Machine Learning Engineer Nanodegree

Capstone Report: Plant Seedlings Classification Using Convolutional Neural Networks

Dorcas Balogun

August 21, 2018

Project Overview

This project is based on the detection of weeds using various pre-processing techniques in conjunction with convolutional neural network(CNN).

Research as shown that various systems have been developed for the detection of weeds over the years, but a true commercial breakthrough of these systems is still to come despite the construction of several prototypes and case studies showing promising results. Researchers in this domain have argued that the most promising approach for site-specific weed control is currently a ground-based computer vision system (Gerhards, 2010), since approaches such as remote sensing need to address a myriad of problems in order to be robust and reliable - many of which (such as solar angle and cloud cover) are beyond the control of the users (Thorp and Tian, 2004). CNNs are used to achieve state of the art result in the field of computer vision. This is the reason why CNNs was the technique used in this project.

The dataset used for this project was provided by the Aarhus University Department of Engineering Signal Processing Group in collaboration with University of Southern Denmark. The images within this dataset are based on ground-based weed or specie spotting, which makes it appropriate for this project.This dataset supports and encourages the development of species recognition techniques and plant appearance analysis for the agricultural industry. However, a subset of the original dataset was used for this project.

Problem Statement

Classification of weeds from crop seedling is quite challenging, even with the human eye and as a result, the growth of weeds have led to the production of chemical substances which are toxic to crop plants(allelopathy), animals, or humans. The effects of weeds on plants are numerous and this leads to reduced crop quality.
To solve this problem, a convolutional neural network model will be developed. A model will be generated from scratch based on series of convolutional layers and dense layers and this will

represent the benchmark model. This model will be compared with a model derived from transfer learning and image augmentation which should generate a more accurate result.
The resulting solution will be measured based on its mean FScore. Therefore, the resulting solution should have an F1 score close to 1.0, within the range of 0.9 to 1.0.

Metrics

The convolutional neural network model generated will be evaluated based on its Mean FScore, which is actually a micro-averaged F1-score. The F1 score is the harmonic mean of precision and recall.

F1 score is the metric proposed by the research team that generated the plant seedling dataset. Therefore, this metric will be used for this project.

Given positive/negative rates for each class *k*, the resulting F1 score is computed this way:

$$Precision_{micro} = \frac{\Sigma_{k \varepsilon C} TP_k}{\Sigma_{k \varepsilon C} TP_k + FP_K}$$

$$Recall_{micro} = \frac{\Sigma_{k \varepsilon C} TP_k}{\Sigma_{k \varepsilon C} TP_k + FN_K}$$

$$MeanFScore = F1_{micro} = \frac{2Precision_{micro} Recall_{micro}}{Precision_{micro} + Recall_{micro}}$$

- True Positive (TP) is an outcome where the model correctly predicts the positive class.
- True Negative (TN) is an outcome where the model correctly predicts the negative class.
- False Positive (FP) is an outcome where the model incorrectly predicts the positive class.
- False Negative (FN) is an outcome where the model incorrectly predicts the negative class.

Data Exploration

The plant seedling dataset consist of 4,750 plant images belonging to 12 species at several growth stages. It comprises annotated RGB images with a physical resolution of roughly 10 pixels per mm.

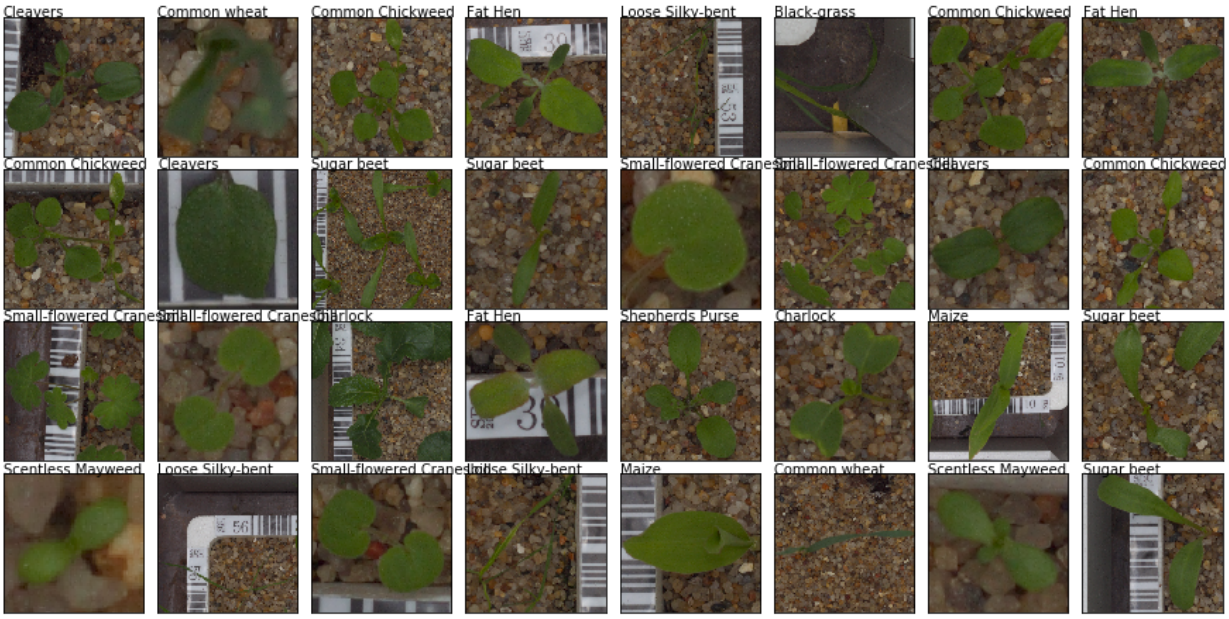

Fig 1: Images of different classes within the plant seedling dataset.

The dataset was divided into two categories namely; the train dataset which consist of 3,801 images and a validation dataset consisting of 949 images.

S/N	Class	Validation	Train	Total
1	Black-grass	53	210	263
2	Charlock	78	312	390
3	Cleavers	57	230	287
4	Common Chickweed	122	489	611
5	Common wheat	44	177	221
6	Fat Hen	95	380	475
7	Loose Silky-bent	131	523	654
8	Maize	44	177	221
9	Scentless Mayweed	103	413	516
10	Shepherds Purse	46	185	231

11	Small-flowered Cranesbill	99	397	496
12	Sugar beet	77	308	385

Table 1: Value count of each class within the validation and train dataset.

From the above table, it can be observed that the Loose Silky-bent class has the highest number of images and the Common wheat and Maize class has the least number of images. It can also be observed that the number of images to train and validate a model is quite small, therefore image augmentation techniques has to be employed to generate more images.

Exploratory Visualization

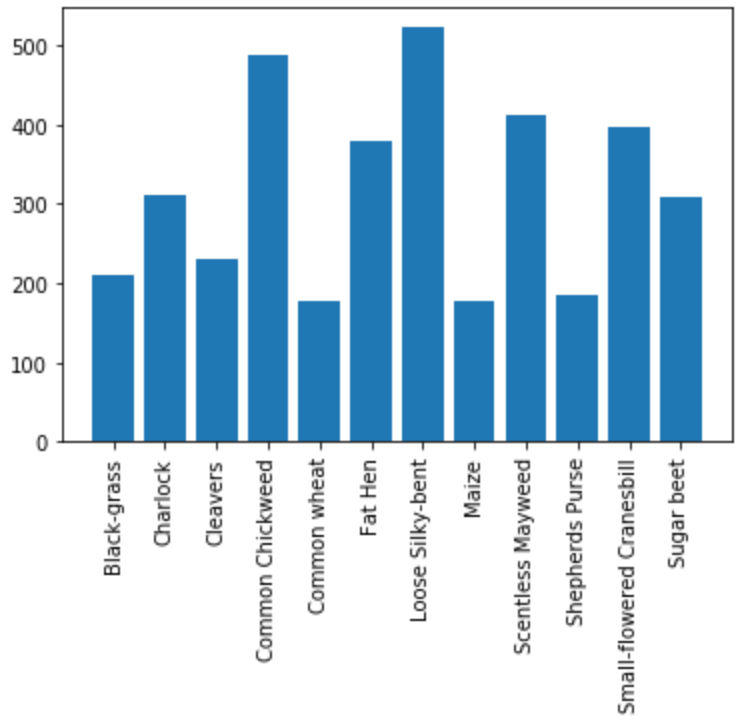

Fig 2: A Bar chart representation of each class in the Train dataset

Figure 2 gives a clear view of the variation present in each class, with Loose Silky-bent class having the highest number of images, and the Common wheat and Maize class having the least. This clearly shows that the dataset is not balanced.

Algorithms and Techniques

This project is based on an image classification task, which will involve the application of computer vision algorithms and techniques. A CNN architecture, transfer learning and image augmentation techniques will be used for the implementation of this project.

Convolutional Neural Networks are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNNs have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. This however, makes it a good fit for this image classification task.

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models. In order to effectively make use of the computing resources available for this project, transfer learning will be employed.

In dealing with image data, there are always some inconsistencies within the images (some are either too big or small, some are rectangular instead of square, etc). Another frequently faced problem is the number of images in the training set which often results in overfitting. Image augmentation techniques helps to deal with these issues. Image augmentation transforms the images in the training set so as to increase the ability of the model to recognize different versions of an image. This increases the breadth of information the model has. It now becomes better suited to recognize target objects in images of varied contrast, size, from changed angles and so on. Image augmentation will be used in this project because the dataset available is quite small, consisting of a total of 4,750 images.

Benchmark

Models will be evaluated based on micro-averaged F1-score which is the performance benchmark recommended in the research paper of this dataset. The benchmark model is a CNN model consisting of three convolutional layers and two dense layers with a total of 1,945,782 trainable parameters. A reLU activation function and a kernel size of three was used. An F1 score of 0.7808 was gotten, which represents the benchmark score. This serves as a benchmark towards building a better model.

Data Preprocessing

When using TensorFlow as backend, Keras CNNs require a 4D array (which we'll also refer to as a 4D tensor) as input, with shape

(nb_samples, rows, columns, channels),

where nb_samples corresponds to the total number of images (or samples), and rows, columns, and channels correspond to the number of rows, columns, and channels for each image, respectively.
A function was defined to take in a string-valued file path to color images as input and return a 4D tensor suitable for supplying to a Keras CNN. The function first loads the image and resizes it to a square image that is 299×299 pixels. Next, the image is converted to an array, which is then resized to a 4D tensor. In this case, since we are working with color images, each image has three channels. The returned 4D tensors will always have shape

(nb_samples,224,224,3).(nb_samples,224,224,3).

Where nb_samples is the number of samples, or number of images, in the supplied array of image paths.

No pre-processing was done concerning the fact that the dataset is not balanced. This abnormally however, was corrected during implementation.

Implementation

All models were trained on the train dataset and they were evaluated on the validation dataset. Two metric were used to evaluate the models, namely; F1 score and accuracy metrics. All models were trained over 16 epochs.

Two transfer learning architectures trained over the imagenet dataset were implemented namely; Xception and InceptionResNetV2 models. A function was designed to take in the following parameters as input:

- The optimizer to use while compiling the model.
- An instance of any of the above mentioned architectures with the top dense classification layer consisting of 1000 nodes removed.
- An optional index that specifies the number of layers to unfreeze and train for the instance of the architecture inputted.

Within this function, a new model was defined that takes the input of the architecture specified as its own input and outputs a dense layer of 12 nodes, each representing the classes in the dataset. This new model was compiled over a softmax activation function and a categorical_crossentropy loss function. This new model is always called before training and a batch size of 32 or 64 is always passed in as arguments while fitting the train dataset to the model. The Adam and RMSprop optimizers were the only optimizers used during training.

While fitting the train dataset to the model, the weights of classes were slightly modified by passing a dictionary mapping class indices (integers) to a weight (float) value, used for weighting the loss function (during training only). This allows the model to "pay more attention" to samples

from an under-represented class. This however solves the problem of the imbalanced dataset.

Image augmentation was also implemented to increase the number of images trained. Each image was modified based on the following parameters:

- Horizontal flip and Vertical flip were set to True.
- Width shift range and an Height shift range were both set to 0.1 respectively.
- Rotation range of 30 degrees.

The ZCA whitening was part of the proposed parameters to be used, but it wasn't implemented as a result of low memory space. A seed value of zero(0) was passed to the image generator flow method. The makes sure that the same set of images are generated every time a different model is applied to the dataset.

During training, a batch size of 32 had 119 steps per epoch and a batch size of 64 had 59 steps per epoch. The number of steps per epoch was calculated using this formula:

$$Number\ of\ steps\ per\ epoch\ = \frac{Total\ number\ of\ images\ in\ the\ train\ dataset}{Batch\ size}$$

Refinement

The aim of this project is to derive a model that performs better than the benchmark model and attains an f1 score within the range of 0.9 and 1.0 on the validation dataset. The CNN model designed for the benchmark model was trained from scratch. In order to derive a better model transfer learning solutions were implemented. The function explained in the Implementation section of this report will be referred to as the base_model because it will be referenced a lot in this section. It should be noted that all F1 score explained in this section represents the F1 score derived from the validation dataset and not on the train dataset and all models are trained only on the train dataset.

The initial solution made use of the InceptionResNetV2 model which was passed as an instance to the base_model. The index parameter of the base_model was set to None and a batch size of 32 and an RMSprop optimizer was used. This yielded an F1 score of 0.8019. The optimizer was changed to the Adam optimizer and this gave a lower F1 score of 0.7924. These F1 scores are quite low and clearly shows that setting index to None which freezes all the layers in the InceptionResNetV2 model does not work well for this dataset.

The above model was modified to have a batch size of 32, an RMSprop optimizer and an index of 3, which means the top 3 layers of the InceptionResNetV2 model instance passed to the base_model will be trained and the remaining layers will be frozen. Image augmentation was also applied to this model. This yielded an F1 score of 0.8809, which performs better than the benchmark score. The batch size of this model was changed from 32 to 64 and a better F1 score of 0.8904 was gotten. The optimizer of the above model was changed from RMSprop to an

Adam optimizer. When a batch size of 32 and 64 was used, an F1 score of 0.8651 and 0.8757 was gotten respectively. These F1 scores are lower when compared with the F1 score derived when an RMSprop optimizer was used. From this observation, it can be concluded that an RMSprop optimizer achieves a better result for this dataset and as a result of this, it will be used for the remaining models that will be generated.

The index value of the above model was increased to 5 with a batch size of 64, but a lower F1 score of 0.8493 was derived.

The highest F1 score gotten from the InceptionResNetV2 model was 0.8904 and this is lower than the F1 score set to be attained in this project, which should be within the range of 0.9 to 1.0. Therefore, a different transfer learning architecture known as Xception model was implemented. An instance of the xception model was passed as a parameter to the base_model. The index parameter of the base_model was first set to 3, with a batch size of 64 and an RMSprop optimizer. This yielded an F1 score of 0.8788. The model was modified to use a batch size of 32 but a lower F1 score of 0.8641 was derived.

It was observed that a batch size of 64 performs better than a batch size of 32, therefore a batch size of 64 will be used for the remaining modifications of the model. The index parameter of the base_model was changed from 3 to 6 and an F1 score of 0.9020 was gotten. Increasing the index value from 6 to 8 achieves a better accuracy of 0.9168. This value falls within the range of F1 score set to be achieved for this project hence, it will represent the final model.

Model Evaluation and Validation

The final model was based on a transfer learning architecture known as Xception, trained on the imagenet dataset. A different transfer learning architecture known as the InceptionResNetV2 model was also tested, but the F1 score derived was less than the F1 score range set for this project. The top eight(8) layers of the xception model was unfreezed and trained, while the remaining layers remained freezed. This achieved a better result when compared with an Xception model whose top 6 layers were unfreezed and trained.

An RMSprop optimizer was used instead of an Adam optimizer because it achieved a better result. Image augmentation techniques were used to improve the performance of the model due to the fact that the dataset for this project is quite small. A batch size of 64 was used, which proved to have a better performance when compared with a batch of 32. While fitting the train dataset to the model, the weights of classes were slightly modified by passing a dictionary mapping class indices (integers) to a weight (float) value, used for weighting the loss function (during training only). This allows the model to "pay more attention" to samples from an under-represented class due to the fact that the dataset used for this project was not balanced.

This model was trained over the train dataset and evaluated on the validation dataset. The model

produced an F1 score and an accuracy of 0.9168 and 91.6754% respectively. The F1 score of the valuation dataset gotten from this dataset falls within the range of F1 score set to be achieved for this project. Hence, it was chosen as the final model for this project.

Justification

S/N	Class	Benchmark model	Final model
1	Black-grass	29	80
2	Charlock	76	78
3	Cleavers	63	54
4	Common Chickweed	143	117
5	Common wheat	42	44
6	Fat Hen	89	97
7	Loose Silky-bent	162	103
8	Maize	38	47
9	Scentless Mayweed	99	102
10	Shepherds Purse	64	53
11	Small-flowered Cranesbill	75	97
12	Sugar beet	69	77

Table 2: Value count of each class predicted by the benchmark and final models on the validation dataset

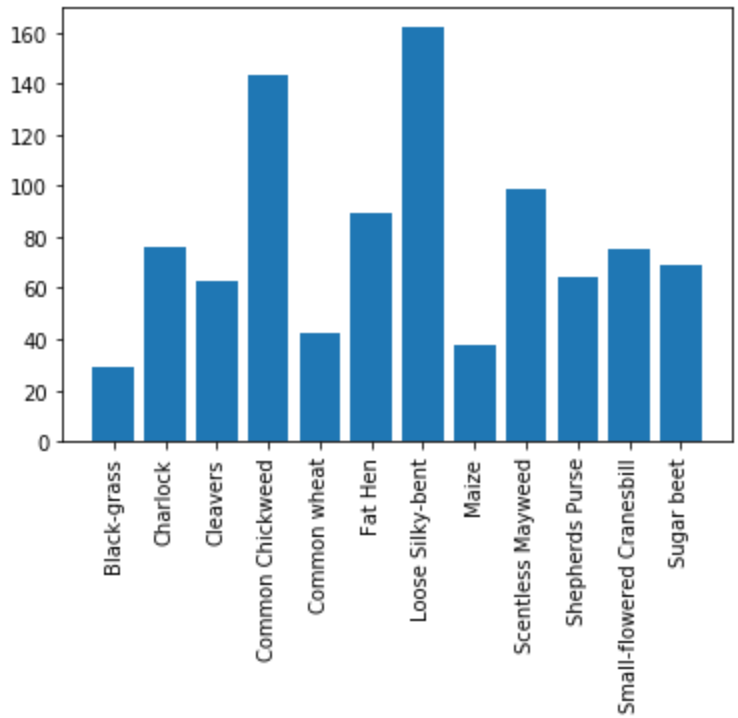

Fig 3: A Bar chart representation of the predicted validation values by the benchmark model

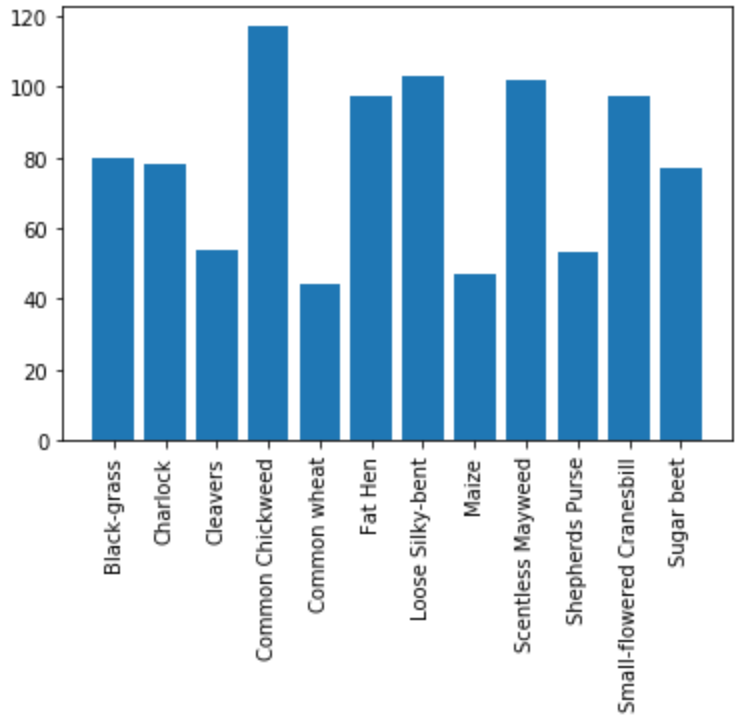

Fig 4: A Bar chart representation of the predicted validation values by the final model

From the above table, the difference between the maximum and minimum value count of the benchmark and final model is 133 and 73 respectively. This clearly shows that the benchmark model paid more attention to some classes than others. The variance between the classes predicted by the final model, is not as high as that of the benchmark model. This means that the final model paid attention to all the classes rather than some set of classes, despite having an imbalanced dataset. In addition, the final model has an F1 score greater than that of the benchmark model, that is within the range of 0.9 to 1.0. This therefore makes the final model the best model for solving this problem

Free-Form Visualization

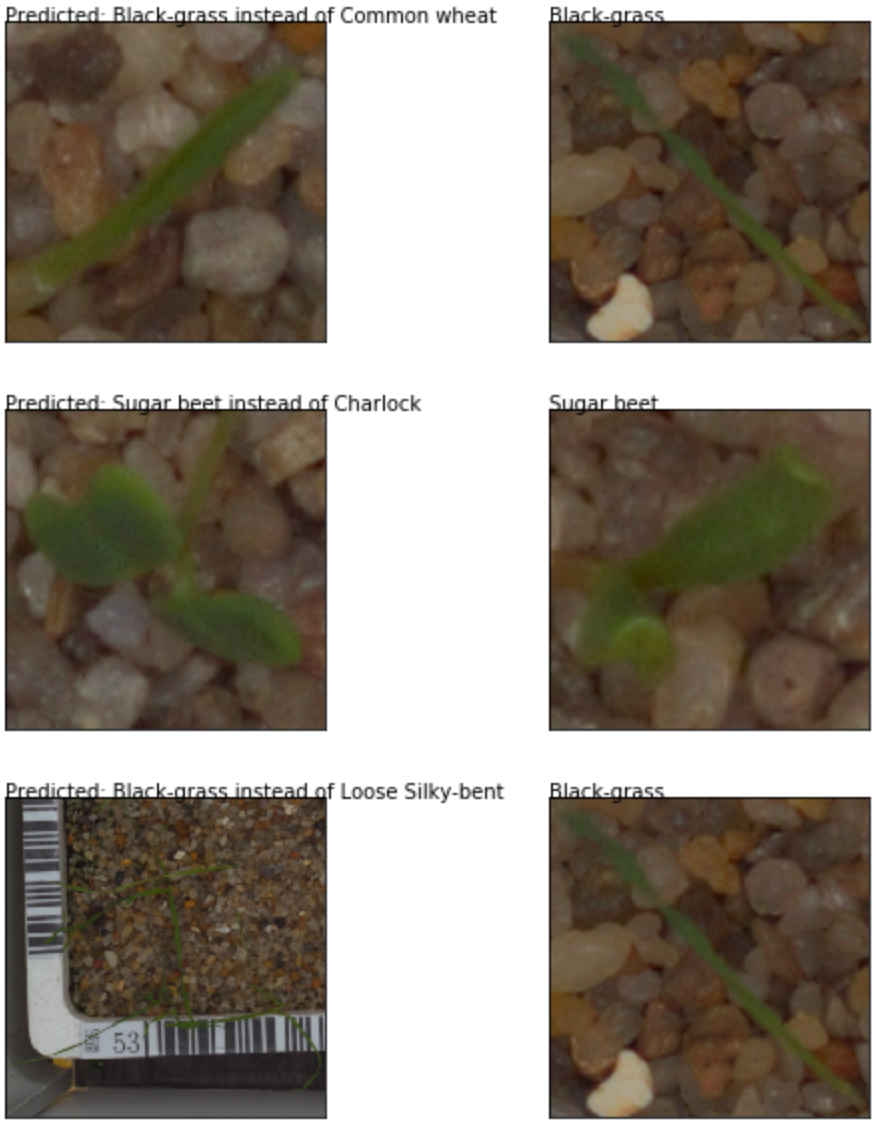

Fig 3: Images predicted incorrectly by the final model.

From the above figure, it can be clearly seen that the images among some classes in the dataset are almost the same. For the first pair of images, a black grass was predicted instead of a common wheat. Even with the human eye, the difference among these images is hard to tell. This applies to the second and third pair of images respectively.

Reflection

This project is based on the classification of plant images belonging to 12 species at several growth stages. This task was quite challenging due to the fact that most plant images look similar. CNNs were considered for this task as it has proven to be successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. A CNN model was designed from scratch to represent the benchmark model for this project. Techniques and algorithms such as transferring learning and image augmentation were implemented to create a model that performs better than the benchmark model. As a result of this, various models with different activation functions and batch sizes were created. Each model was evaluated based on it's F1 score on the validation dataset. The best model had an F1 score that falls within the range of F1 score set to be achieved within this project. Each model was trained for up to 20 to 30 minutes. As a result of the computational time taken to train each model, some transfer learning models such as InceptionV2, ResNet and VGG models could not be tested. The fact that just two transfer learning models were experimented on was quite limiting.

Improvement

A whitening transform of an image is a linear algebra operation that reduces the redundancy in the matrix of pixel images. Less redundancy in the image is intended to better highlight the structures and features in the image to the learning algorithm. Image whitening is performed using the ZCA technique. To implement this technique, the zca_whitening argument is set to True while instantiating the image augmentation class. This technique could not be implemented in this project as a result of low memory space. This technique improves the results of learning algorithms and as a result it would have improved the performance of the final model.

References

1. Thomas Mosgaard Giselsson, Rasmus Nyholm Jørgensen , Peter Kryger Jensen, Mads Dyrmann and Henrik Skov Midtiby (2017, November 16). A Public Image Database for Benchmark of Plant Seedling Classification Algorithms.
2. https://www.kaggle.com/c/plant-seedlings-classification