Machine Learning Engineer Nanodegree

Capstone Proposal

Muthudivya Navaneetha Kannan
April 9, 2018

Proposal

Domain Background

One of major challenges in agriculture is weed management. Sustainable crop production is the need of the hour as we are struggling to supply food for the increasing global population. For effective crop production, weed management is one of the key factors. The ability to automatically classify and remove weeds among the crop at the seedling level will yield huge benefit and higher harvest. This problem is not something that can be totally eradicated, but a solution to this problem can help solve and mitigate weed problem now and, in the years, to come.

My personal interest and motivation for this problem is from my home garden. Both my parents love gardening, but unfortunately my father has challenges finding differences between the desired species and weed seedling unlike my mother who has a keen eye and clarity on the subject. It is sometimes fun to see my father struggle and seek help, but that made me realize there must be numerous 'actual' farmers who might also face this problem. They might have the right motivation and interest to become a potential farmer, but sometimes lack of knowledge like these may affect their harvest and in turn result in damage to their crop and property.

The dataset used for this problem has been donated by The Aarhus University Signal Processing group, in collaboration with University of Southern Denmark, in the hope "*to provide researchers a foundation for training weed recognition algorithms*" [3]. A benchmark for standardizing the results of this classification problem has been discussed in arXiv:1711.05458 using the original dataset.

Problem Statement

Problem is the similarity among different species of plant seedling. In this project, the model should be able to determine the species of a seedling from an image. The dataset

contains images of twelve categories of plant seedling, and hence the solution should be able to classify an image into one of twelve categories.

Datasets and Inputs

Datasets are obtained from Kaggle [4]. A training set and a test set containing them images of plant seedlings at various stages of grown. Each image has a filename that is its unique ID. The dataset comprises of 12 plant species, which are listed below:

Black-grass
Charlock
Cleavers
Common Chickweed
Common wheat
Fat Hen
Loose Silky-bent
Maize
Scentless Mayweed
Shepherds Purse
Small-flowered Cranesbill
Sugar beet

The objective is to build an image classifier that can accurately identify the image and classify the image into one of the above 12 species.
A quick look at the train dataset tells that the number of images available for each class (['Sugar beet', 'Small-flowered Cranesbill', 'Maize', 'Common wheat', 'Common Chickwee d', 'Fat Hen', 'Black-grass', 'Cleavers', 'Scentless Mayweed', 'Charlock', 'Loose Silky-bent', ' Shepherds Purse']) is varied (496, 221, 221, 611, 475, 263, 287, 516, 390, 654, 231), i.e. th e dataset is highly unbalanced. To combat with the unbalanced dataset, one of the man y strategies can be applied. To begin with I will apply under-sampling and observe the in itial results. Moving forward, I will augment the data for under-represented classes and t ry to balance the dataset. However, the problem of unbalanced dataset can also be dealt by calculating the confusion matrix and F1 score for the classifier, as these metrics give u s the more appropriate evaluation of the model. The images are RBG colored and have a background that is undesirable for the problem at hand can be gotten rid of by masking – this can be achieved by converting the RGB images to HSV mode by tuning the related parameters. I intend to use 80:20 as the train to validation split ratio for training my class ifier.

Solution Statement

The solution of this problem is to build and train a model that can classify the new unseen image into one of the twelve mentioned categories accurately.

This can be achieved by:

- For any given image, the model outputs one predicted value (i.e., the species of the seedling with highest probability)
- However, the model outputs probability of every class that the image can belong to, we can design the output to be in such a way that the only category with high probability is shown as prediction for the input image.

Benchmark Model

I plan to build a simple vanilla CNN model, that I can train and test on this dataset. I will refer this model as my benchmark model to compare its results to that of my final classifier. Once I'm able to make clear comparisons between the two models, I intend to refer the submissions from the Kaggle competition to understand where my model stands among other competitor's submissions. In particular, I'm interested in two hosted Kernels with scores 0.9006 and 0.971. I have not yet seen their code as I am yet to build and train my model. I may run these kernels once results for my classifier are ready, to learn and apply tricks that I may find useful to improve performance of my model (I will cite them, if I do so).

Evaluation Metrics

Since this problem is taken from Kaggle [5], I'm going to use their evaluation metric for measuring performance of my classifier – by calculating the Mean F Score.

The F1 score [1] can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relate contribution of precision and recall to the F1 score are equal. The formula for the F1 score is given as:

F1 = 2*(precision * recall) / (precision + recall)

Precision and recall can be calculated as [2]:

Precision = number of TP / (number of TP + number of FP)

Recall = number of TP / (number of TP + number of FN)

Where, TP is True Positives, FP is False Positives and FN is False Negatives.

As this problem statement has multi-class solution, Mean F1 score is the weighted average of the F1 scores of each class.

Project Design

First, I would do some exploratory data analysis to get an idea on the training dataset, to check if the sample available for each category is equal (i.e, if the dataset if balanced), check aspect ratio of images is uniform across all images, evaluate if images need some pre-processing, normalization etc. Once EDA and required cleaning/processing of images are attained, I will move forward in designing the architecture of my neural network.

My intention is to develop a Convolutional Neural Network on top of a pre-trained network such as ResNet50, Inception and Xception rather than designing a network from scratch. I will apply transfer learning for this problem as it is much faster and is more efficient. After extracting the bottleneck features of these pre-trained models, I plan to add few layers on top of them and run some quick comparative tests on these 3 models. Once I decide on which pre-trained model to move forward with (on basis of their performance), I will follow the below approach.

Since I'm using transfer learning to build my network, I will be applying knowledge that I gained about how to efficiently apply transfer learning by comparing the size and similarity of the pre-trained network's data and the dataset in hand. Following are four cases, one of which can be applied for current problem.

Case 1: Small Data Set, Similar Data
If the new data set is small and similar to the original training data:

- slice off the end of the neural network
- add a new fully connected layer that matches the number of classes in the new data set
- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network
- train the network to update the weights of the new fully connected layer

Case 2: Small Data Set, Different Data
If the new data set is small and different from the original training data:

- slice off most of the pre-trained layers near the beginning of the network
- add to the remaining pre-trained layers a new fully connected layer that matches the number of classes in the new data set

- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network
- train the network to update the weights of the new fully connected layer

Case 3: Large Data Set, Similar Data
If the new data set is large and similar to the original training data:
- remove the last fully connected layer and replace with a layer matching the number of classes in the new data set
- randomly initialize the weights in the new fully connected layer
- initialize the rest of the weights using the pre-trained weights
- re-train the entire neural network

Case 4: Large Data Set, Different Data
If the new data set is large and different from the original training data:
- remove the last fully connected layer and replace with a layer matching the number of classes in the new data set
- retrain the network from scratch with randomly initialized weights
- alternatively, could use the same strategy as the "large and similar" data case

As the images in the current dataset is very different from that of the images used to train ResNet50, Inception or Xception and the size of the plant seedling dataset is relatively smaller, Case 2 is applicable.

Hence, as mentioned in Case 2, I will be removing most of the pre-trained network's layer and retain the beginning of the network that detects the basic features of an image. Add fully connected layer with 12 nodes with a softmax function, randomly initialize the weights of the newly formed network and freeze all the weights from the pre-trained network. Finally, will retrain the whole network to update weights of the new fully connected network.

Once I successfully train the network, and observe validation loss, I may have to fine-tuning the architecture by modifying the number of epochs, optimizer, add more dense layers, dropout layers, etc., retrain the network until I get desired accuracy on validation set and low loss value. I may also try data augmentation to further improve the performance if need be.

References

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[2] https://en.wikipedia.org/wiki/Precision_and_recall

[3] https://vision.eng.au.dk/plant-seedlings-dataset/

[4] https://www.kaggle.com/c/plant-seedlings-classification/data

[5] https://www.kaggle.com/c/plant-seedlings-classification#evaluation