are used to predict the disease as it might not work very well when images with multiple leaves are presented to the model. All in all, this model performed extremely fast in both training as well as classification and was tested extensively for variance in the results by performing 10-fold cross-validation, which was found to be very low. Thus, it means that when this model is tested with unseen data, it will perform better than the state of the art performance on unseen data which dropped by almost 31% (Mohanty et al.; 2016; Ferentinos; 2018).

In the future, due to limitations of data, General Adversarial Networks (GAN) as utilized by (Arsenovic et al.; 2019), which will be implemented to create artificial images and attempt will be made to include at least a small percentage of real-world leaf images. This will ensure that the model will perform better on unseen data as well. The developed rest API is also planned to be deployed after testing the model on unseen real-world dataset. Also, due to time and storage limitations, extensive hyper-parameter tuning was not performed as it led to memory error each time. In the future, much more number of parameters will be utilized without performing any dimensionality reduction and improve the model with entire dataset features.

8 Acknowledgment

I would like to express my deepest gratitude to my supervisor and guide Dr. Muhammad Iqbal for supporting and guiding me through this tough project that I undertook. If it wasn't for his constant helpful feedback the results that was achieved and the I was able to present it in a meaningful and sophisticated manner would not have been possible. This experience with him also helped me quickly grasp data analytics concepts and methods as I am a complete novice in this field.

I would also take this opportunity to thank my parents for believing and supporting me throughout this journey and allowing me to achieve my dreams. This would not have been possible without their constant support.

References

Akhtar, A., Khanum, A., Khan, S. A. and Shaukat, A. (2013). Automated plant disease analysis (apda): Performance comparison of machine learning techniques, *2013 11th International Conference on Frontiers of Information Technology*, IEEE, pp. 60–65.

Amara, J., Bouaziz, B., Algergawy, A. et al. (2017). A deep learning-based approach for banana leaf diseases classification., *BTW (Workshops)*, pp. 79–88.

Arsenovic, M., Karanovic, M., Sladojevic, S., Anderla, A. and Stefanovic, D. (2019). Solving current limitations of deep learning based approaches for plant disease detection, *Symmetry* **11**(7): 939.

Bandi, S. R., Varadharajan, A. and Chinnasamy, A. (2013). Performance evaluation of various statistical classifiers in detecting the diseased citrus leaves, *International Journal of Engineering Science and Technology* **5**(2): 298–307.

Barbedo, J. G. A. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification, *Computers and electronics in agriculture* **153**: 46–53.

Brahimi, M., Arsenovic, M., Laraba, S., Sladojevic, S., Boukhalfa, K. and Moussaoui, A. (2018). Deep learning for plant diseases: detection and saliency map visualisation, *Human and Machine Learning*, Springer, pp. 93–117.

Camargo, A. and Smith, J. (2009). Image pattern classification for the identification of disease causing agents in plants, *Computers and Electronics in Agriculture* **66**(2): 121–125.

Chaerle, L., Leinonen, I., Jones, H. G. and Van Der Straeten, D. (2006). Monitoring and screening plant populations with combined thermal and chlorophyll fluorescence imaging, *Journal of experimental botany* **58**(4): 773–784.

Chollet, F. (2016). Building powerful image classification models using very little data, *Keras Blog* .

Chouhan, S. S., Kaul, A., Singh, U. P. and Jain, S. (2018). Bacterial foraging optimization based radial basis function neural network (brbfnn) for identification and classification

of plant leaf diseases: An automatic approach towards plant pathology, *IEEE Access* **6**: 8852–8863.

Chung, C.-L., Huang, K.-J., Chen, S.-Y., Lai, M.-H., Chen, Y.-C. and Kuo, Y.-F. (2016). Detecting bakanae disease in rice seedlings by machine vision, *Computers and Electronics in Agriculture* **121**: 404–411.

Cruz, A., Ampatzidis, Y., Pierro, R., Materazzi, A., Panattoni, A., De Bellis, L. and Luvisi, A. (2019). Detection of grapevine yellows symptoms in vitis vinifera l. with artificial intelligence, *Computers and electronics in agriculture* **157**: 63–76.

Dhingra, G., Kumar, V. and Joshi, H. D. (2018). Study of digital image processing techniques for leaf disease detection and classification, *Multimedia Tools and Applications* **77**(15): 19951–20000.

Eun, A. J.-C., Huang, L., Chew, F.-T., Li, S. F.-Y. and Wong, S.-M. (2002). Detection of two orchid viruses using quartz crystal microbalance (qcm) immunosensors, *Journal of Virological Methods* **99**(1-2): 71–79.

Evans, K. J. and Percy, A. K. (2014). Integrating compost teas in the management of fruit and foliar diseases for sustainable crop yield and quality, *Composting for sustainable agriculture*, Springer, pp. 173–198.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data, *Communications of the ACM* **39**(11): 27–34.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis, *Computers and Electronics in Agriculture* **145**: 311–318.

Francis, M. and Deisy, C. (2019). Disease detection and classification in agricultural plants using convolutional neural networksâĂŤa visual understanding, *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, pp. 1063–1068.

Fuentes, A., Yoon, S., Kim, S. and Park, D. (2017). A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, *Sensors* **17**(9): 2022.

Gao, X., Fan, S., Li, X., Guo, Z., Zhang, H., Peng, Y. and Diao, X. (2017). An improved xgboost based on weighted column subsampling for object classification, *2017 4th International Conference on Systems and Informatics (ICSAI)*, IEEE, pp. 1557–1562.

Gavhale, K. R., Gawande, U. and Hajari, K. O. (2014). Unhealthy region of citrus leaf detection using image processing techniques, *International Conference for Convergence for Technology-2014*, IEEE, pp. 1–6.

Georganos, S., Grippa, T., Vanhuysse, S., Lennert, M., Shimoni, M. and Wolff, E. (2018). Very high resolution object-based land use–land cover urban classification using extreme gradient boosting, *IEEE Geoscience and Remote Sensing Letters* **15**(4): 607–611.

González-Fernández, R., Prats, E. and Jorrín-Novo, J. V. (2010). Proteomics of plant pathogenic fungi, *BioMed Research International* **2010**.

Goswami, M., Maheshwari, S. and Poonia, A. (2018). Performance analysis of classifiers and future directions for image analysis based leaf disease detection, *Recent Findings in Intelligent Computing Techniques*, Springer, pp. 519–525.

Hughes, D., Salathé, M. et al. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics, *arXiv:1511.08060* .

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size, *arXiv preprint arXiv:1602.07360* .

Jiang, S., Zhao, H., Wu, W. and Tan, Q. (2018). A novel framework for remote sensing image scene classification., *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* **42**(3).

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, pp. 1097–1105.

Kuckenberg, J., Tartachnyk, I. and Noga, G. (2009). Temporal and spatial changes of chlorophyll fluorescence as a basis for early and precise detection of leaf rust and powdery mildew infections in wheat leaves, *Precision Agriculture* **10**(1): 34–44.

Liu, T., Chen, W., Wu, W., Sun, C., Guo, W. and Zhu, X. (2016). Detection of aphids in wheat fields using a computer vision technique, *Biosystems Engineering* **141**: 82–93.

Loresco, P. J., Bandala, A., Culaba, A. and Dadios, E. (2018). Computer vision performance metrics evaluation of object detection based on haar-like, hog and lbp features for scale-invariant lettuce leaf area calculation, *International Journal of Engineering & Technology* **7**(4): 4866–4872.

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.

Mantovani, R. G., Rossi, A. L., Vanschoren, J., Bischl, B. and De Carvalho, A. C. (2015). Effectiveness of random search in svm hyper-parameter tuning, *2015 International Joint Conference on Neural Networks (IJCNN)*, Ieee, pp. 1–8.

Mohanty, S. P., Hughes, D. P. and Salathé, M. (2016). Using deep learning for image-based plant disease detection, *Frontiers in plant science* **7**: 1419.

Mudgal, T. K., Gupta, A., Jain, S. and Gusain, K. (2017). Automated system for brain tumour detection and classification using extreme gradient boosted decision trees, *2017 International Conference on Soft Computing and its Engineering Applications (icSoftComp)*, IEEE, pp. 1–6.

Nishio, M., Nishizawa, M., Sugiyama, O., Kojima, R., Yakami, M., Kuroda, T. and Togashi, K. (2018). Computer-aided diagnosis of lung nodule using gradient tree boosting and bayesian optimization, *PloS one* **13**(4): e0195875.

Pantazi, X. E., Moshou, D. and Tamouridou, A. A. (2019). Automated leaf disease detection in different crop species through image features analysis and one class classifiers, *Computers and electronics in agriculture* **156**: 96–104.

Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J. and Hughes, D. P. (2017). Deep learning for image-based cassava disease detection, *Frontiers in plant science* **8**: 1852.

Rangarajan, A. K., Purushothaman, R. and Ramesh, A. (2018). Tomato crop disease classification using pre-trained deep learning algorithm, *Procedia computer science* **133**: 1040–1047.

Rangaswami, G. and Mahadevan, A. (1998). *DISEASES OF CROP PLANTS IN INDIA*, PHI Learning.

Shrivastava, S., Singh, S. K. and Hooda, D. S. (2017). Soybean plant foliar disease detection using image retrieval approaches, *Multimedia Tools and Applications* **76**(24): 26647–26674.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition, *arXiv:1409.1556* .

Too, E. C., Yujian, L., Njuki, S. and Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification, *Computers and Electronics in Agriculture* **161**: 272–279.

Yang, H., Luo, L., Su, J., Lin, C. and Yu, B. (2017). Imbalance aware lithography hotspot detection: a deep learning approach, *Journal of Micro/Nanolithography, MEMS, and MOEMS* **16**(3): 033504.

Zhong, L., Hu, L. and Zhou, H. (2019). Deep learning based multi-temporal crop classification, *Remote sensing of environment* **221**: 430–443.