Identifying Fungal Diseases in Growing Wheat using Convolutional Neural Networks.

Joseph Okonda L.
Stanford University
jlikhuva@stanford.edu

Abstract

We apply a VGG-16 based Convolution Neural Network to the task of identifying fungal diseases of wheat. Our model takes as input an RGB Image and outputs a class score for each of our pre-defined disease classes. We begin by tackling the binary classification problem of classifying whether the plant in the image is healthy or not. We then train a multiclass classifier that classifies an image into 5 categories. Our best model on the binary task achieves 85% accuracy on an unseen test set. On the second task, our best model achieves 56% accuracy.

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[6]. 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 Related Work.

This project was inspired by the observation that Neural Network based models have been successfully applied to various human health diagnostic tasks for instance [9, 5]. We've also had a few projects that attempt to apply Deep Learning techniques to Agriculture. Pryzant et. al. [8] use CNNs and LSTMs to track disease outbreak in Wheat from satellite imagery. Marcio Nicolau et. al [7] employ Convolutional Neural Networks to identify Wheat Kernels that have been damaged by Fusarium.

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

h

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

3 Data.

3.1 Data Collection.

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]. We use a variety of search terms, for instance "Head Blight of Wheat in Field" to retrieve images that we download. We do not manually label the images after retrieval and instead assume that the Search Engine retrieves images associated with only a given class when we run searches with terms from the said class. We do, however, manually inspect all the retrieved images and delete all images that are not associated with wheat e.g images of pdf papers. We do this in order to reduce the amount of noise in our data. Figure 1 gives a summary of the data.

3.2 Data Pre-processing.

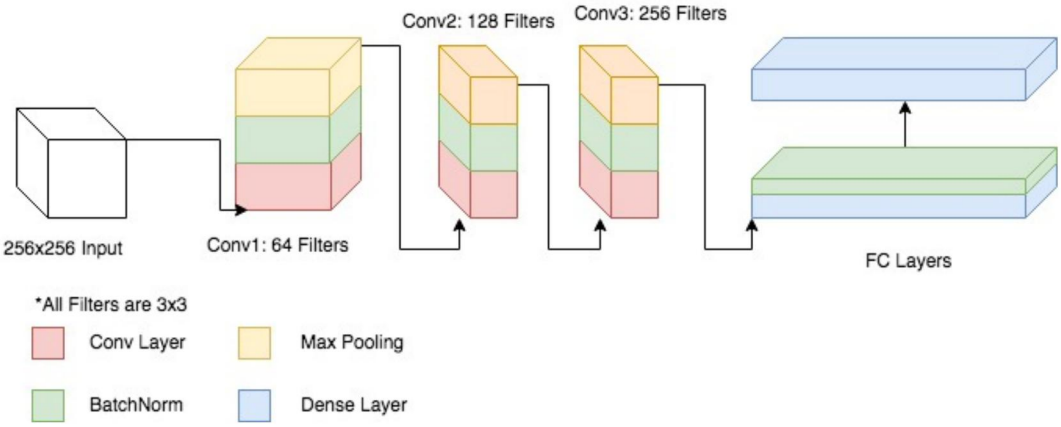
Before training and evaluation, we re-size all images to be 256x256x3. We choose 256 because it is divisible by 2 many times and thus makes our image and the resulting feature maps amenable to our pooling strategy. During Training, we employ the following class preserving transformations to augment our data: random horizontal flips, random brightness adjustment and random saturation adjustments. We do not use the common data augmentation technique of taking random crops of the input image because, in our case it is not class preserving. During both training and evaluation, the pixel values are converted to float values between 0 and 1. We split the data into 80% Training, 10% Development and 10% Test.

4 Models.

For our models, we use transfer learning. Specifically, in all our experiments, we initialize and freeze kernels in a subset of layers using weights from a VGG-16 [11] model pre-trained on ImageNet [3]

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

4.1 Binary Model.



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. This is followed by a Batch-normalization layer which normalizes the outputs of our layer in order to ensure that their distribution does not change as we proceed with training thus speeding up training [2]. This is followed by a pooling layer. We use max-pooling with a 2x2 window with a stride of 2. This divides the Width and Height of our feature maps by 2 Finally, we apply the ReLu activation function to the outputs of the pooling operation. All of the kernels used are 3x3 which extend to the full depth of the input volume, and are applied with a stride of 1. We pad the inputs such that the output of the convolution operation has the same dimension as the input, i.e SAME padding.

In addition to the convolutional layers, the networks we experiment with have between 0 and 3 fully connected layers. Additionally, we apply Dropout [10] as a regularization mechanism immediately after the convolution in the conv. layers or after the linear operation in the FC layers. In our experiments, we vary our 'dropout strength', i.e keep-prob between 0.7 and 0.9.

The network is trained to minimize the Binary Cross Entropy loss, i.e

$$-\frac{1}{M}\sum_{i=1}^{M} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - \hat{y}^{(i)}\right)$$

using mini-batch gradient descent with Adam as the optimization algorithm [4] for between 15 and 20 epochs. Our learning rate is 0.001. We use a batch size of 64 because it's the largest value that fits on the GPU we were working on. The 64 filters and biases in the first layer are initialized using weights from the first layer kernels of the VGG-16 model trained on Image-Net and frozen, meaning we do not update them during back-propagation. We randomly initialize the other parameters using Xavier initialization [1].

4.2 Multi-Class Model.

Our multiclass model is similar to the binary model with a few modifications. First, unlike in the binary model where we double the number of kernels after every layer, we keep the number of filters fixed at 64. We do this to reduce the capacity of our model so as to prevent overfitting. We cannot reduce the number of kernels below 64 because the model from which we're transferring uses 64 kernels in its first layer, therefore, to use fewer than 64 filters, we'd have to throw away some of VGG-16's feature detectors, which we hypothesize would hurt performance. Second, Our loss function is now the full cross entropy loss i.e

$$-\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{n\_classes} y_j^{(i)} \log \hat{y}_j^{(i)}$$

Third, in addition to dropout, we also use L_2 regularization. The increased regularization is needed because, unlike in the binary experiment where we had a few thousand samples per class, we now only have about 500 samples per class. We also use more aggressive dropout by increasing the

probability of zeroing out a give unit up to 0.6. We also use learning rate decay with an initial rate of .001 which we anneal using an exponential decay scheme every 1000 steps. Finally, the final layer is changed to a 5-way softmax.

5 Results and Discussion.

In this section, we present some of the results from our various experiments with the 2 models.

5.1 Binary Model

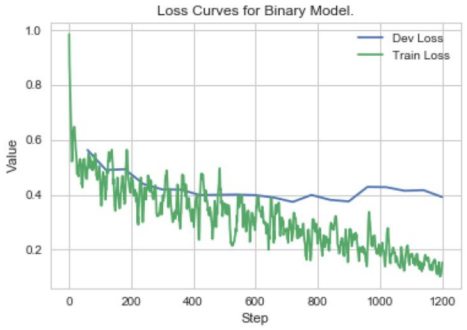


Figure 2: Loss

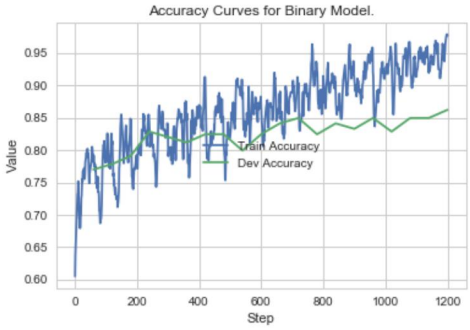


Figure 3: Accuracy

Figure 2 and 3 show the training curves for our best binary model.Evidently, Transfer learning does help the model in generalizing. However, as we shall shortly discover, it can only go so far, In particular, there seems to be a lower threshold to the number of samples per class the model needs in order for transfer learning to work.

5.2 Multi-Class Model

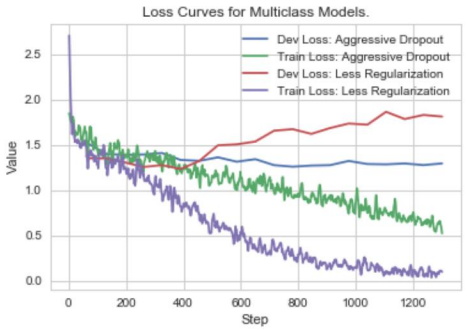


Figure 4: Loss

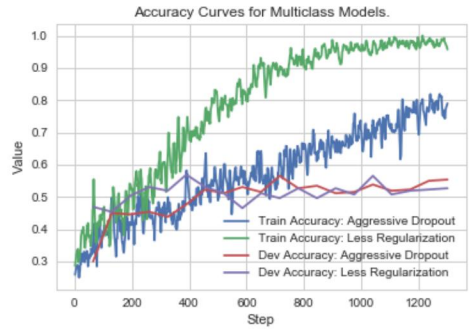


Figure 5: Accuracy

Without strong regularization, the model overfits (as evidenced by the divergence between the red and purple curves in Figure 4), even when we reduce the number of parameters by reducing the number of kernels per layer and freezing more convolution layers. Aggressive dropout, coupled with weight decay and learning rate annealing help reduce the models variance. However, even though this does not lead to a significant increase in performance. The accuracy on the development set (Figure 5) remains roughly the same. We attribute this to the fact that the number of samples per class is below the lower threshold needed for transfer learning to be helpful.

6 Further Work.

The main challenge in this project was collecting a large and rich dataset. As contrast to images closely related to human activity (e.g cats/dogs), it is hard to obtain non-stock photo images of wheat on the internet. Therefore, a natural extension of this project would be to repeat the same experiments with a richer dataset that is better annotated. Finally, since the methods used could easily be extended to other crops or other plant diseases as long as there exists a sizable richly annotated dataset.

7 Appendix 1: Code

https://github.com/jlikhuva/Wheat

References

[1] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Pmlr* 9 (2010), pp. 249–256. ISSN: 15324435. DOI: 10.1.1.207.2059. arXiv: arXiv:1011.1669v3. URL: http://machinelearning.wustl.edu/mlpapers/paper%7B%5C_%7Dfiles/AISTATS2010%7B%5C_%7DGlorotB10.pdf.

[2] Sergey Ioffe. "Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models". In: Nips (2017). ISSN: 1702.03275. arXiv: 1702.03275. URL: http://arxiv.org/abs/1702.03275.

[3] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 248–255. ISSN: 1063-6919. DOI: 10.1109/CVPRW.2009.5206848. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206848.

[4] 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.

[5] Yunzhu Li et al. "Skin Cancer Detection and Tracking using Data Synthesis and Deep Learning". In: (2016), pp. 1–4. arXiv: 1612.01074. URL: http://arxiv.org/abs/1612.01074.

[6] 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.

[7] Márcio Nicolau et al. "Fusarium Damaged Kernels Detection Using Transfer Learning on Deep Neural Network Architecture". In: (2018), pp. 1–8. arXiv: 1802.00030. URL: http://arxiv.org/abs/1802.00030.

[8] Reid Pryzant, Stefano Ermon, and David Lobell. "Monitoring Ethiopian Wheat Fungus with Satellite Imagery and Deep Feature Learning". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2017-July (2017), pp. 1524–1532. ISSN: 21607516. DOI: 10.1109/CVPRW.2017.196.

[9] Pranav Rajpurkar et al. "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning". In: (2017). DOI: 1711.05225. arXiv: 1711.05225. URL: http://arxiv.org/abs/1711.05225.

[10] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. ISSN: 15337928. DOI: 10.1214/12-AOS1000. arXiv: 1102.4807.

[11] Christian Szegedy et al. "Going Deeper with Convolutions". In: *arXiv:1409.4842* (2014). ISSN: 10636919. DOI: 10.1109/CVPR.2015.7298594. arXiv: 1409.4842. URL: https://arxiv.org/abs/1409.4842.