Contents

0.1 Architecture

0.1.1 Feature Extraction

Phase 1 of our model involves extracting the image features using a pretrained (on ImageNet dataset) version of Keras MobileNet [2] model which is sliced at the last convolutional layer. This is accomplished using the encode_images method of class CNN in the python library imagededup [3]. The method generates a feature vector (of length1024 in our case) for each image in the specified directory by propagating the images through the above mentioned pretrained and sliced version of Keras MobileNet and generate required encodings which are then passed as input to the LightGBM[1] classifier (for training and drawing inference).

0.1.2 Classification

We used one of the most widely used gradient boosting framework - Light Gradient Boosting Machine (LightGBM) in this phase. We train the LightGBM model under 7-fold cross validation. The significant hyperparameters are set as follows:

1. metric = [‘multi_logloss’, ‘auc_mu’],
2. learning_rate=0.15 (with Cosine Annealing with max=0.15, min=0.01, T=10)
3. num_boost_round = 2500,
4. boosting = ‘dart’,
5. max_depth = 7,
6. num_leaves = 64,
7. max_bin = 63,
8. reg_lambda (L2) = 0.25
9. drop_rate = 0.4
10. bagging_fraction = 0.55
11. bagging_freq = 4

0.2 Regularization Techniques

In our algorithm, we applied the decision tree approach called "LightGBM" proposed by Ke et al.[4] for efficient classification. Implementation of the technique is based on: https://github.com/microsoft/LightGBM. According to the documentation of the Github project, the package provides two regularization methods: L1 (Lasso Regression) and L2 (Ridge Regression). L1 regularization aims to shrink the less important feature's coefficient to 0 while L2 regularization aims to adjust the less important feature's coefficient to very small non-zero value. Since in the decision tree we have already selected our features, eliminating them would not be helpful. Thus, we used L2 regularization that penalizes the weights to be small instead of 0.

$$Loss_2 = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2 \tag{1}$$

Also, the setting of boosting parameter to DART (Dropouts meet Multiple Additive Regression Trees) and setting limit of 7 on tree depth, a limit of 64 on number of leaves and a limit of 63 on number of bins also help regularize the model. Furthermore, the use of bagging (use fraction of training in each epoch) and and augmentation of training data also result in regularizing the model.

0.3 Training Details

The MobileNet model used to extract features is pre-trained on ImageNet dataset (See `https://idealo.github.io/imagededup/methods/cnn/` for details). The LightGBM classifier model is trained using 7-fold cross validation on feature vectors extracted from training images in prior step.

0.4 Data Processing

Our pre-processing pipeline is like this:

First we did cleaning on the data set and found 280 images where there were no plant, 2 or 3 different plants in the same image, or images where there was something in the camera's way. We found these images by clustering (Gaussian mixture method) features given by MobileNetV3.

Next we augmented the data set to 104k images using Imgaug library. We used: random horizontal flips, random crops, weak Gaussian blur, vary

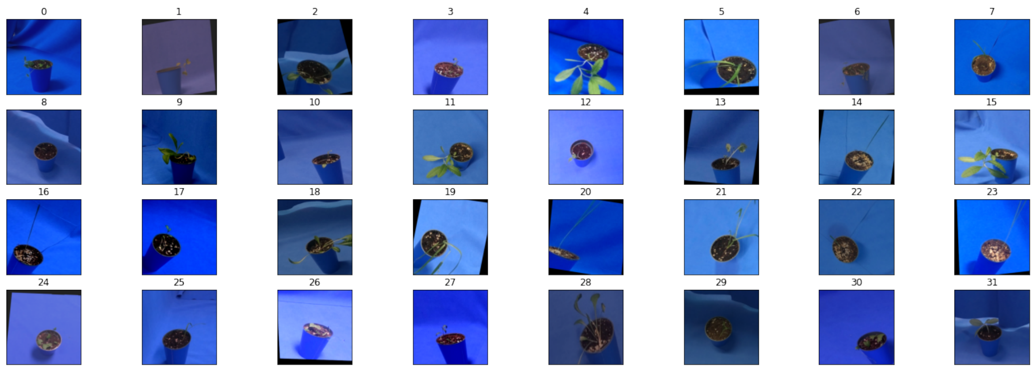

Figure 1: Augmented images

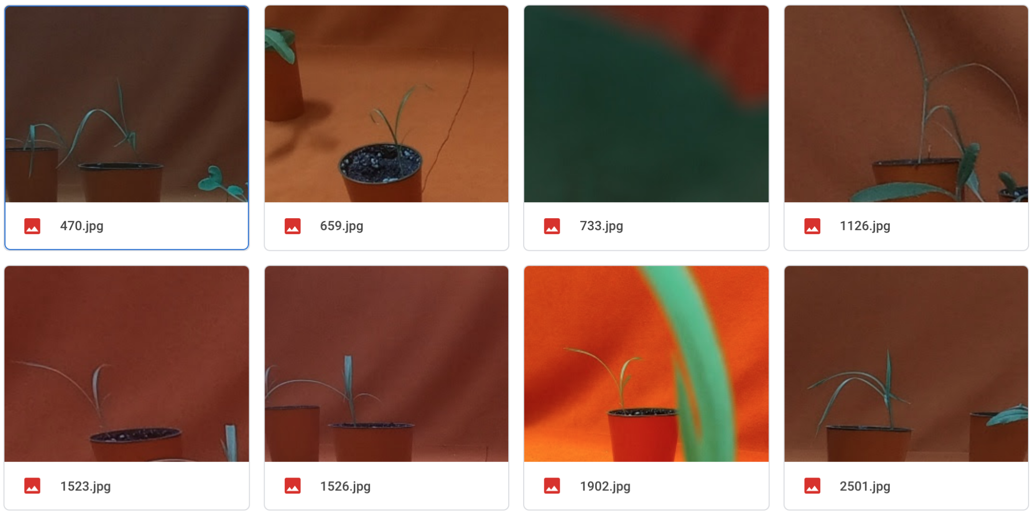

Figure 2: Removed images

the contrast, piecewise affine, perspective transforms, random affine, and changed the lighting of images.

We also tried leaf segmentation using k-mean. Although our methodology segmented leaves perfectly (it also excluded the soil), it didn't improve the test accuracy by more than 63%.

We trained a 7-fold LGBM+DART model on P100 GPU in 57 minutes and the min testing accuracy on these folds were 76.81%, max testing accuracy was 77.89%, and voting gives 77.78% test accuracy.

0.5 Top 1 Accuracy

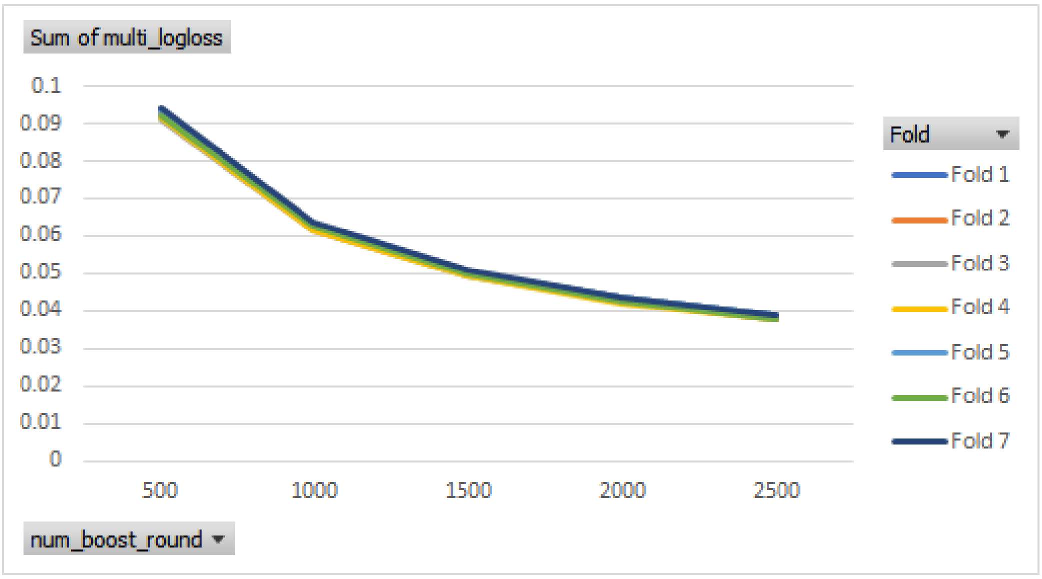

Figure 3: Multi-log loss

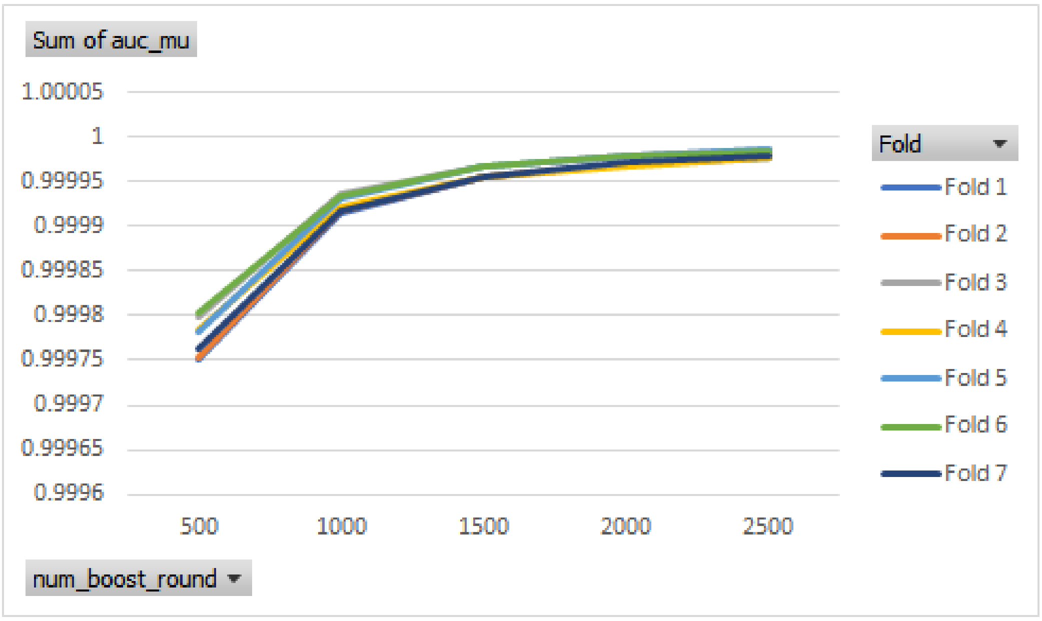

Figure 4: AUC_mu

References

[1] microsoft/lightgbm. `https://github.com/microsoft/LightGBM`.

[2] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[3] Tanuj Jain, Christopher Lennan, Zubin John, and Dat Tran. Imagededup. `https://github.com/idealo/imagededup`, 2019.

[4] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.