Capstone Three Final Report: Creating an Image Classifier to Predict Wheat Blast Disease

Introduction:

This capstone attempts to create an image classification machine learning (ML) model to address the problem of diagnosing the wheat fungal disease known as wheat blast. Wheat blast is a fungal disease that is already affecting wheat yields for farmers on several continents, and is continuing to spread worldwide. Currently, diagnosing wheat blast on wheat crops can require a professional botanist. However, an effective image classification ML model (that could potentially be deployed as a mobile or online app) that can identify wheat blast from photos would likely be cheaper, faster and easier for diagnosing wheat blast. This could potentially help farmers increase profits.

Fernandez-Campos et al's June 2021 publication is the first, to my knowledge, to use a convolutional neural network (CNN) image classifier to attempt to identify wheat blast (Fernandez-Campos et al). In addition to publishing their paper, they also published their dataset, which contains the images they used to build their classifier, and is free to download. The dataset contains about 6,000 wheat spike images photographed from a somewhat standardized distance and with a somewhat standardized lighting. The dataset comes pre-sorted into two folders, train and test, with a test_size of 0.2 (20% of data is in the test set, 80% in the training set). Within both the training and the test folders, the wheat spike images are organized into 3 categories. Category 1 is healthy spikes (0% blast severity), category 2 is moderate blast (0.1-20% blast severity), and category 3 is severe blast (20.1-100% severity). The images were sorted into these categories/folders via inspection by a plant pathologist (Fernandez-Campos et al.). To my knowledge, and judging from the publication (and brief personal inspection of the dataset), no images in this dataset were irrelevant, off-topic, poor quality, or need to be deleted.

Using this dataset, I trained two separate CNNs in an attempt to develop an effective classifier. The CNNs were trained to predict the wheat as belonging to one of the 3 categories of blast severity. One CNN was the pre-trained VGG16 CNN, which is a very deep neural network using a small 3x3 convolution filter that is known to be effective in classifying images. The other CNN was smaller (fewer nodes/layers) and was built from scratch. Line graphs were made of validation accuracy across epochs for both models. Ultimately, if one of my models is proven effective, it could have the potential to be developed into a mobile or online application that could be sold/marketed to farmers.

Problem Identification:

Wheat blast is beginning to affect wheat around the world. Wheat blast is a fungal disease that can heavily cut into the profits of farmers. Fernandez-Campos et al note that: "under conducive field conditions, the fungus can kill up to 100% of susceptible

wheat spikes in a period of 2.5–3 weeks (Gongora-Canul et al., 2020)" (Fernandez-Campos et al). Moreover, this fungus, since first being detected in Brazil in 1985, has since been gradually spreading to more continents (Fernandez-Campos et al). I will attempt to create a convolutional neural network (CNN) that correctly classifies wheat images as having blast or not, and also the severity. This could be used to help farmers identify whether their wheat is diseased or not, without having to consult an expensive botanist. A successful model could be made into an application that helps farmers identify wheat blast severity/presence from iphone/camera photos.

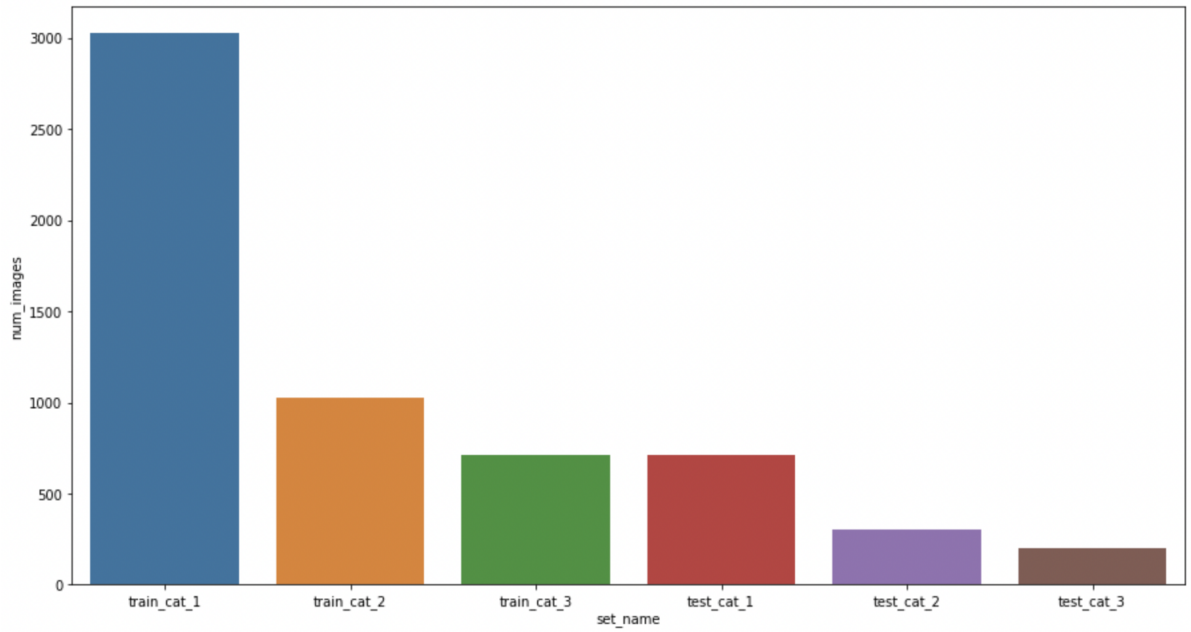
It is possible that utilizing a successful model could be faster, more accurate, and easier than a farmer diagnosing wheat blast/wheat blast severity with their naked eye. Similarly, bringing in a specialist/botanist/consultant would undoubtedly be far more expensive than an app.

The project will be considered successful if a model can be developed that successfully identifies wheat blast (or lack thereof) with reasonable accuracy and effectiveness. The solution space could encompass several different models with varying predictive capabilities. Within these models, a model will have to be chosen that appropriately straddles the tradeoff between predictive power and computational load. A constraint within the solution space is that the images from the wheat dataset were taken in a controlled environment that may not match the conditions in which wheat would be photographed in the future. Stakeholders for this project include potential investors in an application for predicting wheat blast. The only data source used will be the dataset from Fernandez-Campos et al.

Data Description:
The dataset used in this project will be the same one used and supplied (publicly) by Fernandez-Campos et al. This dataset consists of about 6,000 high quality images of wheat spikes (from an approximately standard distance, and approximately standard lighting), which is separated into a training folder (~4,775 images) and a testing folder (~1,231 images). The images provided have already been randomly split into training/testing with a test size of 0.2. Wheat spikes have been labeled by experts as

having no blast, moderate blast (0.1-20%), or severe blast (20-100%). Data may be augmented using standard data augmentation techniques, if deemed necessary.

I thought that the graph below helps illustrate the components of the dataset. ‘Cat’ denotes the blast severity category. This graph illustrates that the dataset is slightly ‘imbalanced’ (more images in some categories than others), and that a 80/20 train test split was applied.



Data Wrangling:
The data was downloaded onto hard disk from its source at https://purr.purdue.edu/publications/3772/1

The zip file was ~3 gigabytes. Then, this entire dataset was uploaded from hard disk to google drive. From there, google drive was mounted in google colab, and the dataset was imported from google drive to google colab, unzipped, and stored in a temporary folder in google colab. The dataset was determined to be clean, based on brief, random inspection of images and the discussion of the dataset in the article by Fernandez-Campos et al. The data came pre-organized into relevant folders for training and testing, as well as split into folders based on category of severity (category 1, 2 or 3).

Exploratory Data Analysis:
Exploratory data analysis consisted of several steps: displaying some of the images, plotting image sizes and visualizing aspect ratio, visualizing the number of images in the dataset/subsets, and image feature analysis.

The first step, displaying some of the images, was conducted in order to inspect and get an overview of the dataset. Images from all 3 categories of blast severity (0% or healthy, 0.1-20% diseased, and 20.1-100% diseased) were shown, and comments above the images displayed which category each image was in. Furthermore, images from both the training and test datasets were shown. Based on visual inspection of these images it could be seen that all images appeared to be photographed in front of a blank background from an approximately standard distance. In addition, all images seemed high quality and high resolution, and each image included the entire wheat spike plus some blank background. To me, it seemed difficult, as an untrained observer, to detect blast with the naked eye. It appeared to me that there was bleaching (which is a reported symptom of wheat blast) evident in the category 3 images, and not the category 1 images, but this difference was somewhat subtle.

In step two, 4 separate graphs were made showing the image sizes for the entire training dataset (including all 3 categories), the subset of the training dataset labeled as category 1, the subset of the training dataset labeled as category 2, and the subset of the training dataset labeled as category 3, respectively. Each dot on the graphs was one image, the x-axes of the graphs denoted the number of rows of an image, and the y-axes of the graphs denoted the number of columns of an image. The mean aspect ratio for the entire training set was 0.5.

In step three, graphs were plotted showing the relative sizes of different subsets of the dataset, including training set relative to testing set, and different categories to another.

In step four, canny edge features for images from category 1, 2 and 3 were plotted to look for useful features. No obvious features (that could predict blast) based on canny edges could be detected with my naked eye. Fast features for images from category 1, 2 and 3 were plotted. Similarly, I had some trouble detecting blast with the naked eye from observing fast features, although some potentially promising differences between categories were detected that deserve more investigation. Plots were made of the intensity of these features. I examined these plots to try and distinguish reliable differences between categories, but still fell short. Some differences seem to disappear once more images were inspected, but it seemed possible that other subtle differences could potentially be discovered with further investigation.

Preprocessing and Training Data Development:

Image Resizing:
Keras' Image Data Generator was used within google colab to feed the images to the model, to train the model. Image Data Generator is useful because it feeds images to the model in smaller batches, rather than all at once, which does not require using as much RAM. When the dataset consists of thousands of images, like the one being used, using smaller batches is necessary, especially with limited computing power. The images in the dataset used were a wide variety of sizes. Image resizing was taken care of within the CNN itself, with each CNN I made having a resizing layer at the beginning to convert all images to 224x224.

Train Test Split:
Importantly, the dataset used came with a train test split already applied. The test size was 0.2. The data also came organized into the 3 categories of blast severity, with each different category being a different folder containing different images. Because this dataset is coming from a reliable source, the researchers at Purdue University (Fernandez-Campos et al), it can be assumed that the train test split and organization of images into categories was done properly. The number of images found in the training and test set confirms a test size of 0.2. When calling the .fit_generator function on a Keras CNN model, the training data folder was assigned to the 'generator' parameter and the test data to the 'validation_data' parameter.

Modeling:

Two models were created in an attempt to effectively predict wheat blast severity based on wheat spike images. CNNs are known to be highly effective ML algorithms for image classification, and so both models were chosen to be CNNs. The first model (let's call it model 1) was a pre-trained model known as VGG16, which is quite deep (many layers and parameters) with 16 layers and was trained on the imagenet data (https://www.kaggle.com/keras/vgg16). VGG16 is also unique because it uses a very small (3x3) convolution filter. This model won the imagenet image classification competition in 2014, and is considered one of the best performing image classification algorithms to date (https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c).

It is thought that pre-trained CNNs can often perform even better on new data than CNNs built from scratch, because they take some of the experience and performance of an already successful model and apply it to a new problem (new images).

The second model (let's call it model 2) was a CNN I built from scratch that was simpler (fewer layers) than VGG16. This model had two convolution layers, followed

by a max pool layer, then two more convolution layers, then another max pool layer, and then two more convolution layers and a max pool layer. I tried to not have too many layers in this simplistic model, but found that the number of convolutions I chose allowed the model to greatly simplify the essentials of the images down (which were initially a large number of pixels) which could then be run through a single dense layer. I chose to have this number of convolution layers because VGG16 and other existing models can be known to be successful because they really cut down on the pixels being analyzed and the convolution layers serve to really strip the image down to its bare/important essentials. Having fewer convolution layers also would have used more computational power and taken longer to run.

A second dense layer with 'activation'='softmax' and units=3 was added at the end of each of the CNNs (VGG16 and the one from scratch) I used, to tell the CNN to sort into 3 categories. A CNN implemented with Keras also knows to perform categorical classification with 3 categories because there were 3 folders inside the training and test set-for category 1, 2 and 3.

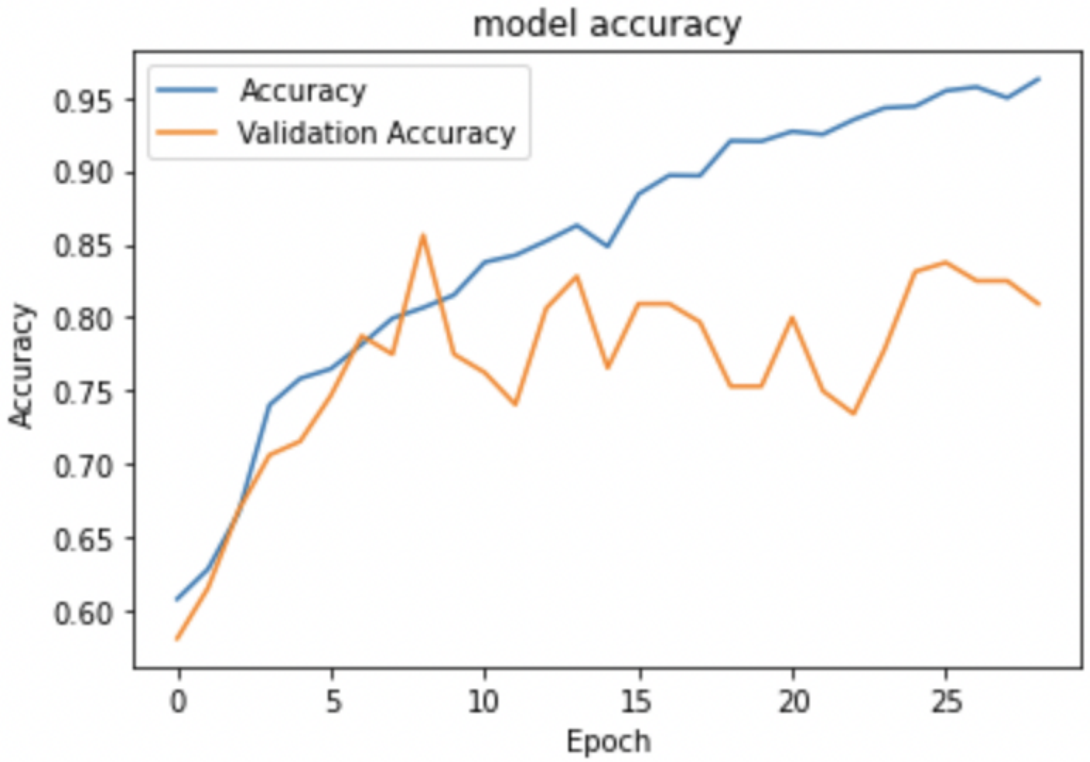
Keras' model checkpoint and early stopping functions were used to save the best performing model every few epochs and to stop the model early if validation accuracy was not found to be improving. Both models were set to run for a maximum of 100 epochs each if early stopping was not activated.

Model 1 ran for 29 epochs until early stopping was activated, after the validation accuracy of model 1 failed to improve from 0.85625 in epoch 9.

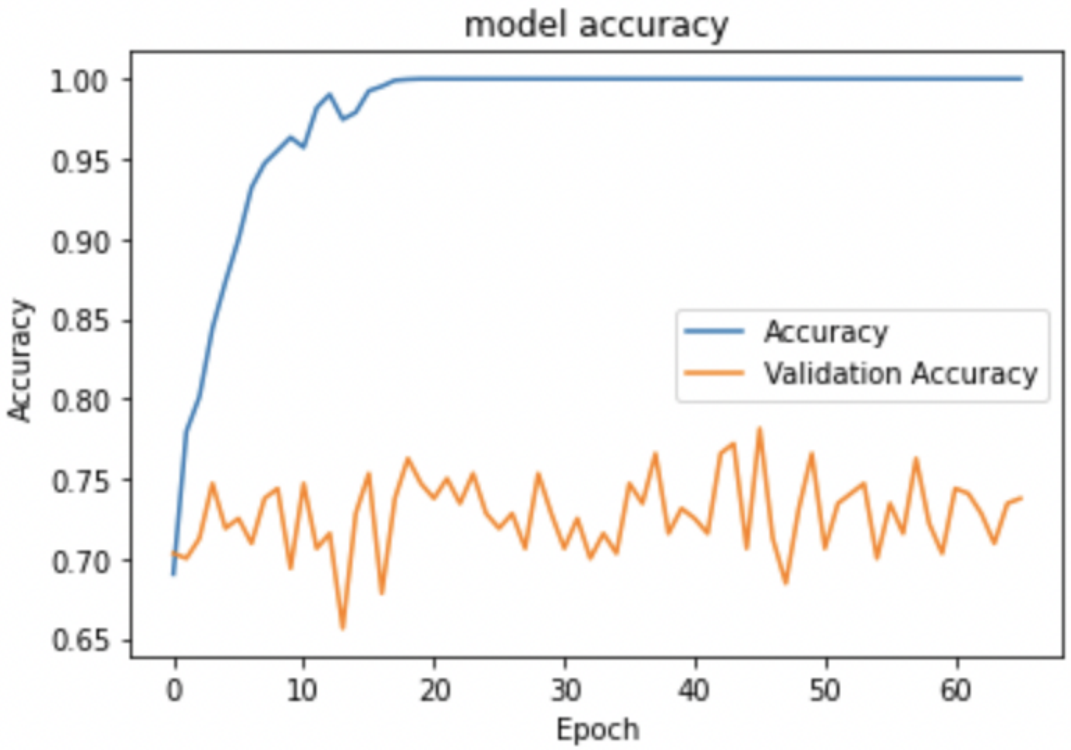
Model 2 ran for 66 epochs until early stopping was activated, after the validation accuracy of model 2 failed to improve from 0.78125 in epoch 46.

Accuracy, validation accuracy, loss and validation loss were plotted over epochs for both model 1 and model 2 after each model was run.

Comparing models:
I think that the two graphs below help illustrate the accuracy of each model. The first graph displayed here is model 1 (pretrained, VGG16) accuracy and validation accuracy over epochs. It can be seen that this model's validation accuracy, after it's initial rise over the first ~7 epochs, seems to hover around 0.80. Notably, the fact that accuracy continually increases in both graphs below is not out of the ordinary. Validation accuracy is a far more useful metric than accuracy because accuracy measures the model's fit on the training set (while validation accuracy is on an 'unseen' set). A model's 'accuracy' can be easily gamed through overfitting.



This graph below show's model 2's (model created from scratch) accuracy and validation accuracy over epochs. It can be seen that validation accuracy seems to hover around 0.73. Notably, the graph above and this graph seem to illustrate that the pretrained model (model 1) has consistently higher validation accuracy than model 2, the simpler, created-from-scratch model.



Final Results and Conclusion:
And so, because model 1 (pretrained, VGG16) has consistently higher validation accuracy than model 2 (simpler, created-from-scratch), model 1 is the better model in a scenario where model performance is the only consideration. It is possible that in a scenario where performance is not as paramount and can be sacrificed a little bit for the sake of simplicity and a lower computational burden, it is possible that model 2 would be a better choice.

Model 1 may require a lot of computational power to make predictions in real time, and thus its usability may be low. Acquiring computational power can be costly, and if the cost is too high, it could make an app unprofitable.

Some ideas for further research are to try other pretrained models, fine tune the hyperparameters of the model, attempt k-fold cross validation with the model, and try data augmentation before modeling. Other ideas for further research are potentially training on additional relevant datasets to improve predictive power, or changing the type of prediction being made by merely trying to predict presence or absence of blast instead of 3 categories of severity.

One client recommendation is to figure out which model, if either, has better predictive power than your average expert, and then choose that model to turn into an application. If model 2 has slightly (e.g. 0.1 accuracy) worse performance than an

expert, could it still be worthwhile to turn it into an app because it could be cheaper and faster for clients than consulting an expert? Could an app be marketed to farmers who want to catch blast early or marketed to farmers who want to identify the more resistant cultivars in their fields?

A second client recommendation for how they can use my findings is-use it to quickly identify more resilient variants of wheat, without having to wait longer to see blast develop more. Could sell this to agriculturalists who are trying to create blast-resistant cultivars.

A third client recommendation is-try and liaise with botanists and ML professionals to figure out more visual features to focus on, try and augment this algo (e.g. perhaps with the ML technique known as ‘attention’) and make it better. Attention has been found to help make some image classification model performance better (Sitaula et al.), and so if it improves performance and does not result in highly increased computational costs, it may be worth implementing.

Another recommendation-even if app recommendation is on par with botanist prediction and not better, the app could be marketed to farmers as a second independent look. This could be especially appealing if it’s quick, cheap and easy. Since iphones/smartphones are prevalent these days, even in some developing countries, the model could be turned into a smartphone app for the broadest use and potentially highest profit and publicity. Farmers may not require a trained specialist to use the app. The app could be used on its own and on the go.

Acknowledgements:
I would like to thank my mentor Branko Kovac for providing me with constructive feedback on this project.

Works Cited

Fernandez-Campos, Mariela, et al. "Wheat spike blast image classification using deep convolutional neural networks." *Frontiers in Plant Science* 12 (2021): 1054.

https://www.kaggle.com/keras/vgg16

https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

https://purr.purdue.edu/publications/3772/1

Sitaula, Chiranjibi, and Mohammad Belayet Hossain. "Attention-based VGG-16 model for COVID-19 chest X-ray image classification." *Applied Intelligence* 51.5 (2021): 2850-2863.