Detection (Bounding boxes) and Classification of Plant Leaf Diseases

Sanjana Aravindan, Harshini Keerthi Vasan, Garima Chhabra

Abstract

Detection and Classification of Plant Leaf Diseases using image processing could be very useful in detecting and preventing the crop diseases which lead to loss of produced yields. Automated Systems like these could be very useful for a country's agricultural production which in-turn will lead to a sustainable development. To achieve the mentioned objective, we have implemented 4 neural models- Resnet-152, AlexNet, DenseNet and a Model Trained From Scratch. We have also experimented with a model to detect the diseases with bounding boxes, which performs localization of the infected regions in the leaves.

Introduction

Although the latest advancement in technology has lead to the betterment of life for human beings, plant diseases cause a serious threat at a global level and rises serious consequences to the farmers whose nourishment depends on these plant crops. A large proportion of farmers live in hunger due to the pathogen derived disruptions in the supply of food. Plant diseases cause a serious problem to the production. To guarantee the quality and quantity of crops, protection of plant leaves from diseases is a vital scenario to consider. The treatment of the plant leaf diseases needs to be done at an early stage for the protection of other plant crops and prevention of disease being spread around other plant types. For instance, a plant disease called as Late blight is very destructive and is prone to affect the plant leaves, fruits and stems. It becomes very difficult for the computer to interpret the succinct information that the image conveys, hence feature extraction is required as an essential step. It is difficult to diagnose the plant leaf disease through the observation of the symptoms on the plant leaves even by experts in the agriculture domain due to the complexities.
Therefore, deep learning comes into play such that it allows the computer to learn without human intervention and features are extracted at every intermediate layer of the convolutional neural network.
We are looking to encounter these issues by convolutional neural network which in essence does the direct exploitation of images and discards the need of handcrafted features by experts in the domain.

In this project, a deep learning system is designed to detect and classify the plant diseases among 38 classes. This will act as a way to alert the farmers about the disease at the right time and take precautionary steps accordingly to prevent the spread of disease. For ensuring the plant growth management, disease control, disease detection, the early detection of plant leaf diseases will act as a treasured and worthwhile source of propaganda. A lay-man needs to be informed about the symptoms present and the process of the detection of the disease. In this way, even inexperienced people in agriculture sector, can gain insights about the plant leaf disease. The following study will show the detection and classification of plant leaf diseases:

1. Comparison of the deep learning architectures (ResNet, AlexNet, Denset and a Model trained scratch) of three versions of images: color, grayscale, segmented of the plant leaf disease dataset.
2. Visualization of the infected plant leaf diseases in the intermediate layers of the deep learning architectures.
3. Detection of the infected plant leaf diseases by the bounding boxes (grid) that are manually annotated.

The use of deep learning image processing algorithms and techniques aids us to segment the disease from the leaves by the method of classification and detection (localization) of plant leaf diseases. Through the different methodologies incorporated, the plant disease can be identified at an early stage such that the pest and infection control tools can be used to solve the pest problems while diminishing the likelihood of the risks to the people and the surrounding.

Literature Survey

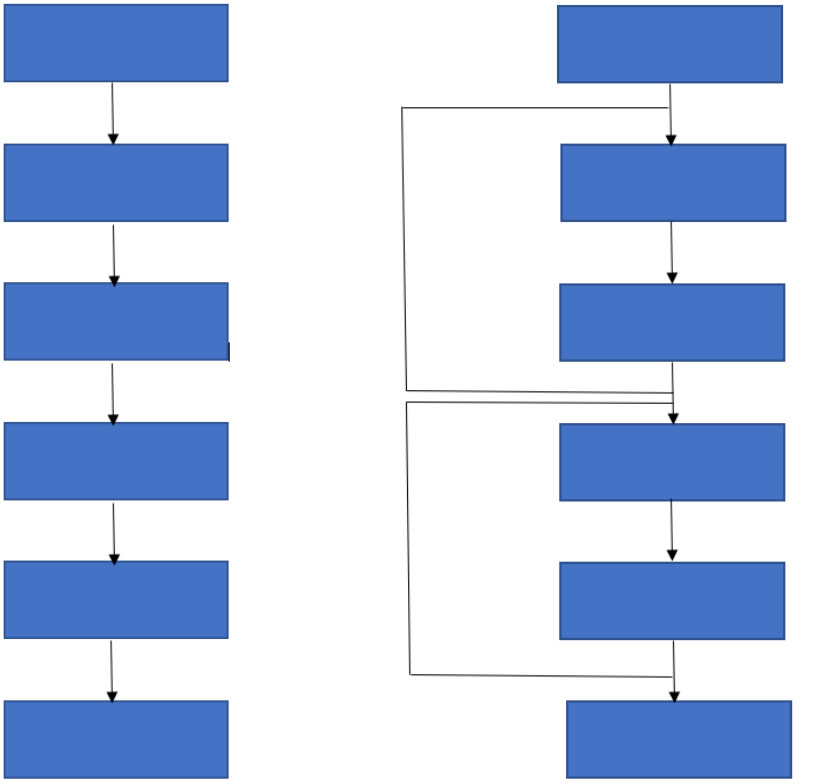
There has been a great deal of advancement in the object recognition and image classification systems using convolutional neural networks. Earlier, feature engineering was widely used for image classification tasks. Hand Engineered Features such as SIFT, SURF and HoG followed by the application of learning algorithm in these features spaces were being used in the past but these techniques were very tedious and complex and required to be revisited every time the dataset or the problem changed. This was one of the common problems observed in all the traditional attempt of plant disease detection using computer vision as they were heavily dependent on the featured engineering. With the introduction of ConvNets, the cumbersome phase of hand-engineered featuring was no more required and the bottle-necks of revisiting these processes with every new problem were eliminated and thus deep convolutional neural networks seemed promising candidates for these kind of practical problems.
Before diving deep into the plant disease detection, there was another problem of identifying various plant species that need to tackled. There has been a lot of work which was already there dealing with this problem. One of the most popular work in this regard was the Gray-Level co-occurence matrix (GLCM), which dealt with how often the pair of pixel with specific values occur in a

spatial relationship in the image. Other than that in regard to Plant Disease Detection there has been an implementation using the HSV features and SVM where the neural networks are used to classify whether the leaf was defected or healthy. DCGAN along with resnet[2] has also been used for Plant Disease Classification which gave the accuracy of 78%.
Another paper[1] has implemented AlexNet and GoogleNet network architectures for Image-Based Plant Disease Detection on three visual representations of the image data- Colored, Grayscale and Segmented. The Plant Village dataset was used for all the visual representations discussed above and the achieved accuracy varied from 85.53% to 99.34% (this was achieved by GoogleNet, transfer learning on colored dataset).

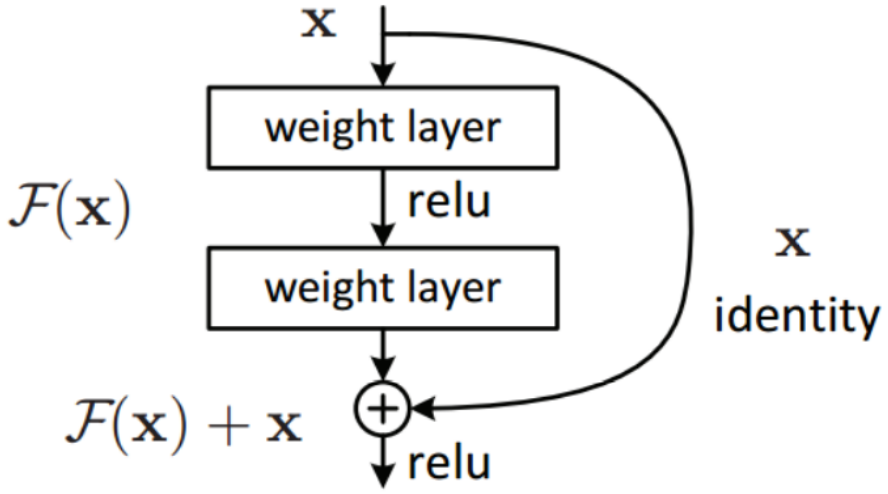
Network Architectures

ResNet

Deep convolution neural networks have proven to be very successful for image classification. Many visual recognition tasks have been greatly benefited from deep neural networks. Deep convolutional neural networks have led to a series of breakthroughs for image classification. Many other visual recognition tasks have also greatly benefited from very deep models. As we go deeper in a network, the accuracy of classification/recognition improves greatly. But, we are also caught into an issue where the accuracy starts saturating and eventually degrades when we go deeper into the network. It gets difficult to handle propagating gradients through so many deep layers. As we go deep into the network, derivatives start diminishing, this phenomenon is known as the vanishing gradient problem. Residual learning helps in tackling this problem by developing residual connections between a layer and layers after the next. The difference between residual network and a plain network is explained in the diagram below.



The diagram above implies that the plain network connections are just capable of directing information from one layer to another (continuous connection) whereas the residual network connection is capable of taking the future map from layer, say A and add it to the output of layer A+2. The residual learning building block has been well defined in [4] with the diagram below :



Suppose we consider a neural network block wherein the input is x and say we would like to learn the true desired output, H(x). The difference between H(x) and x can be represented as:

$$F(x) = H(x) - x$$

We can rearrange the above equation and obtain the following:

$$H(x) = F(x) + x$$

This implies that the residual block, F(x) is attempting to learn the desired output H(x). The image above has an identity function for x, which means the layers in the network are actually trying to learn F(x). This brings us to the fact that the layers in a residual block are trying to learn the residual function F(x). ResNet is a collection of residual blocks. We use ResNet-152 for the classification of plant diseases. A reason to choose ResNet-152 for classification of plant diseases is based on the fact that residual networks improve the performance of both shallow and deep neural networks.

AlexNet

We experiment the use of deep neural networks in the classification problem with the AlexNet model. AlexNet is a deeper model with more filters per layer and they are also stacked with convolution layers. The model is made up of 5 convolution layers, 3 fully connected layers and a softmax layer in the respective order. The first two convolution layers are each followed by normalization and a pooling layer. The final output layer of AlexNet has 1000 outputs in the original version, we modify it according to the number of classes in our dataset. The layers of AlexNet have a ReLu non-linearity activation associated with them,

the fully connected layers (first 2 out of the 3) also have dropout associated with them. One of the main characteristics of AlexNet according to [5] is the expeditious down-sampling of the intermediate layer outputs with strided convolutions and max-pooling layers.

DenseNet

DenseNet network (Dense Convolution Network) is an extension of ResNet. The core difference between ResNet and DenseNet is that DenseNet concatenates the outputs from previous layers instead of the summation operation that is done in ResNet network. The DenseNet model couples each layer to the other layers present in a feed-forward manner. Suppose we consider a network with N layers, a traditional neural network will tend to have N layers with N connections. On the other hand, a DenseNet model will only have N(N+1)/2 connections. This places DenseNet model at an advantageous situation as they reduces the vanishing gradient problem, intensifies feature propagation, and reduces the number of parameters. DenseNet architecture is mainly composed of Dense blocks where in the layers are densely coupled together and each layer receives in input all previous layers output feature maps. A dense block is a group of layers connected to all the respective previous layers. A single layer in a dense block contains Batch Normalization, ReLU activation and 3x3 convolution. The DenseNet model has something known as Transition layers between dense blocks. A transition layer consists of batch normalization, 1x1 convolution and average pooling. Transition layer in DenseNet model helps in the process of down sampling. The number of output maps generated in a layer is know as the growth rate and a DenseNet model generates a lower number of feature making its growth rate low. The model also has something called the Bottleneck which makes sure that DenseNet model maintains a fewer parameters in spite of the concatenation operation. A layer in dense block with bottleneck will contain batch normalization, ReLU activation, 1x1 convolution layer, batch normalization, ReLU activation and 3x3 convolution. The 1x1 convolution is the bottleneck layer.The compactness of the model is further improved by a compression in the transition layer. We have used DenseNet-169 architecture for our project. The DenseNet architecture configurations are well explained in [6]. Below is a picture from the referenced paper written on DenseNet.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Each Conv layer in the above image has a batch normalization , ReLu activation and a convolution layer within it.

Training from scratch model

In order to experiment with our understanding in convolution neural networks, we tried to develop a model from scratch and train it on the dataset. The model consist of 5 layers. The convolution layer is followed by ReLU activation and batch normalization 2d in alternate layers. The network consists of 2 fully connected layers and a log softmax layer in the end. We also added layers such as max pooling and dropout (in the last stage) for better performance.

Experiments

PlantVillage is a not-for-profit project by Penn State University in the US and EPFL in Switzerland. There are 21,730 images of 14 crop species with 38 crop diseases made openly available through the project PlantVillage. The open-source plant leaf disease dataset that we used for the project is available in the following link: Click here to view the dataset . We performed the split of the data as follows (the code is attached) in the folder 60_20_20:
60% - Train
20% - Test
20% - Validation

Data Preparation and augmentation

We have manually annotated the plant leaf images (localization of the infected parts of the leaves) containing the disease with the bounding boxes and label names denoting the disease. This was done to experiment with bounding box prediction using pre-trained model and visualization of the bounding box prediction.
To show some characteristic distinction in the performance of the deep learning

Figure 1: Plant leaf images denoting every plant leaf disease.

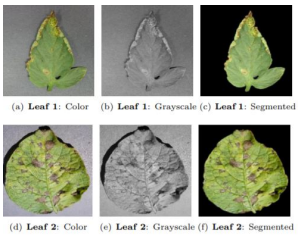


Figure 2: Three different versions of the plant leaf images.

Figure 3: Bounding box manual annotation (using LabelImg

models, we considered three different versions of the leaf images, that is, color (original image), grayscale (converted from RGB to grayscale), and segmented image. The above Figure 2 (which is taken from a paper) is how we have also converted the original dataset (color) into grayscale and segmented images to have three different visual forms. We performed segmentation through reference of the paper[7].
We considered grayscale image to determine the ability to adapt to high level features and patterns in the absence of the color pertaining to certain crops and plants. The segmented configuration of the images were used to evaluate the performance with the foreground images that is, only the leaves being present with the black background.
The images were randomly cropped to be 224*224, horizontal flipping was performed and also random rotation at 30 degree angle was done on the train set. Center cropping and resizing of the images were some of the data augmentation techniques that were done on the validation and test set for evaluation of the performance.

Image Analysis

The main goal of this project is to detect and classify the plant disease present in the image. The following are the different transfer learning techniques, bounding box prediction model, visualization method and the analysis of arriving at the desired hyperparameters for this project.

ResNet

We have used the pre-trained ResNet model with 152 layers. Since the dataset is similar to the pre-trained model's dataset, we followed one of the strategies of transfer learning wherein all of the first few layers are frozen (i.e. the initial convolutional base). This is because initial few layers capture universal features like curves and edges that are relevant to this dataset as well. The weights of these layers are preferred to remain the same. To focus on the high level features of the dataset in the subsequent layers, we added 2 layers that is,

1. Fully connected layer followed by ReLU
2. Fully connected layer followed by a log softmax layer.

The output of the log softmax layer is finetuned such that output is the number of the classes which is 38 for this dataset.

We used log softmax layer in the model so that there is the effect of penalizing the model heavily when a correct class is wrongly predicted. Negative log likelihood loss was chosen since log softmax has been used in the last layer of this model. Since the log softmax pushes the value in the range (-inf,0) for which negative log likelihood is the ideal loss function used in this case. Since the dataset is pretty huge, on trying different epochs, we found that with epochs=10, the accuracy started to become consistent and it converges at an earlier stage. Initially, on trying different learning rates like 0.01, 0.001, 0.0001, 0.00001, etc, we narrowed down the learning rates to two values that worked better than others.
The following are the hyper-parameter settings for achieving a good accuracy on this model:
Batch size: 84
Loss: NLLLoss
Learning rate: 0.001
optimizer: Adam
epoch: 5
Accuracy with different configurations on ResNet152 model

Versions of image	Adam+lr=0.001	Adam+lr=0.0001	SGD+mom+lr=0.001	SGD+mom+lr=0.0001
Color	0.938728	0.918011	0.885906	0.887796
Grayscale	0.856477	0.834409	0.805598	0.812381
Segmented	0.880005	0.926915	0.8357	0.364699

AlexNet

We experimented with two different approaches of transfer learning for AlexNet. One, we tried doing a shallow model training where only the fully connected layers are fine-tuned and the rest of the network acts as a feature extractor. And the other one is deep strategy is where the back propagation optimization starts from the pre-trained network.
The intermediate layers' (that is, 6th, 8th and 10th layers) parameters were set to true for back propagation and updating weights.
Here, we considered cross entropy loss for the classification of the models to move the parameters towards the optimum values. It minimizes the distance between two probability distributions - predicted output and ground truth which is our main requirement.

Challenges

For the deep model, it was noticed that the learning rate of 0.001 was producing insufficient disk allocation for the GPU. We faced some problems on running the deep model of AlexNet on hyper-parameter settings.

On trying with 0.0001 and 0.00001 learning rate, the accuracies that we got are mentioned in the table below.
Hyperparameter settings that worked really well for our dataset for shallow model is as follows:
Batch size : 84
Epochs : 10
Loss : Cross Entropy loss
Optimizer : SGD + momentum (0.9) with learning rate decay
Learning rate: 0.01
The following summarises the different configurations that we had tried for this model.
Accuracy with different configurations on shallow type for AlexNet model.

Versions of image	Adam+lr=0.001	Adam+lr=0.0001	SGD+mom+lr=0.01 (decay)	SGD+mom+lr=0.001 (decay)
Color	0.934809	0.956831	0.966315	0.937935
Grayscale	0.819358	0.860515	0.883589	0.852653
Segmented	0.8934	0.9229	0.929025	0.911390

Hyperparameter settings that worked really well for our dataset for deep model is as follows:
Batch size : 84
Epochs : 10
Loss : Cross Entropy loss
Optimizer : Adam
Learning rate: 0.0001
Accuracy with different configurations on deep type for AlexNet model.

Versions of image	Adam+lr=0.00001	Adam+lr=0.0001	SGD+mom+lr=0.001(decay)	SGD+mom+lr=0.01(Decay)
Color	0.947386	**0.970590**	0.955990	0.960749
Grayscale	0.881114	**0.899955**	0.875171	0.889308
Segmented	0.902941	**0.920776**	0.927813	0.940920

DenseNet

We experimented with Densenet pretrained model with 169 layers. We performed shallow strategy of transfer learning rather than deep model.
Hence, we worked on the shallow network architecture of the densenet model, where all the first few layers are frozen, with some fine-tuning and additions done to the network. The additional layers are:

1. A fully connected layer followed by ReLU and dropout.
2. A Fully connected layer followed by Log Softmax layer with 38 classes.

Log Softmax has been used even in the densenet model for better optimization strategy of penalizing the weights with the wrong prediction. Since log softmax is used, Negative log likelihood loss is being considered in this case.
The configuration that worked well for this shallow model is:
Batch size: 84
Learning rate: 0.0001
Loss: NLL loss
Optimizer: Adam
Epochs: 10

Challenges

We encountered few issues with regards to training the deep model. One of the problems was the allocation of the disk usage for the GPU.
Accuracy with different configurations on DenseNet model

Versions of image	Adam+lr=0.001	Adam+lr=0.0001	SGD+mom+lr=0.001	SGD+mom+lr=0.0001
Color	0.918509	0.922562	0.879158	0.380811
Grayscale	0.830026	0.808227	0.762402	0.329677
Segmented	0.907669	0.890717	0.848799	0.343353

Training from scratch

We developed a deep CNN model which consists of 5 layers. Here, there is a convolutional layer followed by relu followed by batch normalization 2d. There are two linear layers towards the end of the architecture with the log softmax layer. Hence, NLLLoss is being used here. Through this model, we got to know many insights about the dataset and the hyperparameter settings.
The configuration that worked well for this from the scratch model is:
Batch size: 84
Learning rate: 0.001
Loss: NLL loss
Optimizer: Adam
Epochs: 10

Challenges

The main concern with this model was the in channels and out channels for every convolutional layer. We had a hard time and had to do a lot of trial and error to figure out the right set of in channels and out channels. On increasing the in channels in the further layers, the test and val set did not perform well. On trying out the lower number of in channels and out channels we arrived at the best accuracy possible for the self customised model.
Accuracy with different configurations on training from Scratch model.

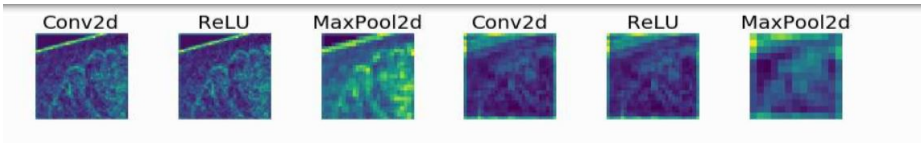
Versions of image	Adam+lr=0.001	Adam+lr=0.0001	SGD+mom+lr=0.001	SGD+mom+lr=0.0001
Color	0.863233	0.820783	0.685488	0.387775
Grayscale	0.632814	0.632895	0.586466	0.294218
Segmented	0.777770	0.735826	0.681130	0.365951

Visualization of the output of various layers in AlexNet

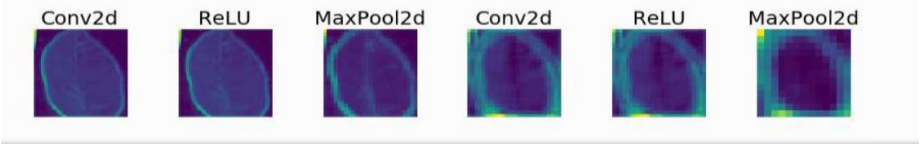
Visualizing the output of the various layers provides a way to display the feature maps. This gives us an idea of the learning of the network of how the input is decomposed into different filters. Since every channel encodes independent features, the best way that these outputs are visualized, is by plotting the contents of every channel as a 2d image.
The following are the interpretations of the various layers of the AlexNete model.

- The first layer that is convolutional layer almost retains all of the leaf features present in the image. There are some filters which are not activated.

- The activation and the features become abstract and less visually explicable. It can be noticed from the below visualizations that going deeper in the models, lesser information about the image is depicted and features are increasingly focused about the classes of the image.
- Higher level features like borders, corners and angles of the leaves are visible in the visualizations of the deeper layers.



Visualization of the output of some of the layers in AlexNet model of training images



Visualization of the output of some of the layers in AlexNet model of an image belonging to a different class

Detection of Bounding box using ResNet18 model

We manually annotated the plant leaf images using LabelImg software with multiple bounding boxes around the infected parts of the leaves. We got the XML files with bounding box coordinates after doing the manual annotation. We mapped the image with the bounding box coordinates and the label names in the data pre-processing step. We considered the grid size to be 14*14 so that we can use the ResNet18 pre-trained model. We discarded last few layers of this model and further added a convolutional layer to have only one output channel. MultiLabelSoftMargin loss is chosen for the detection of bounding boxes since it performs the optimization for the multilabel one vs all. It performs the sigmoid (logits functionality) and it is based on max entropy.

Based on the analysis below, we inferred that the model with optimizer SGD + momentum was performing worse than than the model with adam.
Also, through trial and error, we inferred that The following are the hyperparameter configurations that worked really well.
Number of epochs: 40
Loss: Multi label soft margin loss
Batch size: 12
Optimizer : Adam
Learning rate: 0.0001
epochs=20 and batch size=28

Image type	Adam+lr=0.0001	Adam+lr=0.001	SGD+mom+lr=0.001	SGD+mom+lr=0.0001
Color+bounding boxes	0.90267	0.8964	0.57570400	0.5332704521

epochs=40 and batch size=12

Image type	Adam+lr=0.0001	Adam+lr=0.001	SGD+mom+lr=0.001	SGD+mom+lr=0.0001
Color+bounding boxes	0.93874	0.8964	0.91911232	0.540951

Metrics

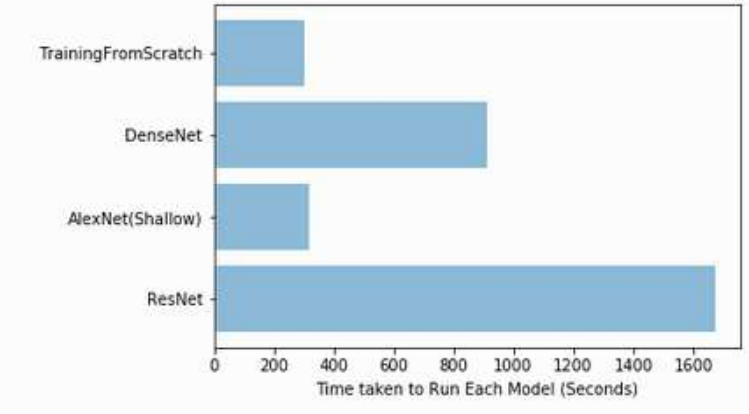
The performance metrics that we used for the evaluation of the classification and detection of the plant leaf disease are:

- **Accuracy**

Models / **Datasets**	**ResNet**	**AlexNet**		**DenseNet**	**Training From Scratch**
		Deep	**Shallow**		
Colored	0.938728	0.970590	0.966831	0.922562	0.863233
Grayscale	0.856477	0.899965	0.883589	0.830026	0.632895
Segmented	0.926915	0.940920	0.929025	0.907669	0.777770

Table 1: Accuracy of Different Models on Different Datasets

- **Area under Receiver Operating Characteristic (AUROC)** Color+bounding boxes gave an AUC score of 0.93.
- **Comparison of Running Time of all the models**



BarChart representing run time for each model

- **Macro and micro F1-score** Macro F1 score performs the computation of the metric for each class and then considers the average whereas the micro f1 score performs the overall average metric i.e., the aggregation of all classes.

Models / Datasets	ResNet	AlexNet		DenseNet	Training From Scratch
		Deep	Shallow		
Colored	0.921645	0.960299	0.948217	0.881152	0.822225
Grayscale	0.796710	0.83107	0.83208	0.754084	0.514897
Segmented	0.910712	0.9277475	0.90944533	0.877876	0.708549

Table 2: F1-score(Macro) of Different Models on Different Datasets

Models / Datasets	ResNet	AlexNet		DenseNet	Training From Scratch
		Deep	Shallow		
Colored	0.939574	0.970934	0.963132	0.923512	0.865075
Grayscale	0.858191	:0.87501	0.87501	0.832186	0.637601
Segmented	0.927718	0.941807	0.929862	0.909035	0.780857

Table 3: F1-score(Micro) of Different Models on Different Datasets

Conclusion

This project has helped us understand how different ImageNet models perform the classification tasks on a dataset. We were able to understand the significance of Hyperparameters and how varying them varies the efficiency and accuracy of the different models. By running different models- Resnet, AlexNet, DenseNet and a model trained from scratch we could infer that AlexNet deep Model gives the most promising results not only in terms of accuracy but also in terms of the time it took to run. We were able to achieve and accuracy of 97.059% using Adam optimizer and learning rate of 0.0001 for the colored dataset.Also, the visualization of the intermediate layers helped us in understanding the feature maps extracted out of every layer and also the role of activations throughout the network.The bounding box prediction was producing average results due to a small set of manually annotated dataset. Out of the predicted positive results, the accuracy came out to be high since the positive results were almost on par with the ground truth positive classes (i..e infected regions).
Finally, in future work the following things could be experimented-

1. The model from the scratch could be made more dense by experimenting around with additional layers. Also, the model could be trained on a much larger dataset and a more insightful study could be carried out to optimize models which could serve well specifically for this problem set.
2. GoogleNet would be another model that one could implement and study how to model performs. Because of the GPU restrictions we weren't able to successfully deliver the analysis for GoogleNet
3. If given more powerful GPUs the dense models for denseNet and ResNet could be experiemented upon.
4. Manually annotate many images so that there will exist a huge dataset to perform bounding box prediction.

References

[1] Prasanna Mohanty, Sharada and Hughes, David and Salathe, Marcel Using Deep Learning for Image-Based Plant Disease Detection arXiv preprint arXiv:1604.03169 2016

[2] Cortes, Emanuel Plant Disease Classification Using Convolutional Networks and Generative Adverserial Networks Stanford University Reports, Stanford 2017

[3] Singh, Vijai and Misra, Ak K Detection of plant leaf diseases using image segmentation and soft computing techniques Information processing in Agriculture, 2017

[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[6] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700-4708. 2017.

[7] Automatic Leaf Extraction from Outdoor Images Nantheera Anantrasirichai*, Sion Hannuna and Nishan Canagarajah Merchant Venturers' School of Engineering, University of Bristol, UK