

# Using Deep Learning to Determine Honeysuckle Bark

Brett Huffman - CSCI 5390 - Main Project Phase VII

## **Abstract**

The objective of this project was to build a deep-learning model capable of spotting several species of invasive Honeysuckle in the wild. The model has been trained to properly differentiate between Honeysuckle and the many forest plants found in the Illinois/Missouri habitat.

## **1 Overall Problem To Be Solved**

The Engineering and Biology Departments at Principia College are teaming up to build an autonomous rover that will poison unwanted species of plants.

After a year of work, they have demonstrated the ability to maneuver around a space, then when manually activated, chemically treat an unwanted plant.

The Biology department has identified a herbicide that is only poisonous to Honeysuckle – the main plant which they want to eradicate.

The problem with the herbicide is that it must be delivered into the stem. Thus, to treat a plant, the rover lowers a grinder boom which takes some of the bark off the plant. Next, a few drops of the herbicide is sprayed into the plant. Correctly applied, the plant dies within days ([Web20]).

The last big problem for the team to solve is how to autonomously determine if the plant is a target Honeysuckle.

This project is an attempt to see if the species of plant can be accurately identified from other plants in the target area.

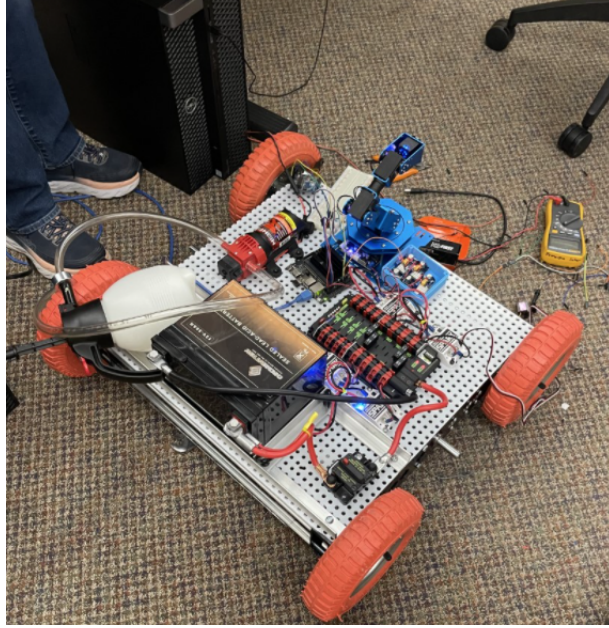


Figure 1: A view of the early rover prototype

## 1.1 Honeysuckle

Honeysuckle is an invasive species brought into the United States in the early 1900's as an ornamental plant. It has been used for erosion control, but quickly became invasive to many other species of native plants. It invades areas that have been disturbed, such as forest fire scorched areas and flood plains. It rapidly out competes native plants for nutrients and sunshine ([Wik22]).

Further, Honeysuckle produces a thick canopy that prevents sunlight from getting to lower levels of the forest and effectively chokes off new growth.

For these reasons, eradication of the honeysuckle in wild areas is an important goal for botanists ([oC20]).

## 2 Phase 1 Data Preparation

The main purpose of this phase was to select a project and acquire the proper amount of data for testing and validation. Participants needed close to 1000 images to perform training,

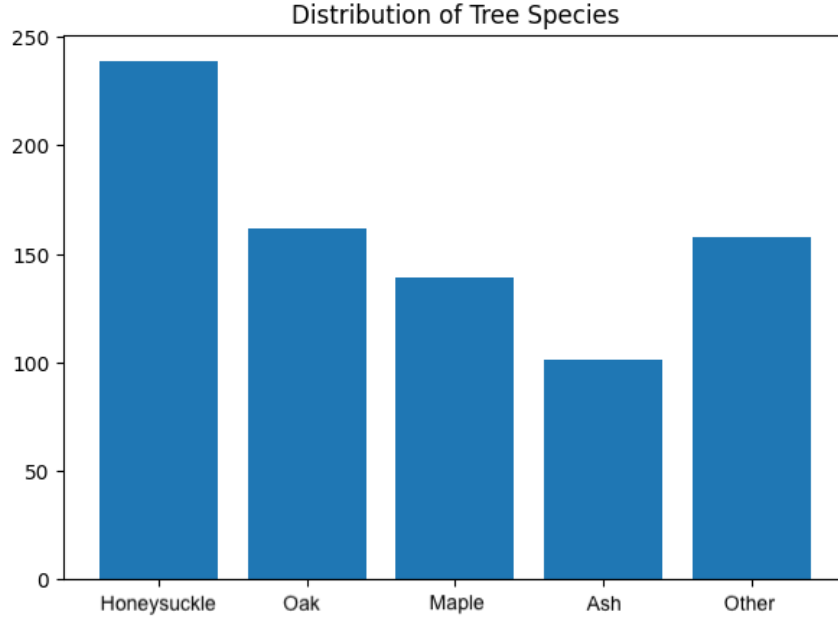


Figure 2: Initial Species Image Distribution

validation and testing.

The data was accumulated through several photo sessions through the Illinois countryside. They were taken at various times of the year: late Winter, and early and late Spring. This should yield a good variety of natural foliage conditions that the rover might encounter.

One of the big challenges was to make sure all images were a large variety of resolution, angles and colors. At one point, later in the project, it was discovered that many of the Honeysuckle were black and white images. These images were removed so as to not improperly influence the training by one class of results focusing solely on black and white images.

In the end, all images were standardized to 224 x 224 x 3 colors. This turned out to be the best size as all transfer learning models in Phase 6 utilized this image size.

### 3 Phase 2 Data Preparation

In Phase 2, the objective was to build a convolutional neural network that would overfit our dataset with the smallest number of parameters. Additionally, we built a model that would

## Model 2 Listing

Layer ( <b>type</b> )	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
sequential (Sequential)	(None, 150, 150, 3)	0
conv2d (Conv2D)	(None, 148, 148, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 37, 37, 64)	0
conv2d_1 (Conv2D)	(None, 35, 35, 32)	18464
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 32)	0
conv2d_2 (Conv2D)	(None, 6, 6, 16)	4624
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 10)	5770
dense_1 (Dense)	(None, 5)	55

Total params: 30,705

Trainable params: 30,705

Non-trainable params: 0

-----

overfit immediately by feeding the output as an additional input channel.

After much trial-and-error, the following model was developed. It had only 30k parameters, which seemed to be considerably smaller than all other models. In addition, the model was able to achieve 99.5% accuracy.

This “Model 2” would act as a basis for all the models built later in the course.

In addition to the CNN size study, we learned to check that a model was wired correctly by routing the output as a model input. This caused the model to immediately overfit as seen in figure 3.

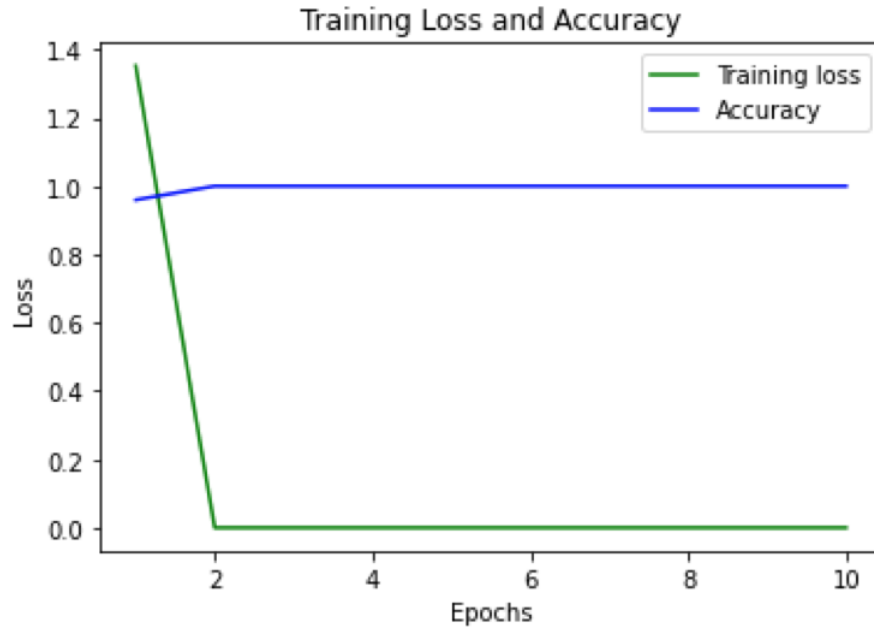


Figure 3: Output As Input Model Training Accuracy and Loss

## 4 Phase 3 Data Preparation

Phase 3 had us splitting our data into training, validation, and test sets. This supported our objective to obtain the very highest accuracy possible.

To obtain the results desired, the data was randomly split 70% for Training, 15% for Test, 15% for Validation. Next, all the images were moved into their respective validation, training and test folders. Finally, three Data Generators were developed using the “flow\_from\_directory” functionality which automatically built test classes for the project.

This part of the project turned out to be the most challenging coding part of the class. The images were not in the proper shape for my model to consume. It took really learning how Numpy manipulation works to solve this problem.

In this phase, EarlyStopping and ModelCheckpointing was also introduced. At this point, accuracy was still quite low (59%) and loss was quite high: .41 (as seen in figure 4). It wasn’t until later phases that the model was able to achieve quite impressive results.

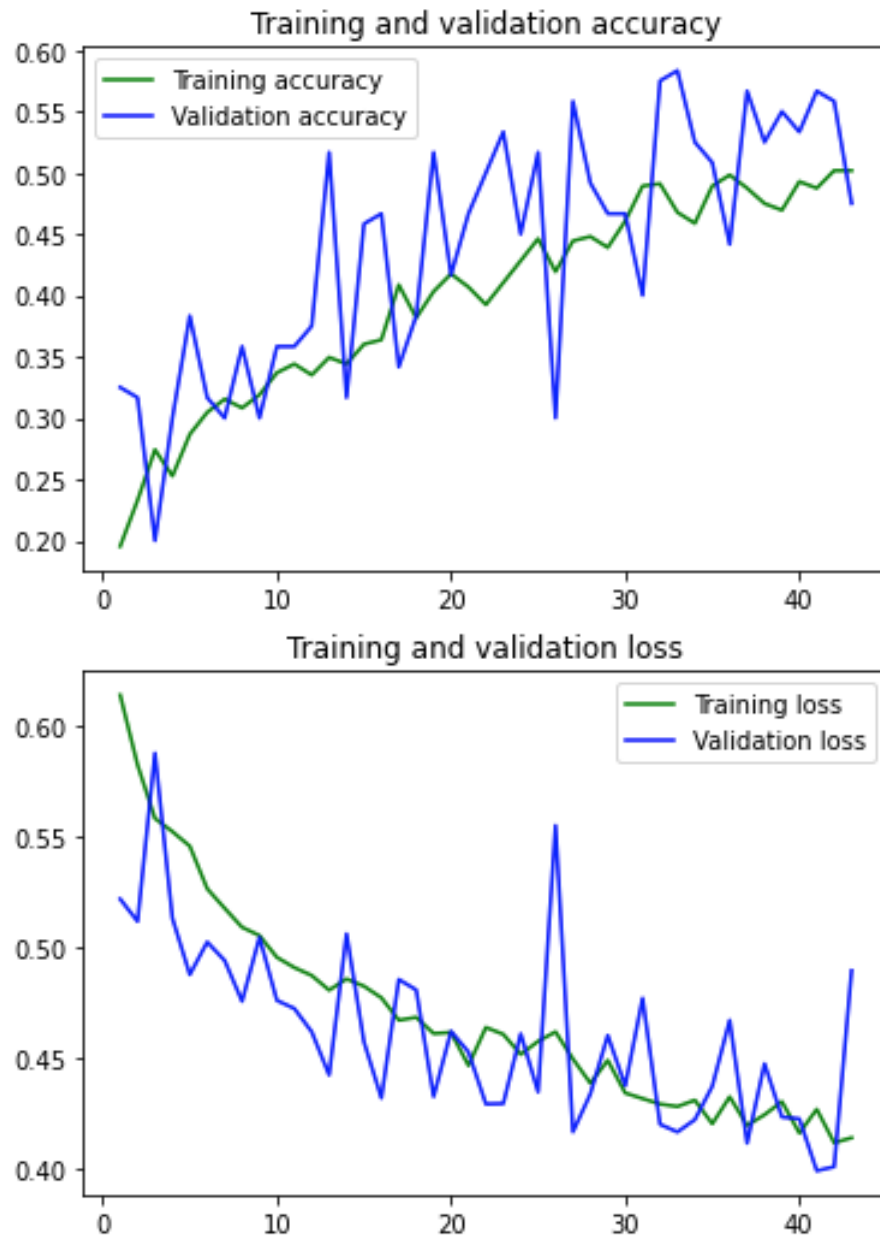


Figure 4: Phase 3 Best Validation Accuracy and Loss

## 5 Phase 4 Data Preparation

In this phase of the project, another 400 images were added. Additionally, the project was changed to a binary classification. Not only did these two changes dramatically improve the results of the study, they also better aligned with the project intent. The project was supposed to tell the difference between Honeysuckle and all other forest plants, not necessarily classify all the different types.

The project goal was to add Data Augmentation to the project. This was rather simple as the model was already setup as a multi-input, multi-output functional model. I only needed to add the `data_augmentation` feature and make sure the Image Generators were normalizing the data.

With all these changes, model performance went through the roof. It suddenly achieved:

Accuracy: 0.91

Precision: 0.94

Recall: 0.9

F1: 0.91

## 6 Phase 5 Effects of Regularization

The most dramatic phase of the entire project had to do with setting up Regularization in the project. Focus was concentrated on three Regularization techniques:

- Batch Normalization;
- Dropout;
- L1 and L2 Regularization

After establishing a baseline, each Regularization technique was accomplished against the model in an attempt to find the best parameters of each.

The impact of all Regularization was dramatic. It often resulted in longer training times, but much better accuracy and lower loss. The most successful techniques are shown in table 1.

Value Studied	Training Notes	Visualization
Batch Normalization added to all layers except dense	<ul style="list-style-type: none"> <li>• Slower than normal training time</li> <li>• Trained in 109 epochs</li> <li>• Reached 94% Accuracy</li> </ul>	
Conv Layer Dropout = .5 and Dense Layer Dropout = .5	<ul style="list-style-type: none"> <li>• Much slower than normal training time</li> <li>• Trained in 127 epochs</li> <li>• Reached 94% Accuracy</li> </ul>	



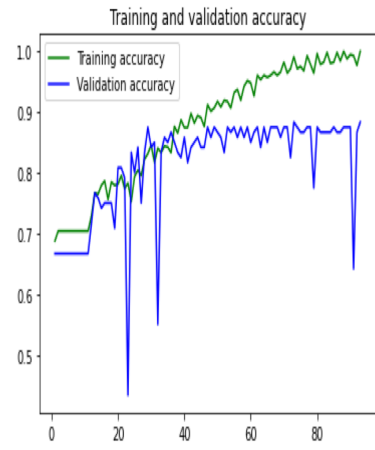
Value Studied	Training Notes	Visualization
L1 Regularization = .002	<ul style="list-style-type: none"> <li>• Slower than normal training time</li> <li>• Trained in 93 epochs</li> <li>• Reached 88% Accuracy</li> </ul>	 <p>Training and validation accuracy</p> <p>Training accuracy (green line) and Validation accuracy (blue line) over 93 epochs. Training accuracy starts at ~0.7, rises to ~0.95 by epoch 93. Validation accuracy starts at ~0.68, rises to ~0.88 by epoch 93, with significant fluctuations.</p>
L2 Regularization = .001	<ul style="list-style-type: none"> <li>• Slower than normal training time</li> <li>• Trained in 180 epochs</li> <li>• Reached 94% Accuracy</li> </ul>	 <p>Training and validation accuracy</p> <p>Training accuracy (green line) and Validation accuracy (blue line) over 180 epochs. Training accuracy starts at ~0.7, rises to ~0.98 by epoch 180. Validation accuracy starts at ~0.68, rises to ~0.94 by epoch 180, with significant fluctuations.</p>

Table 1: Dropout Regularization Analysis

As a final step in exploration of Regularization, the best model from the prior phases was changed to include Batch Normalization, two Dropout layers with 0.5, and an L2 Regularization of 0.002.

The model yielded the highest accuracy of the entire semester. Accuracy was 95% trained over 200 epochs. Final validation loss was .28.

## 7 Phase 6 Effects of Regularization

In the final phase of the project several pretrained models are evaluated against the baseline model. Pretrained model's classifiers were systematically placed on top of the dense layers of the existing model.

The pretrained models studied were:

- VGG16;
- DenseNet121;
- and TensorHub Plant Model V1

After much analysis, it was concluded that the VGG16 and Plant Model V1 actually had a negative impact on the model accuracy. In both cases, it proved nearly hopeless to get accuracy levels above 71%. This was nearly a 30% drop in accuracy from our best model in Phase 5.

However, the DenseNet121 model was quite different. It took quite a while to train, but once complete, it achieved the highest accuracy of any model in the study: 96.8%. Maybe the most interesting thing is that it obtained these excellent results in just 46 epochs (See table 2).

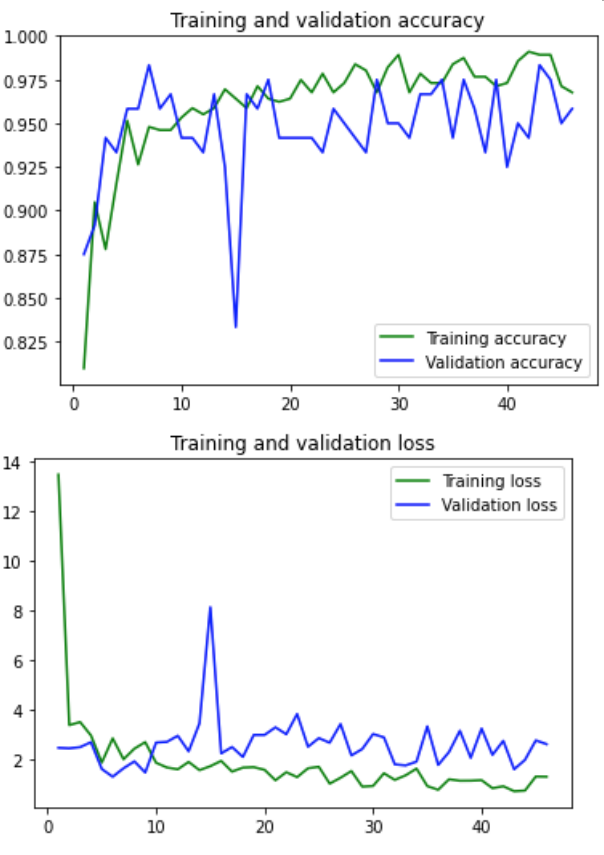
VGG16 Results	Visualization
<ul style="list-style-type: none"> <li>• Accuracy: 0.968</li> <li>• Precision: 1.0</li> <li>• Recall: 0.9375</li> <li>• F1: 0.9667</li> </ul>	 <p>The visualization consists of two line graphs. The top graph, titled 'Training and validation accuracy', plots accuracy from 0.825 to 1.000 over 45 epochs. The training accuracy (green line) starts at approximately 0.825 and rises to about 0.975 by epoch 10, then fluctuates between 0.95 and 0.99. The validation accuracy (blue line) starts at approximately 0.875, peaks at 0.98 around epoch 5, drops sharply to 0.83 at epoch 15, and then fluctuates between 0.92 and 0.98. The bottom graph, titled 'Training and validation loss', plots loss from 2 to 14 over 45 epochs. The training loss (green line) starts at approximately 13.5 and drops sharply to about 2.5 by epoch 5, then fluctuates between 1.5 and 2.5. The validation loss (blue line) starts at approximately 2.5, peaks at 8 around epoch 15, and then fluctuates between 2 and 4.</p>

Table 2: DenseNet121 Results

## 8 Conclusion

The semester-long course project that was created will act as the first-generation detector for the Honeysuckle eradication robot. I'm sure that many more iterations will need to be developed, but this will act as a good prototype with which the team can work.

More importantly, the project was an amazing adventure into Convolutional Networks, Tensorflow/Keras, benchmarking, and modern deep learning methods. Using data augmentation, regularization and transfer learning, we were able to produce an accurate, first-class model that is critical in understanding of how to build deep learning machines in the future.

## References

- [oC20] Missouri Department of Conservation. Bush honeysuckle control. <https://mdc.mo.gov/trees-plants/invasive-plants/bush-honeysuckle-control>, 2020.
- [Web20] Integrated Pest Management Website. Weed of the month: Bush honeysuckle—an ornamental gone wrong. <https://ipm.missouri.edu/ipcm/2015/9/Weed-of-the-Month-Bush-honeysuckle-an-ornamental-gone-wrong/>, 2020.
- [Wik22] Wikipedia. *Lonicera japonica*. [https://en.wikipedia.org/wiki/Lonicera\\_japonica](https://en.wikipedia.org/wiki/Lonicera_japonica), 2022. Invasive Honeysuckle Species Description.