RustID: Rust Identification in African Wheat Plants

Rohan Rajpal
IIIT-Delhi
2017089
rohan17089@iiitd.ac.in

Nirav Diwan
IIIT-Delhi
2017072
nirav17072@iiitd.ac.in

Tejas Oberoi
IIIT-Delhi
2017367
tejas17367@iiitd.ac.in

Abstract

Wheat Leaf Rust is a devastating disease affecting the crops of millions of African farmers. Detecting such plant diseases early can save tonnes of agricultural produce every year. This will benefit not only the general population by reducing hunger, but also the farmers by ensuring they get the harvest they deserve. To address this challenge, we propose a deep learning algorithm which given an image of a plant, classifies if the plant is healthy, has stem rust or has leaf rust. We explore a variety of state-of-the-art models and come up with a modified DenseNet Model of our own. Our experiments explore the performance of our model with various ablations. We submitted our model at ICLR Workshop Challenge 1: CGIAR Computer Vision for Crop Disease. We stood 39th among 847 participating teams and our model achieved a competitive log-loss of 0.265. For the IIIT-Delhi Deep Learning Course, we are ***group-18****.*

1. Introduction

1.1. Motivation

Wheat leaf rust is a fungal disease that affects wheat, barley and rye stems, leaves and grains. A significant number of African crops are affected by this devastating plant disease every year. This affects the livelihoods of numerous farmers posing a threat to food security across the continent. Control and eradication of this disease are challenging since it is not easy to monitor at a large scale.
Devising a feasible technological solution to resolve the pressing issue is the need of the hour. We propose building an image recognition model that is capable of detecting wheat rust in any image. This could enable a crowd-sourced approach for monitoring African crops through avenues of the likes of social media and smartphone images.
This challenge represents a potential breakthrough in our ability to monitor and control plant diseases like wheat rust that affects the livelihoods of farmers all over the world.

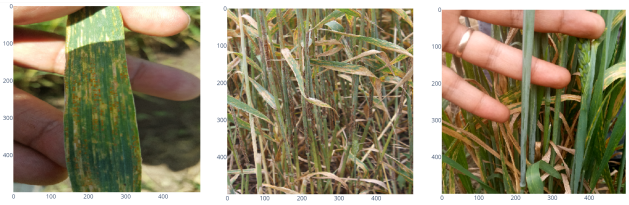

Figure 1. A sample from each class:Leaf Rust, Stem Rust and Healthy Wheat respectively

1.2. Problem Statement

The objective of the challenge is to build a deep-learning algorithm to correctly classify if a wheat plant is healthy, has stem rust, or has leaf rust. It might be the case that a few images have both stem and leaf rust. The goal is to classify the image according to the dominant type of wheat rust in the image. Sample images for the three categories can be observed in Figure 1.

2. Related Work

Analysis of characteristics of wheat diseases done in the past concludes that wheat disease primarily concentrates on the wheat leaves and stem and can be identified by computer vision techniques. Dixit et. al [1] (2018) claim that Image processing and machine learning technique over support vector machine have been extensively explored for plant disease study for their merits of invasive, rapid, continuous and precise measurement capacities.
Rong et al. [14] (2013) have utilized hybrid algorithms of template matching and support vector machine (SVM) to achieve Early Cercospora leaf spot detection in sugar beet. A three-stage-framework was adopted to achieve accurate results. Their work claimed feasibility with continuous quantization under natural daylight conditions.
Rajleen and Sandeep [4] presented the results on support vector machine and neural network with a suitable comparison. Accuracy, time and area detection were the three parameters the results incorporated. Their findings indicated SVM to be better classifiers than neural networks.

Peifeng Xu et al. [10] (2017) designed a wheat-leaf disease detection system based on embedded image recognition. The steps included firstly transforming the captured images into G single G channel image of the RGB model. They used the Sorbel operator method for virtual detection of edges in the grey image, performed background elimination and extraction of binary feature point set of disease spot. They used a flood filling algorithm for filtering out the noisy points found in the point set. Finally, they compute the area ratio of disease spots and leaf area to attain a precise diagnosis of disease level. They claimed to achieve accuracy levels comparable to human vision.

3. Methodology

3.1. Dataset

The dataset has been acquired from the ICLR Workshop challenge: CGIAR Computer Vision for Crop Disease [1] . There are 876 images in the train set and 610 images in the test set. We performed a stratified random sampling with an 80:20 train-val split on the original train set to acquire a new train set consisting of 700 images and a validation set consisting of 176 images. The major part of the data is collected from CIMMYT [2] and a few have been acquired from public Google images. The distribution of the training set is summarized in table 1 . The train set is found to have class imbalance with Healthy Wheat having the least number of samples. See Figure 2.

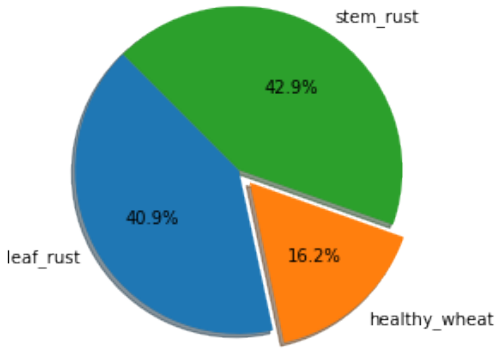

Figure 2. Pie chart indicating class-wise distribution of Train-set.

3.2. Analysis

The RGB channel values of the images were analysed (Refer Figure 3). The channel values were found to have a

Class Label	Number of Samples
Leaf Rust	358
Stem Rust	376
Healthy Wheat	142

Table 1. Class wise Distribution of Images in original Train Set

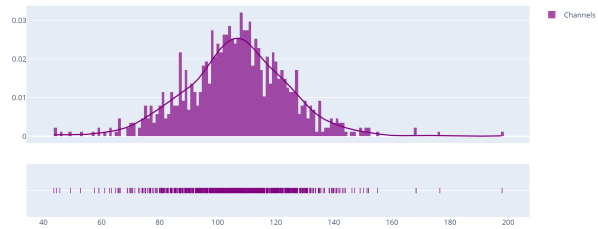

Figure 3. Distribution of Channel Values

roughly normal distribution centred around 106. The maximum channel activation is 255. This means that the average channel value is less than half the maximum value, which indicates that channels are minimally activated most of the time.

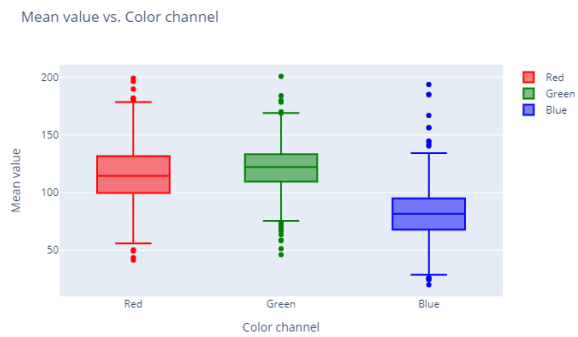

Figure 4. Mean Value vs Colour Channel Plot

Figure 4 displays the mean value vs colour channel statistics. Lower end of the box indicates q1,middle line indicates the median and the upper end of the box indicates q3. Q1 is the median (the middle) of the lower half of the data, and Q3 is the median (the middle) of the upper half of the data. Green is the most pronounced channel with 122 as median value, followed by red with 114.5 as median value and blue with 81.5 as median value.

3.3. Evaluation

We use log-loss as our metric of classification In a single image, more than one class could be present. Thus multiple classes will have non-zero probabilities. It penalizes those predictions more severely which are incorrect with high confidence values. Therefore we deem log-loss as a suitable

[1] ICLR Workshop Challenge 1
[2] CIMMYT

measure for the classification problem at hand. Loss for a single data sample is mentioned in equation (1)

$$loss = \text{-} \sum_{c=1}^{M} y_{\text{o,c}} log(p_{\text{o,c}}) \tag{1}$$

M denotes the number of classes which is 3 in our case. y is a binary indicator (0 or 1) denoting if class label c is the correct classification for observation o. p indicates the predicted probability of the event that observation o belongs to class c.

3.3.1 Design Choices

Since we are provided with the correct labels for the train set, we approach the problem as a supervised learning problem. Based upon discussion with Dr Saket Anand, our objective is to construct an end-to-end deep learning model which provides the probability of a wheat plant having rust on it's stem or on its leaves. We approached the problem by:

1. Training architectures we had learned throughout DL Course such as ResNet, DenseNet etc. on common training paradigms.
2. Exploring State of the Art (EfficentNet) which were not in our course material.
3. Exploring different architectures with varying training paradigms.

The model selection startegy proceded as follows -

1. Firstly, using a set of default paradigms we trained models mentioned in section 3.3.4. Out of them, we selected the best model based on the training paradigms we we trained the models for.
2. Next, various ablations were performed on the selected model.

It is to be noted that since EfficientNet is harder to train, we tried more than just the default training paradigms for them. We have included them in our final results.

3.3.2 Augmentations

1. **Vertical and Horizontal Flips** - Randomly images are vertically and horizontally flipped in the dataset.
2. **Augmix*** [2] - Augmix randomly samples augmentations of an image to generate a combined representation. This also acts as regularizer which produces a better overall performance. However, we did not include it due to the lack of time.
3. **Mixup***[12] - involves mixing up pairs of data samples to generate transformed images. Mixup has shown to act as a strong regularization technique. Mathematically, it can be represented as -

$$x_t = \lambda \cdot x_i + (1 - \lambda) \cdot x_j$$
$$y_t = \lambda \cdot y_i + (1 - \lambda) \cdot y_j$$

   where x_t, y_t are the transformed images. x_i, x_j are a pair of images and y_i, y_j are the one-hot encoded labels.
4. **Cutmix*** [11] - Like Mixup, Cutmix involves mixing up pair pf images to generate transformed images. Unlike Mixup, CutMix replaces an image region with a patch from another training image and generates more locally natural image than Mixup does.

3.3.3 Optimizers

1. **Adam**[5] the current benchmark for optimizers and is popularly used in many deep learning models.
2. **R-Adam**[6] - R-Adam or Rectified Adam provides a slight adjustment to the Adam optimizer by varying the adaptive learning rate based on a detailed study into the effects of variance and momentum during training.
3. **Ranger*** - It is a optimizer which combines the R-adam with LookAhead [13] optimizer.

3.3.4 Architectures

1. **ResNet**[7] - Uses gated shortcut connections in deep neural networks to combine knowledge from previous layers with new layers, and to solve the vanishing gradient problem.
2. **SE-ResNeXt** [9] - A combination of ResNet, inception network and Squeeze and Excitation Network. The outputs of different paths are combined together along with adding skip-connections in between the different variations.
3. **DenseNet** [3] - In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Each layer receives the 'collective knowledge' from all preceding layers.
4. **EfficientNet*** [8] - The overall architecture of EfficientNet consists of an inverted residual blocks along with a squeeze and excitation block injected in between the blocks. This is the current state-of-the-art on ImageNet as of January 2020. We experimented with EfficientNet b6 architectures.

* denotes architectures and methods which were not present in the Deep Learning curriculum of IIITD (winter 2020).

Test Time Augmentation We also employed test time augmentation (TTA) whose purpose is to perform random modifications to the test images. Instead of showing the regular, "clean" images, only once to the trained model, we will show the model the augmented images several times. We will average out the predictions of all corresponding predictions and then average out the predictions of each corresponding image and take that as our final guess.

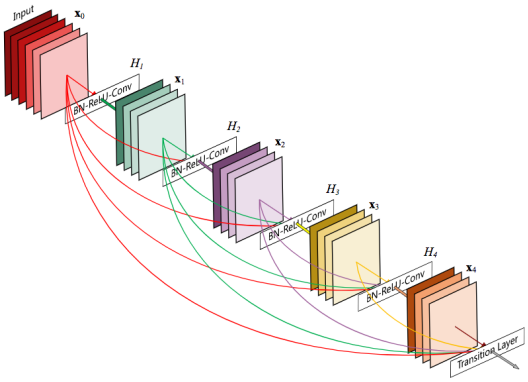

Figure 5. Architecture of DenseNet-201 which gave the best performance on the CGIAR dataset. In the typical DenseNet-201 Architecture [3], we added batch-normalization layers and a dropout of 0.5 after every block.

3.3.5 Training Procedure

Firstly, to prepare the validation set we **stratified split** the training set and performed **5-fold cross-validation**. Next, in our model pipeline, images were randomly flipped vertically,horizontally or by 90 deg. We also varied our learning rate by using **cosine annealing**. In, cosine annealing, initially the learning rate is kept high. As the model approaches the local minimum, the learning rate is accordingly lowered. For all models, we initialized the models using the pre-trained weights (trained on ImageNet). We used a maximum learning rate of 0.001 and a minimum learning rate of 0.00002.

3.4. Experiments and Observations

3.4.1 Best Model

DenseNet-201 architecture produced the best performance. The modifications included adding a batch normalization layer and a dropout layer of 0.5 after every block of convolutional layers [c.f fig 5]. Secondly, we trained the first three blocks with a maximum learning rate of 0.001 and the latter two blocks with a learning rate of 0.00002. Our reasoning for doing so is that for the lower-level features are more finetuned to our dataset, the change may be reflected in higher level features itself. Hence, a lesser learning rate is needed to finetune the higher level features. We experimented with the above mentioned augmentations and optimizers. We observed that the Adam optimizer performed better than R-Adam, at default settings of $\beta_1 = 0.999$ and $\beta_1 = 0.9$. The best modified DenseNet-201 produces a log-loss at 0.265 , refer figure 6. The final results obtained along with the other baselines are mentioned in table 2. **We beat the state of the art EfficientNet - b6 using our modified DenseNet-201 architecture**. Although it is to be noted that EfficientNet was not designed for this particular task.

Model	Log Loss
Resnet-18	0.542
ResNet-34	0.469
Densenet-121	0.400
SE-ResNext	0.307
EfficientNet-b6	0.296
DenseNet-201	**0.265**

Table 2. The combined performance on the test set of all baselines considered and the the modified DenseNet-201 model.

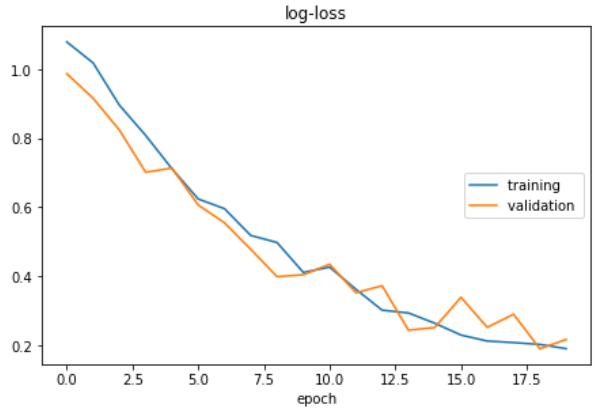

Figure 6. Loss plot of the modified DenseNet-201, the validation loss appears to stabilize after 20 epochs.

3.4.2 Separation of classes

We also visualize the features obtained by our model on the validation set as seen in figure 7. We conclude that our model can effectively classify the samples into the mentioned classes.

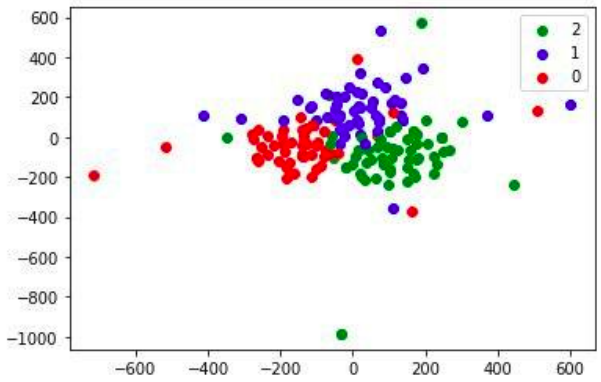

Figure 7. A TSNE visualization of features of the validation samples, depicting separability of the validation samples. Red, blue and green represent the healthy wheat, leaf rust and stem rust respectively.

3.4.3 Class-wise performance

We examine the class wise performance and found that the healthy wheat class is best classified class, followed by leaf rust and stem rust respectively, refer table 3. We obtain an overall accuracy of 92.57% on the validation set.

Class	Accuracy	Precision	Recall	F1 score
Healthy Wheat	97.71	0.96	0.96	0.96
Leaf Rust	93.71	0.92	0.91	0.92
Stem Rust	93.71	0.9	0.91	0.91

Table 3. Performance on different classes. (Accuracy is denoted in percentage)

Model	Log Loss
EfficientNet-b6	0.325
EfficientNet-b6 + Cutmix	0.307
EfficientNet-b6 + Mixup	0.297
EfficientNet-b6 + Mixup + VHF	0.296

Table 4. Abalative results of the EfficientNet-b6 + TTA Models

Model	Log Loss
DenseNet-201	0.287
DenseNet-201 + Cutmix	0.281
DenseNet-201 + Mixup	0.275
DenseNet-201 + Mixup + VHF	**0.265**

Table 5. Abalative results of the DenseNet-201 +TTA Models.

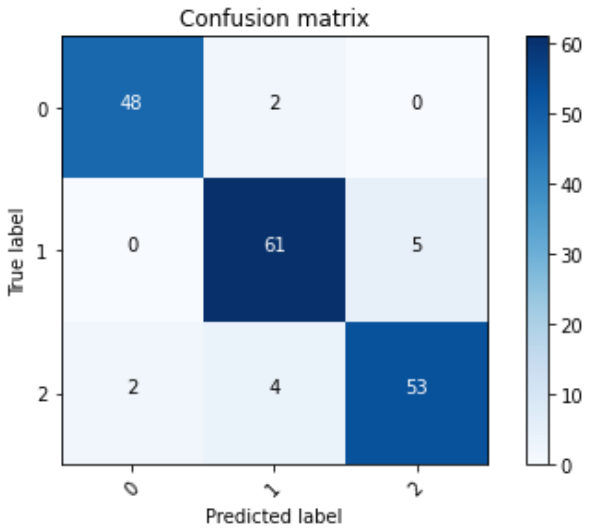

Figure 8. Confusion Matrix of the final modified DenseNet-201 architecture. 0,1 and 2 represent the healthy wheat, leaf rust and stem rust classes respectively. The model best identifies healthy plants.

Model	Log Loss
no ensemble	0.296
with ensemble	0.297

Table 6. Efficientnet with ensembles for all 5 folds.

Model	Log Loss
no TTA	0.532
with TTA	0.421

Table 7. Densenet and TTA

Figure 9. A sample image from the dataset which depicts, intuitively why vertically flipping the image should not work.

3.4.4 Ablation

1. **Stratified Split increased the loss:** On applying stratified split to make validation set, we observed that the

log loss on the test set increased by a value 0.23. This may be because the test set in itself is imbalanced. Alternatively, our model is perhaps better at learning one class than the other.

2. **Oversampling increased the loss:** Oversampling the healthy wheat class drastically increased the log loss on the test and validation set, although it decreased the training loss, indicating that adding many duplicates to the model may cause over-fitting.

3. **Effect of cosine annealing:** Cosine annealing marginally improved the performance, decreasing the loss by 0.07.

4. **Vertical Flips actually decrease the log loss:** Initially, due to images such as in figure 9 which has a clear distinction between the sky and the grass, we expected that vertically flipping the images may increase the loss. However, the inverse happened, we noticed that the loss actually decreased. This may be because images like these may add much needed noise, thus improving the model.

5. **Mixup augmentation performed the best:** Out of the mentioned augmentations, Mixup augmentation performed the best. This may be beacuse, using Cutmix may be leading to wrong random patches of the images being shuffled (i.e a rusted stem may be replacing a healthy stem in another image.)

6. **Test Time Augmentation (TTA) improved the test log-loss:** Utilizing test time augmentation decreased the test loss on the modified DenseNet-201 by 0.17. The augmentations used on the test set were exactly the same as on the train and validation set.

4. Work Distribution

The work distribution was as follows -

1. Rohan Rajpal(2017089) - implemented the baselines of ResNext, EfficientNet-b6. Studied and implemented all optimizers.

2. Nirav Diwan(2017072) - implemented the baselines of DenseNet-121, DenseNet-201. Studied and implemented all augmentations and pre-processing.

3. Tejas Oberoi (2017367)- implemented the baselines of ResNet-121, ResNet-201. Performed EDA on the dataset.

Additionally all three of us contributed in performing the ablations on the final model and contributed in creating the literature survey. We are group number - 18.

References

[1] A. Dixit and S. Nema. Wheat leaf disease detection using machine learning method—a review. *Int. J. Comput. Sci. Mob. Comput*, 7:124–129, 2018. 1

[2] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019. 3

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 3, 4

[4] R. Kaur and S. S. Kang. An enhancement in classifier support vector machine to improve plant disease detection. In *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*, pages 135–140. IEEE, 2015. 1

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[6] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 3

[7] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 3

[8] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 3

[9] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 3

[10] P. Xu, G. Wu, Y. Guo, H. Yang, R. Zhang, et al. Automatic wheat leaf rust detection and grading diagnosis via embedded image processing system. *Procedia Computer Science*, 107:836–841, 2017. 2

[11] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019. 3

[12] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3

[13] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9593–9604, 2019. 3

[14] R. Zhou, S. Kaneko, F. Tanaka, M. Kayamori, and M. Shimizu. Early detection and continuous quantization of plant disease using template matching and support vector machine algorithms. In *2013 First International Symposium on Computing and Networking*, pages 300–304. IEEE, 2013. 1