Tree Species Image Classification

Martin Engström
Chalmers University
martine@chalmers.se

Abstract

In this technical report, three techniques are used to improve a baseline CNN architecture with the task to classify twelve unique species of trees from a dataset containing textures of tree bark. The techniques used were standardizing the image data, adding batch normalization, and using early stopping. These techniques were proven to be effective for reducing overfitting, decreasing training time, increasing training stability, and increasing evaluation accuracy through an ablation study. The final model utilizing all of the techniques reached a maximum accuracy of 77.2%, which was an improvement of 12.67% over the baseline model. When extended to classify in distribution and out of distribution samples with the use of K-Means clustering on the features from the output layer, the new model achieved 95.3% accuracy.

1. Methodology

A network architecture with three convolutional layers followed by two linear layers was chosen. This architectures includes a dropout layer for the hidden linear layer to reduce overfitting, as well as batch normalization and ReLU activation layers. Max pooling was used to reduce the complexity of the model, facilitating shorter training time and condensing local features. The full network architecture with batch normalization included can bee seen in Listing 1. The base model that was improved upon is similar, but without the batch normalization layers. The Adam optimizer was selected since it tends to perform well by adapting the learning rate during the training.

The methods used for improving the base model will be detailed in the following sections.

1.1. Pre-processing

The pre-processing method that was chosen for this task was standardization of the datasets, performed with the mean and standard deviation of the pixel values in the images from the training dataset. This approach is beneficial as it reduces the variance in the input data and makes the model more robust to uniform changes, such as variations in image brightness. We expect to see improved accuracy when pre-processing is performed.

Listing 1. Architecture

```
Conv2d(3, 20, kernel_size=4, padding=2),
MaxPool2d(2, 2),
BatchNorm2d(20),
ReLU(),
Conv2d(20, 40, kernel_size=4, padding=2),
MaxPool2d(2, 2),
BatchNorm2d(40),
ReLU(),
Conv2d(40, 60, kernel_size=4, padding=2),
MaxPool2d(2, 2),
BatchNorm2d(60),
ReLU(),
Flatten(),
Linear(60*5*5, 50),
BatchNorm1d(50),
Dropout(0.5),
ReLU(),
Linear(50, 12)
```

1.2. Batch Normalization

Batch normalization is a technique that, like pre-processing, standardizes the inputs, but instead of applying it directly to the input to the network, this is performed over the output from the preceding layer from where the batch normalization layer is located.

It is useful for similar reasons: to make the network more robust. However, it also includes learnable parameters that allows for the network to shift and resize the scale of the features. The expected improvements is to reduce overfitting and make the training converge to a solution faster.

1.3. Early Stopping

Early stopping is useful to both decrease training time and overfitting. If a model is trained for too long, then it might start to memorize the training data and lose the ability to generalize to unseen data. However, if we do not train the model for long enough, then the model will underfit the data. Instead of guessing the optimal number of epochs, we can create an algorithm that determines when the training should stop. The early stopping method that was used

looks at the evaluation loss of each epoch and compares it to the minimum loss achieved. If a user-defined number of epochs transpires and the loss does not go below the adaptive threshold in that duration, then the training stops early.

1.4. Activation Functions

In addition to the ReLU activation function, a new model was tested using the soft plus activation function. In theory, the soft plus function should be better than ReLU since it has similar characteristics but does not have the "dying ReLU" problem. However, the initial model that utilized ReLU was empirically proven to be slightly better, so this experiment was discontinued and the previous three methods will be selected for further analysis.

1.5. Extending the CNN to classify Id and Ood samples

Since the CNN architecture does not have a layer on the output that obscures the information, such as a softmax layer, the output vector from the network can be directly used for the classification task of in distribution (Id) and out of distribution (Ood) images. The hope was for the two classes to have very different distributions, and thereby we can separate them by further modeling this twelve dimensional vector space.

To perform this task, K-means clustering was used, with k set to 2 clusters. This algorithm is trained on only the outputs generated from predicting the in distribution and out of distribution samples with the CNN model. It will find the mean of two clusters in the data that minimizes the distance between each cluster center and each point that belongs to that cluster. It is a fairly simple, but effective method to perform unsupervised learning in the case where the distributions are thought to be well-separated.

2. Experimental Evaluation

The setup of the experimental evaluation and the results will be described in the following sections.

2.1. Experimental Setup

The models were trained with the training dataset in batches of 32. The Adam optimizer was used with an initial learning rate of 10^{-3} because of its adaptive learning rate, which should make it easier for the model to find the local optimum of the feature landscape. Momentum was not considered for this project. The cross entropy loss function from pytorch was used, which applies the log-softmax function to the output of the model and then calculates the negative log likelihood loss. A maximum of 50 epochs was used for training, since the evaluation score in the training history seemed to plateau slightly before this on the base model. The training data was split into two datasets with a 90%/10% ratio, one used for training, and the other one used for evaluation during training. All models were evaluated on the test set after the training was complete.

2.2. Improvements

In this section, the improvements will be briefly detailed.

2.2.1 Pre-processing detailed

To perform the standardization of the datasets, the three color channels of the training dataset were separated, and their means and standard deviations were calculated. Before standardizing, we can see an approximately normal distribution of the pixel values in the 0 to 255 range, with a mean around 140, see Figure 1.

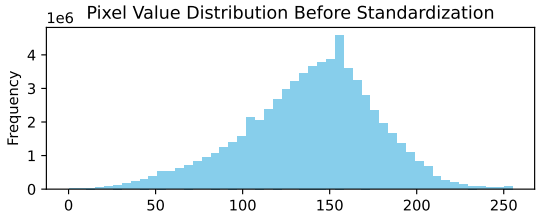

Figure 1. Pixel value distribution of training data before standardization

After obtaining the statistics, all pixel values in each datasets were standardized with those values, see Figure 2.

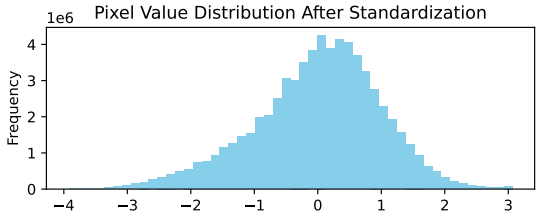

Figure 2. Pixel value distribution of training data after standardization with standard deviation [36.55, 37.87, 39.76] and mean [143.47, 139.37, 133.09] for color channels: red, green, and blue, respectively.

2.2.2 Early stopping detailed

Early stopping was used to stop the training after 6 successive epochs were processed without any decrease in minimum loss. Without early stopping, we can see that the model was trained for the full 50 epochs, when the increase in evaluation accuracy plateaued much earlier on, see Figure 3.

With early stopping enabled, the training stopped much earlier, effectively halving the training time while increasing the accuracy, see Figure 4.

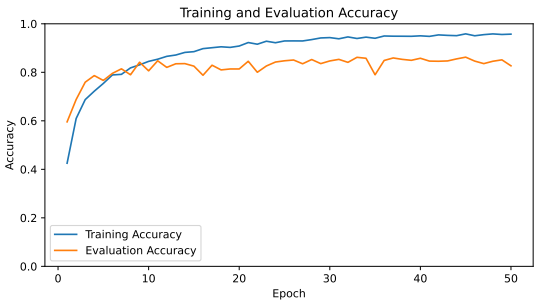

Figure 3. Training history of final model, excluding early stopping

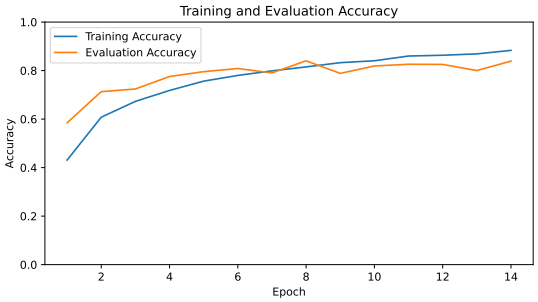

Figure 4. Training history of final model

2.2.3 Batch Normalization detailed

Batch normalization did little effect on the final result of the model. However, based on the comparison between the the training accuracy curves in Figures 4 and 5, we can tell that it took slightly longer for the model without batch normalization to learn the training data, passing 60% evaluation accuracy only after 4 epochs, as opposed to 1 epoch.

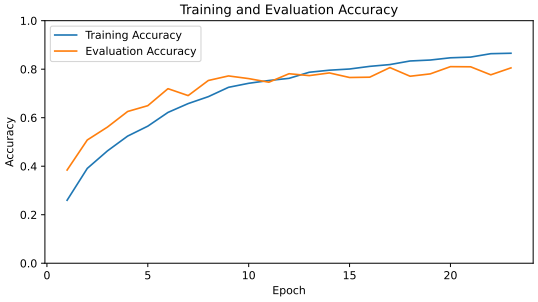

Figure 5. Training history of final model, excluding batch normalization

2.3. Ablation study results

In this section, the results of the ablation study will be detailed.

2.3.1 Accuracy

The best results were achieved with the final model that utilized all of the improvement methods, with a test-set evaluation accuracy of 77.2%, see Table 1.

Model.No.	Model 1	Model 2	Model 3
Accuracy	0.7720	0.3814	0.7457
Model.No	Model 4	Model 5	
Accuracy	0.7237	0.6453	

Table 1. Accuracy of the different models, evaluated on the test set. Model 1: Final model; Model 2: Excluding pre-processing; Model 3: Excluding early stopping; Model 4: Excluding Batch normalization; Model 5: Base model

The results of the model that did not make use of data pre-processing is a clear outlier, only achieving 38.1% accuracy. It needs to be noted that the true accuracy of this model is inconclusive, since when running it without early stopping, evaluation accuracy on the evaluation set fluctuates erratically from 30% to 85% while the model starts to overfit to the data, see Figure 6. The highest recorded test set accuracy for this model without early stopping was 68.8%,

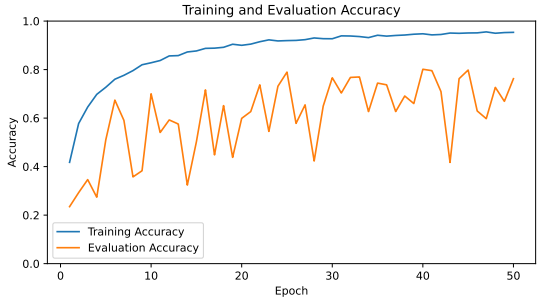

Figure 6. Training history of final model, excluding pre-processing

2.3.2 Confusion Matrices

We can observe from the confusion matrices, which display the predictions on the test set for each model, that all models except for the the one excluding pre-processing exhibit a clear diagonal pattern. This indicates that the models are accurately predicting the majority of the test samples, see Figures 7, 8, 9, 10, and 11.

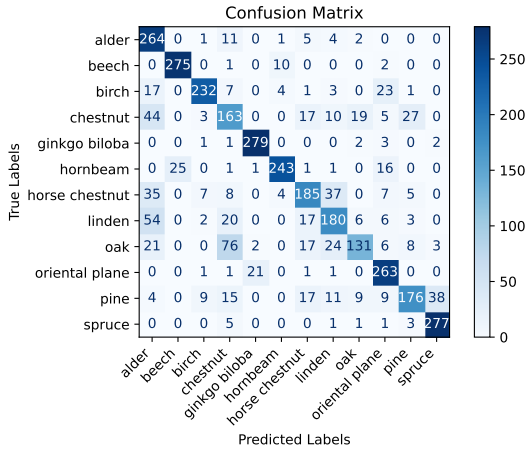

Figure 7. Confusion matrix of final model

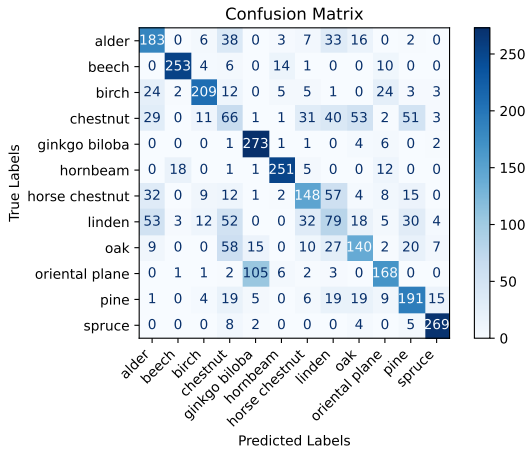

Figure 8. Confusion matrix of base model, without any of the improvements

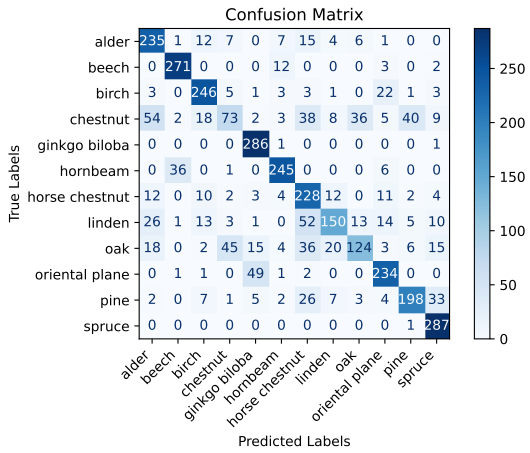

Figure 9. Confusion matrix of full model, excluding early stopping

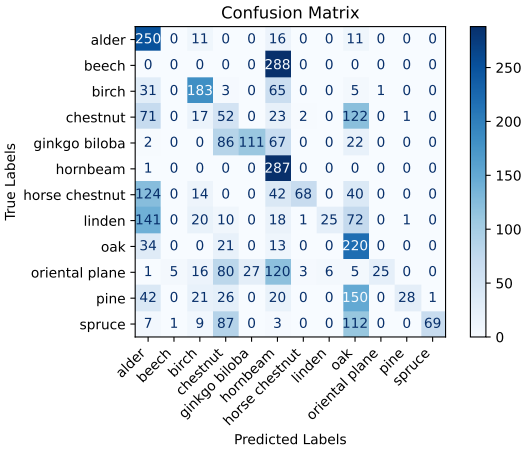

Figure 10. Confusion matrix of full model, excluding image preprocessing

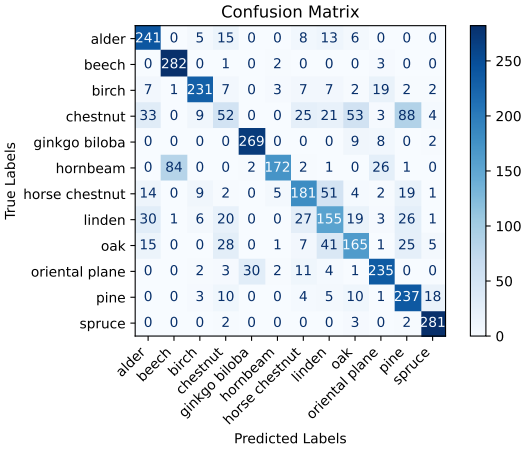

Figure 11. Confusion matrix of full model, excluding batch-normalization

2.3.3 Visualization of predictions

Finally, we can display a few of the images that the model predicted correctly and incorrectly on, see Figure 12. Sadly, it is not immediately clear from the images what the major difference is that made the model predict incorrectly on them, especially in the case of the beech class. It appears as if the major difference there is that the bad prediction is slightly greener than the other.

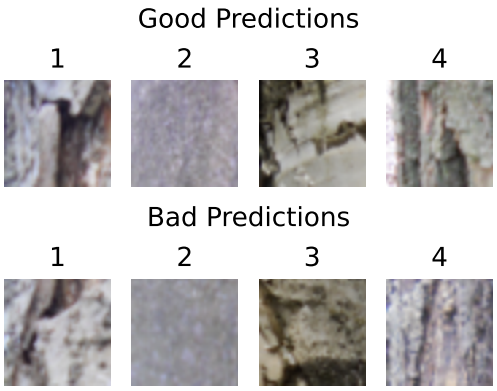

Figure 12. Four image pairs from different classes that the final model predicted correctly and incorrectly on. The numbers above the images are the class labels that the images correspond to: 'alder', 'beech', 'birch', and 'chestnut', respectively.

2.3.4 Results of expanding CNN to classify Id and Ood images

The K-Means clustering algorithm managed to learn the distributions of the two classes from the outputs of the CNN quite well, achieving an impressive 95.3% accuracy on the 2000 images. The histograms of the bad predictions can be seen in Figure 13.

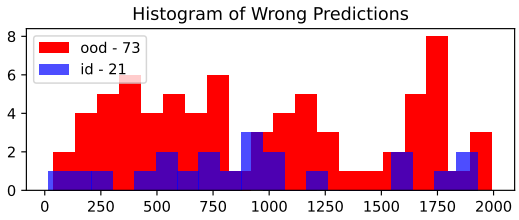

Figure 13. Histogram of predictions on id and ood samples

We can inspect the wrong predictions to try to figure out why this happened, see Figure 14. It is possible to see from the fourth example from the bad in-distribution predictions that the crevice in the wood could look like the intersection of a bean, and that the second example in the bad out-of-distribution predictions might look a bit like the trunk of a tree. However, from the rest of the images it is more difficult to tell why the model got the classes of those samples wrong. Another reason could be that some of the images of the beans are quite zoomed in, which might not make it look very different from a tree trunk.

Figure 14. Visualization of four images that were predicted incorrectly from the in-distribution and out-of-distribution samples.

We can also use PCA to perform dimensionality reduction on the data to attempt to view the clusters in 2D, see Figure 15. We can tell from the plot that two clusters are fairly separated, which explains in part why we got such good results.

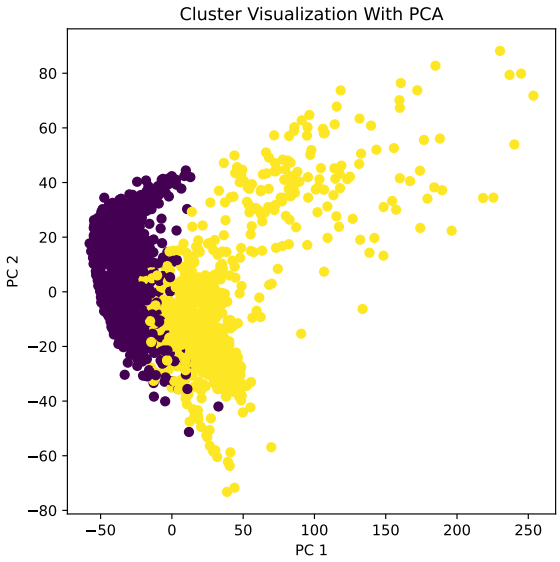

Figure 15. Visualization of the two clusters of data, made with PCA.

2.3.5 Conclusion

It seems that all the improvements achieved what was expected from them. Pre-processing made it easier for the model to learn the features and better generalize to the test data. Batch normalization accelerated the training, and might have had a slight impact on overfitting. Early stopping decreased overfitting and generally increased accuracy when the training was stable. The extended model was also quite able to differentiate between in-distribution and out-of-sample distribution samples.