Identifying Fungal Diseases in Growing Wheat using Deep Convolutional Neural Networks.

Joseph Okonda L.
Department of Computer Science
Stanford University
jlikhuva@stanford.edu

Abstract

In this paper we present a method for identifying diseases in growing wheat*(Triticum spp.)* plants. We use labelled images of wheat plants to train a convolutional neural network image classifier. We begin by exploring the binary task of identifying whether a plant is healthy or not. We then present a classifier trained to identify four common fungal diseases of wheat.

1 Motivation

Wheat is the second most consumed crop in the world. It is an important source of livelihood to both large-scale commercial farmers in highly industrialized nations and small scale producers in low income countries. The crop is susceptible to a variety of diseases which can lead to significant yield losses and thus loss of income[2]. Currently, farmers deal with diseases by either planting cultivars resistant to specific diseases or by using fungicides. The former, while effective, may be prohibitively expensive and thus not readily available to all farmers. Moreover, it is not a long term solution because, often, mutations of disease causing organisms give rise to new disease strains that affect the cultivars. The latter, on the other hand, requires farmers to know when to apply the fungicide. This usually requires farmers to periodically inspect their fields to ensure that they catch any outbreak early and contain it a tedious process that does not scale. A system that inspects fields (using UAVs, for instance) and automatically identifies any diseases, can be used to streamline large scale wheat production. Such a system would need a way to identify the various diseases.

2 Data.

We use Google Image Search to collect samples for training. The data are automatically scraped from the search results using an open source tool[1]. The images are then manually inspected to eliminate images that do not contain wheat plants. The table below gives a summary of the data.

2.1 Data Pre-processing.

Before training, we re-size all images to be 256x256.During Training, we augment the data using the following class preserving transformations: random horizontal flips, random brightness adjustment and random saturation adjustments. We also experiment with normalizing the data using Image-Net statistics, but this leads to a drop in performance, so we abandon this strategy. During both training and evaluation, the pixel values are converted to float values between 0 and 1.

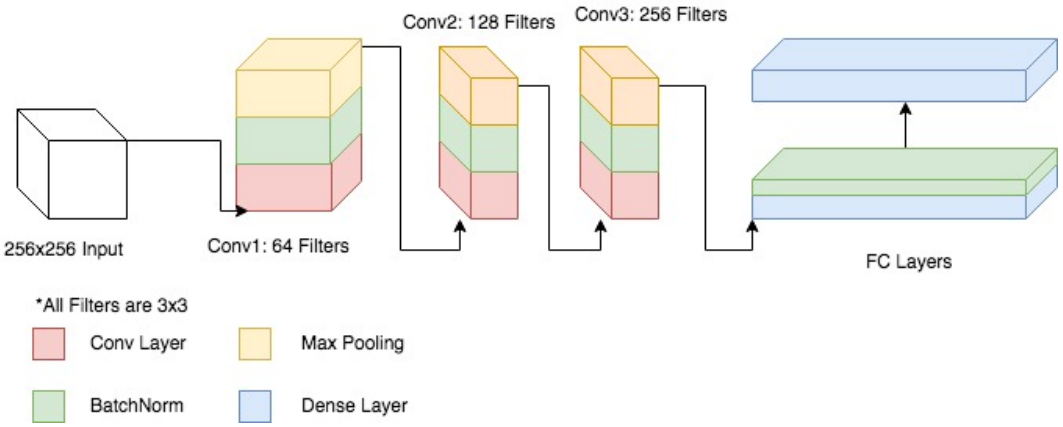
[1] https://github.com/hardikvasa/google-images-download

Disease	# of Samples
Healthy	838
Fusarium Head Blight (FHB)	526
Rust (Leaf and Streak)	515
Septoria Triciti Blotch (STB)	557
Powdery Mildew (BG)	296

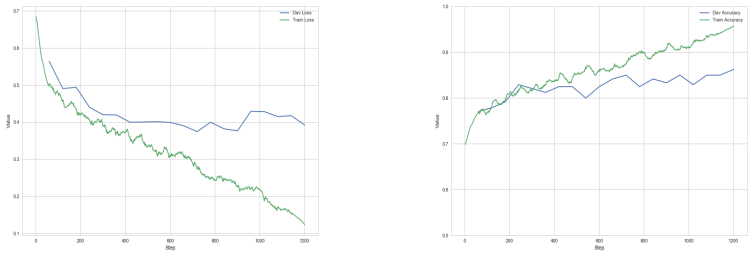
(a) Distribution of Samples

Figure 1: The Distribution of samples and Examples from each class.

3 Binary Classification



We first create a binary CNN classifier.Each convolutional layer is made up of a convolution operation between the input and all Kernels of that layer, a Batch Normalization step and a Max pooling layer. All Kernels are 3x3 applied with a stride of 1 and 'SAME' padding. The pooling layers use 2x2 windows and a stride of 2. In addition to the convolutional layers, the network has 2 fully connected layers. Dropout is applied to both the convolutional and dense layers. The network is trained to minimize the Binary Cross Entropy loss using the Adam optimization algorithm[1] with a learning rate of 0.001. The 64 Filters in the first layer are initialized using weights from the VGG-16 network trained on the ImageNet dataset. These Kernels are not modified during backpropagation. The other Filters and Weights are randomly initialized using Xavier Initialization. The graphs below show the Loss and Accuracy curves.



(a) Loss. (b) Accuracy

Figure 2: The Accuracy and Loss on both the Train and Dev. Set.

4 Further Work.

As the loss curves suggest, we are still over-fitting on the training set. Therefore, the next step will be to test more regularized models. Also, we'll explore holding initializing more Kernels from VGG-16 and holding them fixed during back-propagation. After this, well transition to building a multi-class classification model. We will also analyze the models qualitatively using occlusion sensitivity testing.[3]

5 Appendix 1: Code

https://github.com/jlikhuva/Wheat

References

[1] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014), pp. 1–15. ISSN: 09252312. DOI: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[2] G. M. Murray and J. P. Brennan. "Estimating disease losses to the Australian barley industry". In: *Australasian Plant Pathology* 39.1 (2010), pp. 85–96. ISSN: 08153191. DOI: 10.1071/AP09064.

[3] Jason Yosinski et al. "Understanding Neural Networks Through Deep Visualization". In: (2015). arXiv: 1506.06579. URL: http://arxiv.org/abs/1506.06579.