Using Image Classification to label plant species, Project 2 for Big Data & Analysis

Henry Stuklis (a1706223)

October 22, 2019

Abstract

In this report I go through my working and understanding of an image classification problem posted on Kaggle. I explain what a Kaggle competition is and how different submission are ranked against each other publicly on the site. I introduce the problem which involves classifying plant species based on physical images of the plant. I go through and explain what a convolutional neural network is and how it will be used in this problem. I go through my methods of pre-processing the data such that the neural network extracts the wanted features from the image. I also introduce a very important tool that is used in tandem with the neural network, the image data generator. I then give a brief run down of my attached code and discuss the results from my experiment. The neural network is found to have a testing accuracy of 0.9558 and a training accuracy of 0.9416.

1 Introduction

This project for Big Data & Analysis required us to complete one of four Kaggle competitions. Kaggle is a website that lists interesting data sets along with problems to solve using that data. Mainly these problems involve building a statistical model to perform prediction. On the Kaggle website you are allowed to submit your statistical models predictions and see how much error it has in predicting the true known values. Kaggle then displays a leaderboard of all submitted models and the main aim of competing in these Kaggle competitions is to try and rank as high as possible. The less error present in a model the better and more robust the model is in prediction. So the workflow pipeline for a Kaggle competition is similar to as follows.

- Clean and pre-process the base data (given as training and testing data)
- Complete some feature engineering (transforming, dropping or creating variables)
- Pick a reasonable statistical model to predict the required values
- Train the statistical model on the training data
- Analyse the error in predicting the testing data (either by submitting on Kaggle or calculating the error yourself)

My detailed and commented Python code that goes through this entire problem pipeline can be found attached separately to this report in plant-species-analysis-code.ipynb.

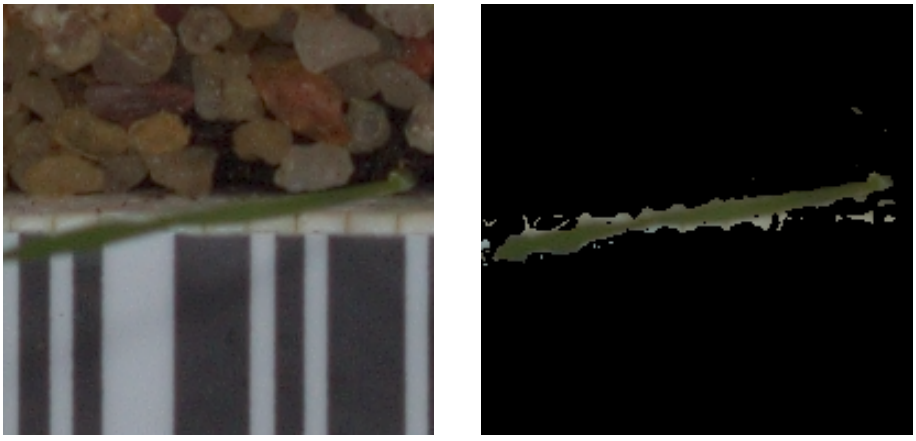
The Kaggle competition that I chose out of the four is a multilevel image classification problem. The given data contains 4750 pictures (in `.png` format) of twelve different plant species. These species are

The aim of the predictive model to be constructed is to correctly classify each plant species just based on the provided image. To do so we will need to split the images provided under `train.zip` on Kaggle into a separate training and testing data. As we have not been given a percentage split in the problem statement and we have a very large amount of images, I have elected to make the training to testing split 80% to 20%. So our predictive model will need to be able to learn the differences between these twelve plant species based on the provided visuals in the training data.

The type of predictive model I will be using for this problem is known as a convolutional neural network. This is a very commonly used neural network in image problems as it is able to progressively simplify each image down to a certain image and then compare with each canonical image (one for each plant species) the trained model has. It then returns the species label of the canonical image each image matches the most.

2 Methodology

Before I discuss the type of predictive model that will be used in this classification problem there is another crucial component to this problem, pre-processing. First on the pre-processing agenda is the question of independence between samples. The data we have been given is a set of plant images of plants located on a university campus taken from a top down perspective. So we can assume that all of the twelve plant species are located in similar garden beds. Even more troubling is that each species is probably located directly next to each other in the images. So the neural network to be constructed could potentially not learn the key features of the plant species but instead learn the key features of the background rocky garden beds. One example of this non-independence showing through is with images in the black-grass plant species. We can see just by some brief observation that many of these black-grass images have a characteristic "barcode" edge in the background. So potentially the neural network to be constructed could identify this as a key feature of the black-grass class and use this to classify images as black-grass. This is exactly what we want our neural network to not do. As we want the network to be able to get the key features of the plant species and nothing else. So the way in which I dealt with this problem was to simply discard the background. Thankfully, all plant species photographed were distinctly green against a non-green rocky background. So all that was needed to be done was to apply a green-mask over each image. Additionally, this was done as the very first step of pre-processing so that the higher resolution images could retain some of there distinct qualities. Here is an example of the green-mask being applied to a black-grass image with the "barcode" feature visible in the background.



(a) Original image (b) Green-masked image

Figure 1: Green-mask applied to a black-grass image

After this green-mask has been applied to all images we then require that all images have square dimension and be of the same size for use in a neural network. After some analysis of the provided images (see attached code) I found that the smallest given image is 49-by-49 pixels. So as shrinking an image down to a smaller size is a much easier task that zooming in images to a larger size I opted to transform every given image to size 48-by-48 pixels. Fter doing this I also opted to apply a slight amount of gaussian blur to each image, as some images still looked sharper than others. So the gaussian blur helps normalise such that most images look to be of similar quality.

As previously mentioned the main tool that will be employed in this working is the convolutional neural network. A neural network is a type of self learning predictive model that mimics how neurons interact in the human brain. The neural network starts with the base input (an image in this case) which is fed through different layers of the network with some end goal in mind (in this case identifying the plant species out of the 12 possible species). Each layer is not restricted to performing a certain action and as such there is a large amount of freedom in constructing a neural network. Different types of layers can be mixed and matched however we want. However, in this case as I am still quite new to the concept of neural networks, in this problem I have opted to use a very basic structure. I use four convolution layers which are the exact same one after the other, with the only difference between the layers being the number of output filters in each layer. Such that as we go through the layers in the neural network the filters get larger in size. Then the final layer flattens the resultant image and compares to the canonical images. So the basic structure of each of the four layers is as follows

- Two two-dimensional 3-by-3 convolution layers that the effect of filtering the image down further to the canonical image
- A dropout layer that helps prevent overfitting by switching off 10% of the neurons at each layer
- A batch normalisation layer which normalises the convoluted images at each layer
- A max pooling layer that does similar to the convolution layers above but it takes the maximum pixel value on a 3-by-3 grid

After four layers of the above with filter sizes 128, 256, 512 and 1024 then we have the final layer which is responsible for classifying the convoluted image. This final layer has a flattening layer and two dense layers which result in a classification being made out of the set of possible species.

Additionally, I was able to employ the use of another very helpful tool in this neural network, known as an image data generator. This data generator acts as a way of viewing the image from a different perspective. As we can

agree that an image of a plant taken from above compared to an image of the same plant from a slightly different perspective (diagonally, mirrored or shifted slightly to the left or right ...etc), is still an image of the same plant species. As we are always prone to overfitting the model to the training data in these types of problems, the image data generator is a great way to stop this from occurring. The image data generator takes effect on each training epoch (training loop) over the set aside training data. It randomly alters each image in the training set by a small amount such that the neural network doesn't just learn from the exact training data every single time but from slight variations of the training data. Some of these alterations are: zooming in the image, rotating the image, shifting the image to the left or right and flipping the image. This image data generator has the effect of essentially creating a new set of data (within the bounds of our already gathered data) on each training loop. Which is almost comparable to having gathered a larger amount of data. This is extremely helpful in focusing the model on learning the key features of each of the twelve plant species as well as making sure the model does not overfit the training data.

3 Experiment

First, I setup my PC such that I would be able to use my GPU to accelerate the to be created convolutional neural network. This is crucial as the amount of computational power that we can throw into constructing the neural network will greatly effect the final accuracy of the model. I then checked the labels of each of the species and created directories to store my to be pre-processed images. I checked how many images we had been given and also how many images were in each species. I then went through and read the image sizes such that I could find the minimum images dimensions to shrink all images down to. After finding this, I went through, pre-processed and saved each image according to the methodology described previously. After doing this I sequentially setup the neural network as discussed above and then split the data into 80% training data and 20% testing data. I then reformatted the pixel data of the preprocessed images and dummy variable encoded the classifications of the training data such that it could be used in the neural network. I then setup the image data generator with the ability to rotate the image, shift the image up, down left and right, shear the image, zoom the image in and out, horizontally flip and vertically flip the images on each training epoch. Finally, I let the neural network train on the training data and score its accuracy on the testing data for each training epoch. I was able to have a batch size (number of images processed at once) of 32 with 20 sub-epochs in each of the 30 epochs for a resultant total of 600 total training loops through the image data generator.

4 Discussion & Conclusion

The final epoch of training was able to score an accuracy of 0.9558 on the testing data and 0.9416 on the training data. This shows that the convolutional neural network was able to find many of the identifying factors that separate the twelve plant species. The neural network trained for roughly 2 hours as I found that training it for any longer resulted in a good deal of overfitting, reducing the testing accuracy. With regards to the Kaggle public leaderboard this testing error gives me a position of roughly 400 out of the 800 submissions. This shows a roughly average score when compared with all the other Kaggle submissions but this is not too bad considering that the neural network only had access to 80% of the training data. So I believe this predictive model to be a little bit better on average than the other Kaggle predictive models.

There is still room for a lot of improvement in my model however. The main place for improvement would have to be the actual structure of the neural network. As I have only just been introduced to neural networks I still have a very basic understanding of what types of neural network structures to use in which problems. Perhaps there is a different setup which could greatly improve the models accuracy whilst not falling into overfitting like mine did. Perhaps you could also improve the computation time and maybe even not need the use of a GPU with the right type of neural network structure.

Another issue with my predictive model is that of the pre-processing steps I used on the training and testing data. The green-mask I applied would not be reasonable to use if we wanted to create a predictive model that can be used on plants in general. As we cannot always have that the background of the image is non-green. So if we had a lot more data to work with we could potentially not have to apply the green-mask.

So I believe the neural network was successful in predicting the species based on the given images to a very high degree of accuracy. There is still room for improvement but it does a great job using a very basic workflow.