Machine Learning Engineer Nanodegree

Plant Seedlings Classification Capstone Project

Gajula Karthik Kumar
Jan 7th, 2019

1. Definition:

1.1 Project Overview:

The project has derived from the agricultural domain, our goal is to classify the plant seedlings which is very helpful for the formers in differentiating a weed from a crop seedling. The data for this project has collected from kaggle competitions the datasets are provided by The Aarhus University Signal Processing group, in collaboration with University of Southern Denmark which are images of different types of plant seedlings. The images has taken from the commercial camera in a setup, containing images of approximately 960 unique plants belonging to 12 species at several growth stages.

The publicly available database found in Sderkvist (2001) but their usability for developing machine vision systems was yet to be determined. Botanists have been dealing with species categorization for centuries and substantial progress is reported in the field of content-based image retrieval and analysis of images and video.

The present authors believe that one problem is a lack of benchmark databases. Several studies on species recognition contain a description of preprocessing steps such as image acquisition, segmentation and annotation which suggest that researchers have spent time on these topics although each of these tasks is an area of its own. The paper https://arxiv.org/pdf/1711.05458.pdf describes the difficulties about the getting the data and many useful techniques to which will helpful to classify plant seedling in high intra-class variations, it has also documented the procedures they have gone through to create the database which gives insight for many researchers. The paper also describes the implementation of the Naive Bayes technique and the F1 score used as the metric to measure the performance. The page 12 of the paper references many good works from different authors which can be useful for the research in this field and they also can provide great technique and insights helpful for the tasks in this field.

1.2 Problem Statement:

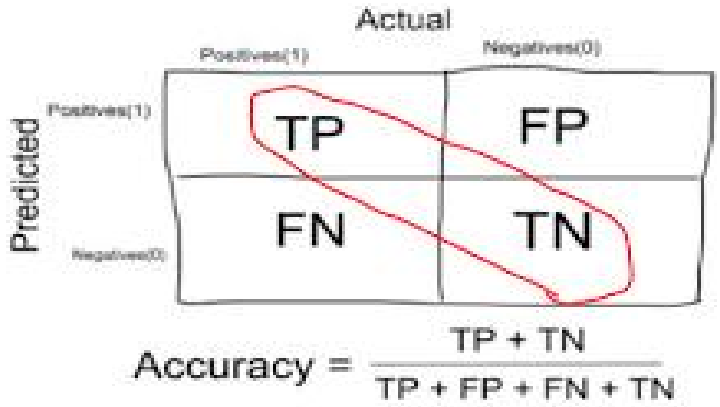
Since this project is intended to classify the plant seedlings based on the image data for which we have considered Convolution neural network model which is best suitable for the image classification tasks. The convolutional neural networks determines the hidden patterns using various types of filters. The accuracy of the predictions on the tests sets used as metric to measure the success of the model. We have used categorical cross entropy as to predict the loss.

1.3 Metrics :

The evaluation metrics we have used is Accuracy score which is enough to quantify the performance of the model. The Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Higher value of the accuracy score indicates the higher accuracy in the prediction. For image classification task Accuracy is very good metrics.

Accuracy is a good measure when the target variable classes in the data are nearly balanced and during our data exploration we have found that our 12 target variable classes are nearly balanced, which is one of the reason for selecting accuracy as our metric.

The formula for binary classification is as follows.



2. Analysis:

2.1 Data Exploration:

The input data are images of approximately 960 unique plants belonging to 12

species at several growth stages. We were provided with a training set and a test set of images of plant seedlings at various stages of grown. Each image has a filename that is its unique id. The dataset comprises 12 plant species. Further to our investigation we have found that all these images have taken from commercial camera setup. The images in each classes are as below.

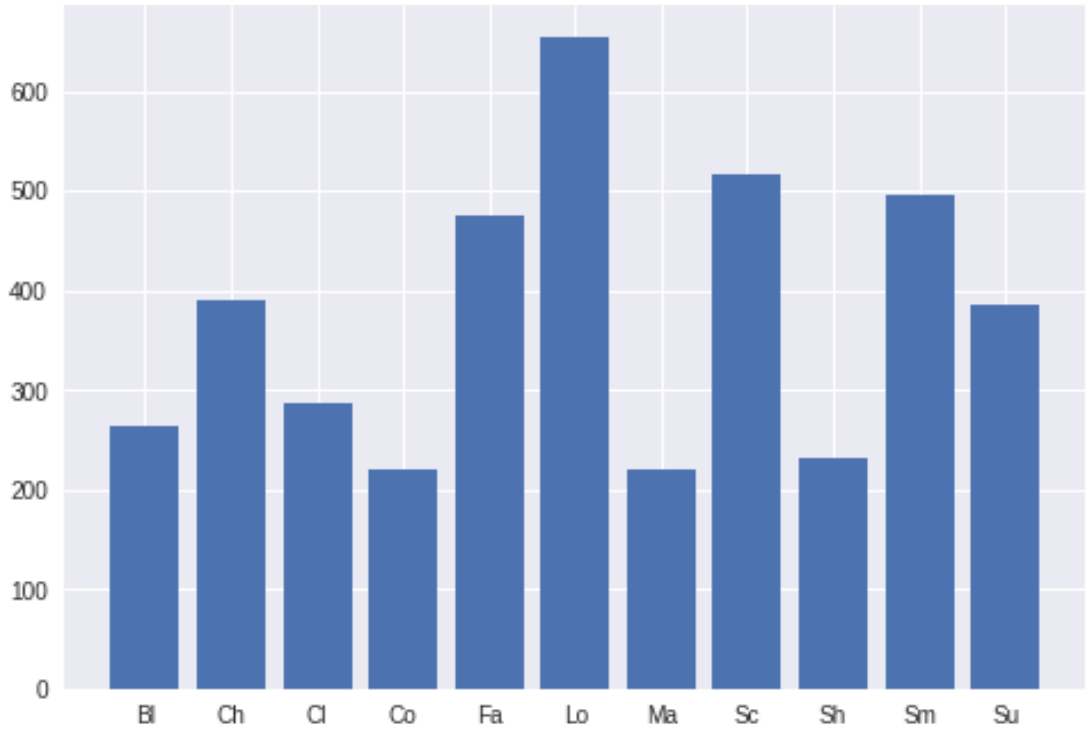
1) Common wheat -> 221
2) Black-grass -> 263
3) Charlock -> 390
4) Scentless Mayweed -> 516
5) Maize -> 221
6) Cleavers -> 287
7) Loose Silky-bent -> 654
8) Small-flowered Cranesbill -> 496
9) Shepherds Purse -> 231
10) Sugar beet -> 385
11) Common Chickweed -> 611
12) Fat Hen -> 475

The dataset contains 4750 annotated RGB images with a physical resolution of roughly 10 pixels per mm.

2.2 Exploratory Visualization:

The dataset consists of 4750 images of plant seedlings images across 12 different categories. It is usually very tough for a human being to classify the plant seedlings into different species by just looking over the images. The following image shows 12 example images for each category.

By just looking over the images many of them look same. Hence we use Convolutional Neural Networks to classify the species. To classify and predict the species, the dataset provided should have considerable number of labeled training dataset. The dataset has enough number of samples across each category to choose CNN architecture. The Following figure shows the distribution of number of samples across each category.



2.3 Algorithm and Techniques:

Convolution Neural Networks are a type of deep neural networks that are especially used for image recognition tasks. In CNN, the convolutions are performed on the input data with the use of a filter to produce feature maps. Numerous Convolutions are performed on the image where each operation uses a different unique filter. These filters are randomly intialised at the start and are learned through the network later through multiple forward and back propagation algorithms using gradient descent algorithms. Each filter convolves on input performs a weighted sum and pass it through a activation function to obtain a non-linear output. There will such numerous filters in a layer and we can add as many as layers. In the very first layers the filters detects minor properties like line,edge, blobs of colors. Likewise, the later layers detect some important characteristics like eyes, beaks, feathers which help in recognition of objects and all these are fed to fully connected layers which is then passed through activation functions(mostly softmax in

classification problems) which provides an probability of classification.

INPUT IMAGE

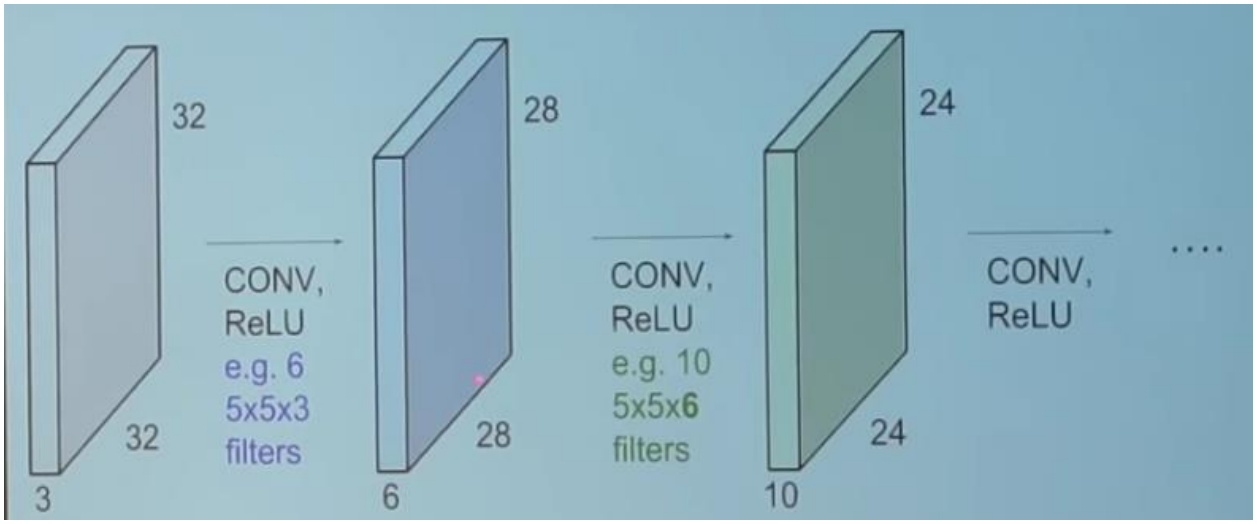
18	54	51	239	244
55	121	75	78	95
35	24	204	113	109
3	154	104	235	25
15	253	225	159	78

WEIGHT

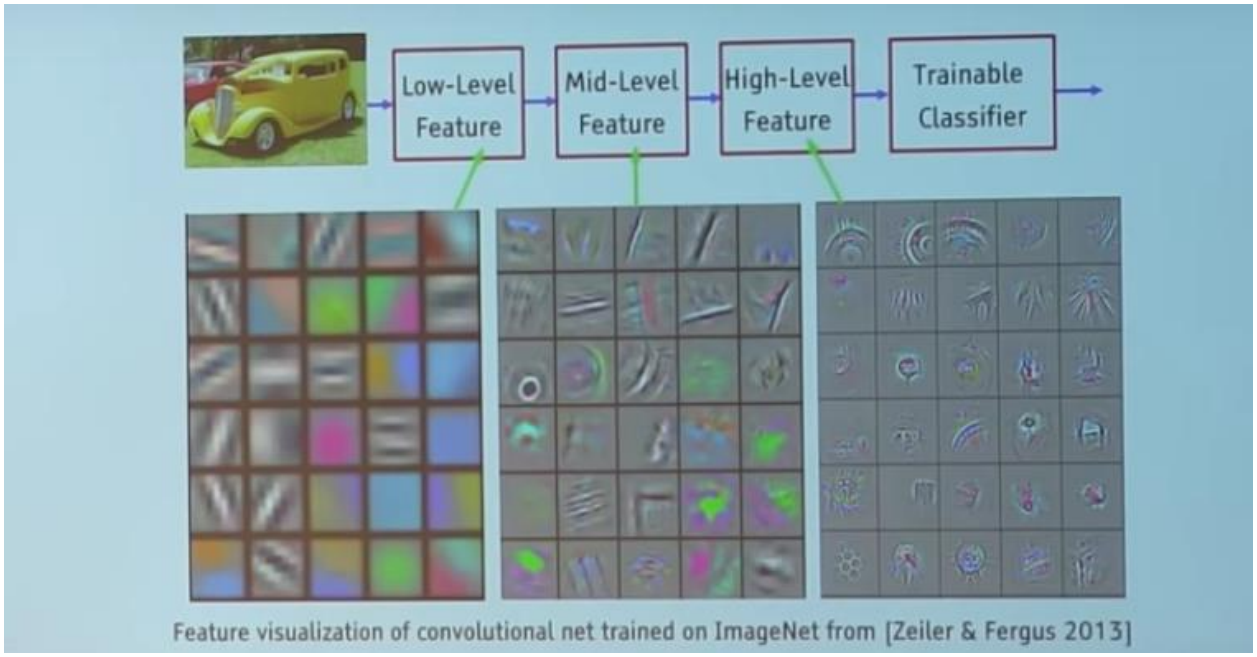
1	0	1
0	1	0
1	0	1

429

Convolution of filter on input image.



Convolution layers



Filters in a trained network

Convolution Layer:

The Convolution Neural Network consists of several layers. The layer has 3 dimensions width,height and depth. The neurons in one layer are not connected to all the neurons in the next layer but only to a fewer regions. The final output will be reduced to a single vector of probability scores. In CNN layers we can tune the hyper parameters like filter size, number of filters, stride, padding and activation functions.

Max Pooling:

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation reducing its dimensionality of the images by removing some of the pixels from the image.

DropOut:

Dropout is a simple and effective technique to prevent the neural network from over-fitting during the training. Dropout is implemented by only keeping a neuron active with some probability p and setting it to 0 otherwise. This forces the network to not learn redundant information.

Global Average Pooling:

Like Global Average Pooling layers minimize over-fitting by reducing the total number of parameters in the model.GAP reduces the dimensions of the tensor from h x w x d to 1 x 1 x d.

Dense *:*

Dense layers are the fully connected networks each input node is connected to every other output.The last dense layer maps the scores of the convolutional layers into the correct labels with softmax activation function.

ReLu activation:

The ReLU also known as *Rectified Linear Unit* is the most used activation function. Relu computes the function f(z). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Softmax activation:

Softmax activation is a more generalized logistic activation function used in multiclass classification.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{j} e^{y_j}}$$

RMSProp:

RMSProp (RMS stands for Root Mean Squared Error) decreases the learning rate by dividing it by an exponentially decaying average of squared gradients.

Batch normalization is a technique for improving the performance and stability of neural networks. It is a technique to provide any layer in a neural network with inputs that are zero mean/unit variance. It is normalization of hidden layers. Similar to normalizing input data, where the values have a wide range are reduced to a smaller range usually between 0 and 1, the batch normalization is applied for hidden layers which speeds the training by reducing
to a smaller range values.

Our input data are images hence we have considered Convolution neural network as our model. Convolutional neural networks model is one of the efficient method for image classification tasks. The convolution layers find the different shapes of the objects in the image and compare it with respective label to train the network, this process continues with forward and back-propagation where the weights updated in each epoch to get the final weights.

The details of the model we have considered.

Batch size = 20
Activation function of convolution layers = Relu
No. of Convolution layers = 5
No. of Pooling layers = 4
No. of Dense layers = 2
Activation function of convolution layers = Relu and soft max respectively
No. of Drop out layers = 2 with probability 0.2, 0.3 respectively.
Loss function = Categorical Cross Entropy.
Optimizer - rmsprop.
Metrics - Accuracy score.

Since the task here is classification of plant seedlings from the input images of plants, we need to scan through the image data in training sets and we need to

get weights and bias from training data from which we can predict the class of the new image of plant. The convolution layer is very successful model for this kind of application because it uses the filters of various sizes with weights which scans through the images in the training sets. Since we get chance to define the number of filters in each layers and their scan patterns (strides parameter) after each layer it produces the feature map from that convolutional layer.

The combination of convolution layers will be able to detect the patterns in the image data after many updation of the weights by front and backpropogation, from which the prediction of the training images takes place and it will be compared to the original label to calculate the accuracy score.

2.4 Benchmark Model:

We have considered prestigious Kaggle competitions https://www.kaggle.com/mnishant2/custom-vanilla-cnn-0-91-f1 kernel as our benchmark model which implements the vanilla convolution neural network model on the same datasets. The benchmark model has utilized the image augmentation technique in the preprocessing stage of the data along with normalization technique and the model has also well described the 4 convolutional layers along with pooling layers. The benchmark model has produced acceptable accuracy of 77% which is good metric to compare to our model.

3 . Methodology:

3.1 Data Preprocessing:

Once the most appropriate raw input data has been selected, it must be preprocessed, otherwise the neural network will not produce accurate forecasts. The decisions made in this phase of development are critical to the performance of a network. Transformation and normalization are two widely used preprocessing methods. Transformation involves manipulating raw data inputs to create a single input to a net, while normalization is a transformation performed on a single data input to distribute the data evenly and scale it into an acceptable range for the network.

We have imported the datasets from the kaggle data source and unzipped the files to use in our project and we populate number of images in training dataset

and number of species in plant seedlings, get the ideas of number of images and number of images in each classes which is very important to verify the data.

The data only have images for the training and we have to split the data into training and testing sets for that purpose, we have used the train_test_split from Sklearn library.

To load the data to Convolutional neural network the data should be in tensors. When using Tensor Flow as back-end, Keras CNNs require a 4D array (which we'll also refer to as a 4D tensor) as input, with shape (nb_samples, rows, columns,c hannels). The path_to_tensor function takes a string-valued file path to a color image as input and resizes the image to (51, 51) pixels and returns a 4D tensor suitable for supplying to a Keras CNN. Then normalization of the input will be done which makes the algorithm work efficiently.

3.2 Implementation :

The preprocessed and normalized data loaded to convolution neural network, here we have given the image with the shape (51, 51, 3) as input to our first layer which has 16 filters each filter with size (2,2) which uses single stride across the tensors with activation function relu .

The second layer has 32 filters each filter with size (2,2) which uses single stride across the tensors with activation function relu. We have used Max-pooling layer after the second layer with size (2,2) which uses two stride across the tensors and which will removes the unwanted spatial information and reduces the no of parameters.

The third layer has 64 filters each filter with size (2,2) which uses single stride across the tensors with activation function relu. We have used Max-pooling layer after the third layer with size (2,2) which uses two stride across the tensors, which will removes the unwanted spatial information and reduces the no of parameters.

The fourth layer has 128 filters each filter with size (4,4) which uses single stride across the tensors with activation function relu. We have used Max-pooling layer after the fourth layer with size (2,2) which uses two stride across the tensors and which will removes the unwanted spatial information and reduces the no of parameters.

The fifth layer has 256 filters each filter with size (2,2) which uses single stride across the tensors with activation function relu. We have used Max-pooling layer after the fifth layer with size (2,2) which uses two stride across the

tensors and which will removes the unwanted spatial information and reduces the no of parameters. To further reduce the no, of parameters whose affect is

very less for the output we have used Global average pooling layer.

The output of the fifth layer then flattened and given to the fully connected network of 200 nodes in which probability of 20% nodes random dropping of nodes has taken and we have used same relu activation function which makes the network faster and avoids the vanishing gradient for some great extent.

The last layer of the network is also fully connected network which receives input from previous layer and has 12 nodes (due to 12 classes), we are using soft max as activation function here which gives the probabilities of each classes.

We are using rmsprop as our optimizer which updates the weights after each epoch and we are using categorical cross entropy as our loss function to define the loss function and to measure the performance of our model we have considered accuracy score which is simply a ratio of correctly predicted observation to the total observations.

The summery of the above explained model is as follows:

Layer	(type)	Output Shape	Param #
conv2d_1	(Conv2D)	(None,50,50,16)	208
conv2d_2	(Conv2D)	(None, 49, 49, 32)	2080
max_pooling2d_1	(MaxPooling2)	(None, 24, 24, 32)	0
conv2d_3	(Conv2D)	(None, 23, 23, 64)	8256
max_pooling2d_2	(MaxPooling2)	(None, 11, 11, 64)	0
conv2d_4	(Conv2D)	(None, 8, 8, 128)	131200
max_pooling2d_3	(MaxPooling2)	(None, 4, 4, 128)	0
conv2d_5	(Conv2D)	(None, 3, 3, 256)	131328
max_pooling2d_4	(MaxPooling2)	(None, 1, 1, 256	0
global_average_pooling2d_1		(None, 256)	0
dropout_1	(Dropout)	(None, 256)	0

dense_1	(Dense)	(None, 200)	51400
dropout_2	(Dropout)	(None, 200)	0
dense_2	(Dense)	(None, 12)	2412

Total params: 326,884, Trainable params: 326,884, Non-trainable params: 0

As a beginner I have come across some of the issues while doing project which can be useful feedbacks for the others. Firstly when I found the plant seedlings classification problem, I tried to implement the model in Kaggleôs Kernel which was refer to the Jupiter notebook. I faced sudden shutdown of the kernel due to the CPUôs memory limitations. Then I tried on googleôs Colab. I spent most of the time doing research for importing the data to the colab. Later I found a way to import the kaggleôs dataset to the colab from one of the solutions in stack-overflow answers. Once I imported the data everything was easy until I implemented CNN architecture. The model was performing poorly. Later I normalized the dataset and implemented CNN. Then I got a good training accuracy but a very low test accuracy. because the model was over-fitting, I used dropout layers and max pooling layers in between to reduce over-fitting. I implemented Batch Normalization to further improve the model.

3.3 Refinement:

Our model initially produced lot of parameters due which training got lot slower and resource consumption had increased, so we used the Max-pooling layers after the second to fifth convolution layer and have used global average pooling layer before we flatten the output of fifth layer to feed to fully connected layer which reduced the unwanted spacial informations and reduced the number of parameters, by taking these steps our model refined and started to get good training time.We could have chance to used the batch normalization method, but we choose to experiment without batch normalization to compare the results with and without batch normalization.

The Convolution Neural Network model I initially implemented without normalizing the data. Even after tweaking many hyper-parameters like filters, filter size and stride there was no improvement in training accuracy. Both the training and test accuracy were around 10 percent. The model accuracy and loss plots which were plotted against the number of epochs are shown below. I have attached the initial model jupyter notebook along with report for reference.

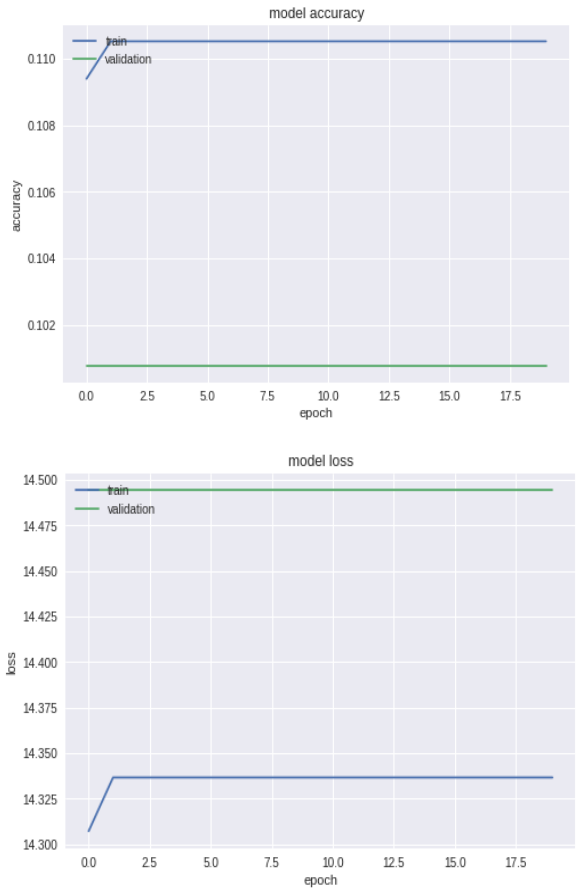
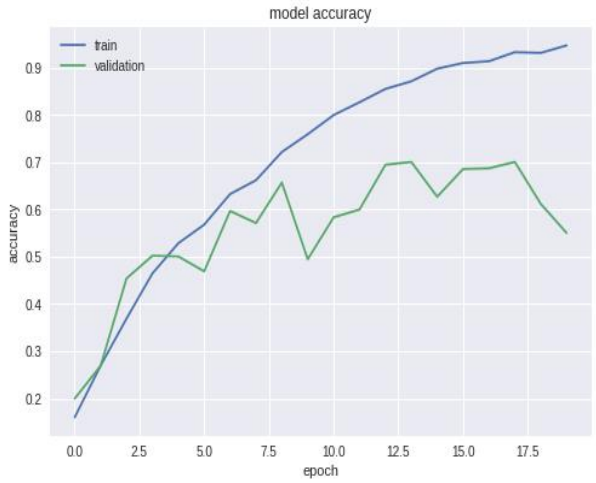
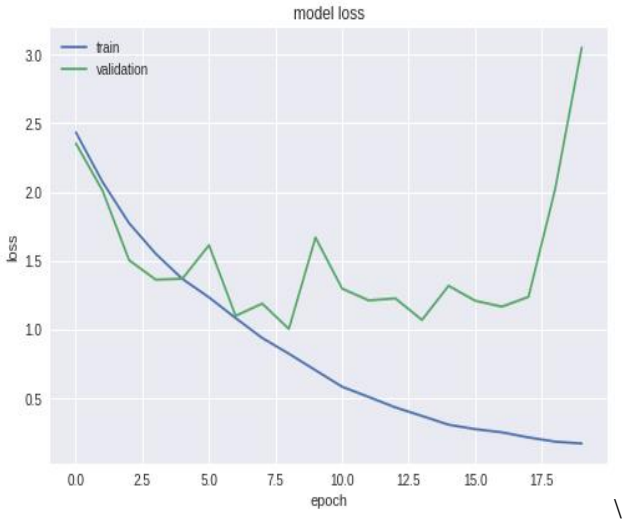


Fig 3.3.2: Model loss without Normalization

After I normalized the data there was a huge improvement in the model accuracy both on training and test datasets. The training accuracy was around 90 percent. But the model test accuracy was around 55 percent, which was the case of over-fitting the model. The training, validation accuracy and loss were clear signs of over-fitting the model. The model accuracy and loss plots which are plotted against the number of epochs are shown below.





\

In the above figure we can clearly see that validation accuracy decreases or remains constant while the training accuracy keeps on increasing around 90%. The validation loss keeps on increasing while the training loss decreases to zero. Both the characteristics of model indicates over-fitting of model. Thus I tweaked the parameters of convolution layers. I reduced the kernel size from 3x3 to 2x2 and setting the stride to 1. I also changed the number of filters in each layer and added max-pool and dropout layers to reduce over-fitting. Then the model performed well. The training accuracy was around 89% and testing accuracy was around 77%. I further used Batch Normalization layers to improve the model.

4.Results:

4.1 Model Evaluation and Validation :

To evaluate our model we take the 20% of the training set as validation set which will be used in each epoch to test the performance of the model. Even-though we have test set we cannot use it in the training, because of the chance of its effect on the generalization of the solution. Hence we use the validation set to measure the performance. We calculate the training sets loss and validation set loss after each epoch and records the results, when the validation loss starts to increase while training sets loss decreasing even more than previous epoch, we stop the training since it shows the model is over fitting for the training data.

After the each epoch the checkpoint has created and only the weights and biases of the where validation loss was very low has recoded and will be used to predict the fresh input images. Our model generalized well for the unseen data because we have trained and tuned the model with training sets and validation sets, our test set was not part of the training due to that there was no

influence of test set on the trained weights. When we used our test set with random images we are able receive a healthy output based on this we can conclude that the model is generalized for even unseen data and has acceptable robustness in it.

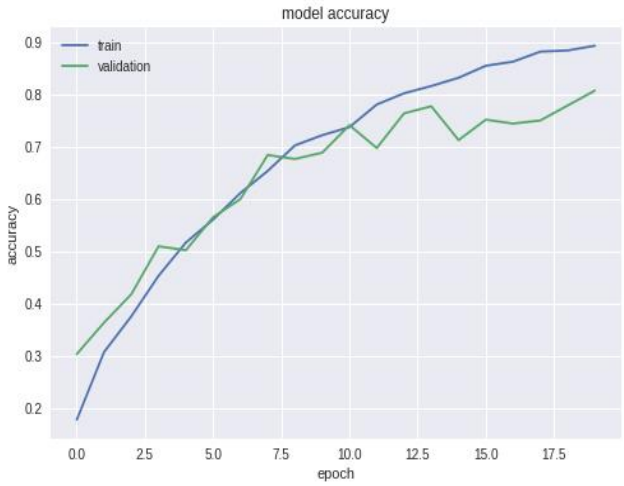
4.2 Justification:

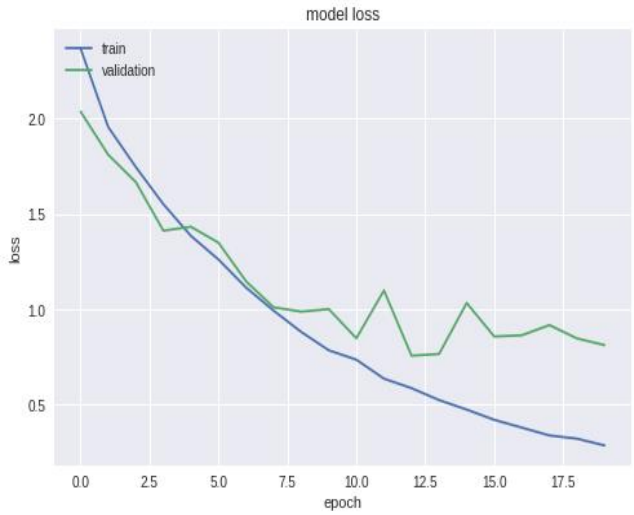
The weights which were saved during the training sets used on the testing sets and we predict the class profitabilities and the class with the higher probability has taken as the predicted label and used to calculate the accuracy score. Our model has done well in predicting the most test-sets classes with the accuracy score more than 80% which is much better than the Bench mark model which had only provided the accuracy of 77%. Hence we can assume that our model has done its job well to classify the plant seedlings.

5.Conclusion:

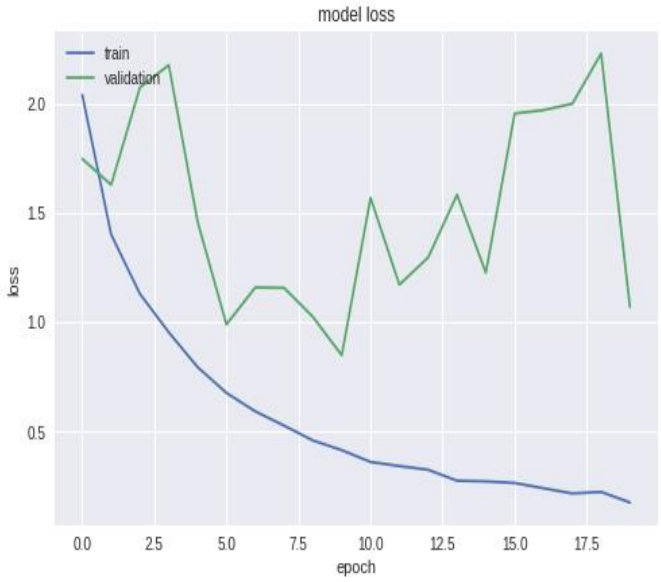
5.1 Free Form visualization:

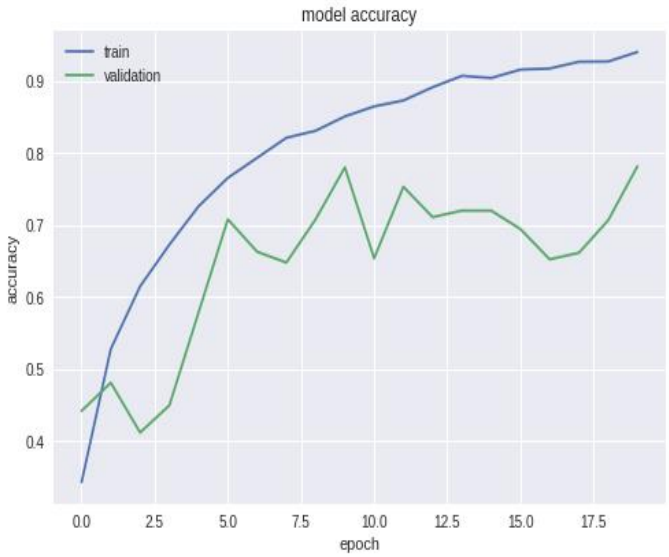
The accuracy and loss of the convolution neural network is calculated and plotted for each epoch as shown below.





The model accuracy is plotted against number of epochs in the first image. It has two accuracies. one is training accuracy and other one is validation accuracy.both the training accuracy and validation accuracy increases till specific number of epochs. the training accuracy keeps on increasing even after specific epochs whereas validation accuracy decreases after 11 epochs. This is where the over-fitting of the model happens. After specific epochs we can also clearly see the validation loss increases but the training loss keeps on decreasing. The model is over-fitting after specific epochs.





The CNN model has accuracy of 77-79% on test data, we can further increase the accuracy score by adding batch normalization to the CNN model.The accuracy and loss of the convolution neural network with batch normalization is calculated and plotted as shown below.

5.2 Reflection:

In the project, we developed a convolutional neural network model for the classification of species of plant seedlings. We were provided only with training dataset, we split the data into training and test datasets using train_test_split sklearn library. Then we converted the respective images to a numpy array. Then we transformed the data and applied normalization. The CNN models perform better in classification, hence the results we got are as expected. Even though achieved the desired result by just parameter tuning, adding conv and dense layers, we further used Batch normalization which improved the result. The final model met the expectations.

By this Capstone Project the most important thing I have learnt is implementing a Convolution Neural Network model from scratch. The most difficult problem in implementing this model was over-fitting. Dropout and max-pooling came to the rescue. Further implementing batch normalization reduced the over-fitting of the model. The Kaggle Competition site had huge datasets to choose from. Firstly, I used to think Kaggle was only for the Machine learning pros. Now I got enough confidence to participate in machine learning competitions.

5.3 Improvement:

We improved our CNN model further by introducing batch normalization. Batch normalization is a technique for improving the performance and stability of artificial neural networks. It is a technique to provide any layer in a neural network with inputs that are zero mean/unit variance. Similar to normalizing input data, where the values have a wide range are reduced to a smaller range usually between 0 and 1, the batch normalization is applied for hidden layers which speeds the training by reducing to a smaller range values. We could further improve our model using transfer learning such as VGG16, VGG19, ResNet50, Inception or Exception.