Using Deep Learning to Determine Honeysuckle Bark

Brett Huffman - CSCI 5390 - Main Project Phase II

Abstract

The objective of this project is to build a convolutional neural network which can accurately classify based on a learned images set.

This specific project will build a model capable of spotting several species of invasive Honeysuckle in the wild. The model will try to determine which species is invasive out of the many desired forest plants in the Illinois/Missouri habitat.

1 Phase 2 Objective

There multiple objectives for Phase 2.

- The first is to build a Convolution Neural Network that delivers near to 100% accuracy.
- Once an accurate model was created, the model's Conv2D and Dense layers were paired down to try to discover the smallest network and it's impact on performance (accuracy, precision, and recall).
- Finally, Graduate Students were to try out there model with the output being provided as the input.

The results of all these tests are discussed in this paper.

2 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.

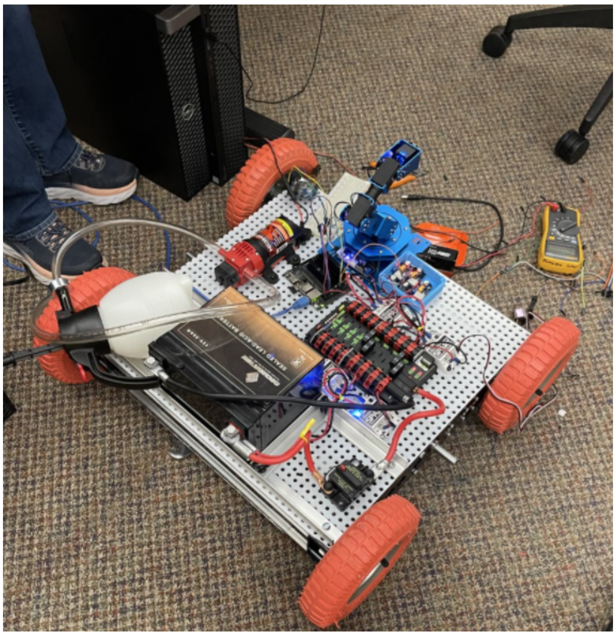

Figure 1: A view of the early rover prototype

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.

2.1 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]).

3 Phase II Procedure and Results

The first thing needed for Phase II was to setup the Convolutional Neural Network for training. This involved changing the code from Phase I to support using a generator for training dataset. The code for this generator is shown below:

```
# Generator for training dataset
train_dataset = my_generator.flow_from_directory(
    './CSCI_5390/MainProject/Images', target_size=(150, 150),
    batch_size=40, class_mode='categorical')
```

After a substantial research to make sure the generator was working as required, the recommended model was built (See figure). Additionally, Data Augmentation was used to supply fresh images to the network.

Many models were tested, but three seemed to show the best results and allowed for good conclusions to be made.

```
Model 1 Listing
_________________________________________________________________
 Layer (type)                  Output Shape               Param #
=================================================================
 input_1 (InputLayer)          [(None, 150, 150, 3)]      0
 sequential (Sequential)       (None, 150, 150, 3)        0
 conv2d (Conv2D)               (None, 148, 148, 32)       896
 m_pooling2d MaxPooling2D      (None, 74, 74, 32)       0
 conv2d_1 (Conv2D)             (None, 72, 72, 64)         18496
 m_pooling2d2 MaxPooling2D     (None, 36, 36, 64)       0
 conv2d_2 (Conv2D)             (None, 34, 34, 128)        73856
 flatten (Flatten)             (None, 147968)             0
 dense (Dense)                 (None, 5)                  739845
=================================================================
Total params: 833,093
Trainable params: 833,093
Non-trainable params: 0
```

3.1 Model 1

The 1st model tested is shown in Model 1 Listing.

This model eventually produced near-perfect results on the training data. However, it took much longer to achieve than other models tested later in the project.

Overfitting was definitely achieved by Model 1 with a final accuracy of 99.7%. This was achieved in 927 Epochs.

One thing to note is that the Convolution layers in this network were inverted (smallest at the top). This is how one model is shown on Page 216 of the textbook ([Cho21]), but is different than as described in class.

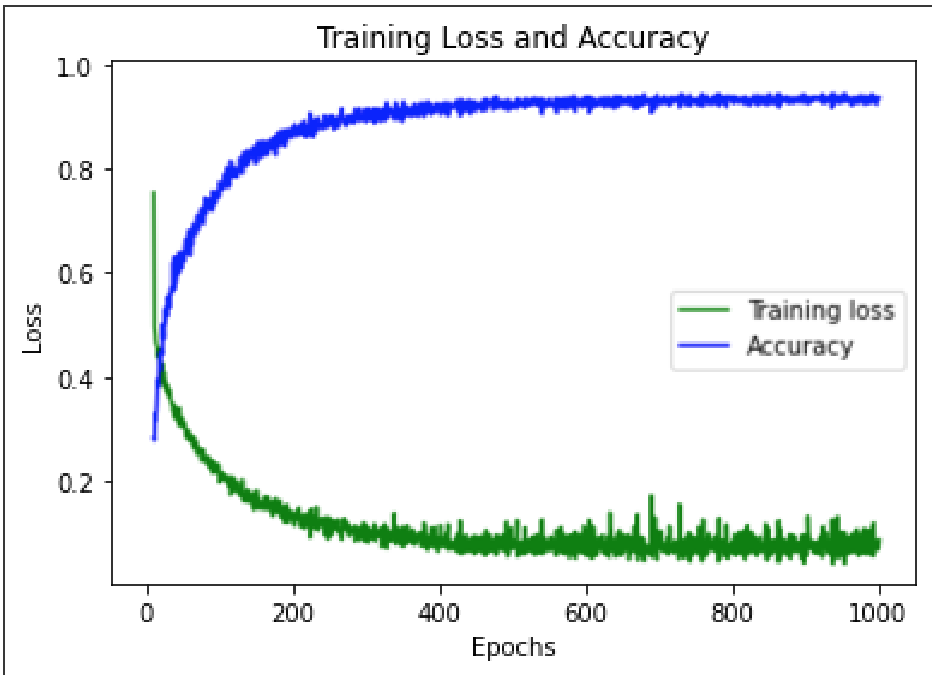

Figure 2: Model 1 Training Accuracy and Loss

3.2 Model 2

This second set of models were closer to the suggestion in the project summary. They had Conv2D layers of 64 / 32 / 16 with one dense layer.

This model shown in Model 2 Listing was far more successful than Model 1. It achieved near-perfect accuracy in about 600 Epochs. So it overfit on the training data much faster. Additionally, it needed many less parameters than Model 1. Model 1 needed 833K parameters to Model 2's 30K. That is a significant downsizing in both system requirements and training speed.

3.3 Model 3

This third model was an almost exact match of Model 2 except the first Dense layer was removed.

Model 3 (and it's many derivations) was also able to overfit within 1000 Epoch's, but it's performance was not nearly as impressive. It achieved 99.5% accuracy, but it did it in 990

```
Model 2 Listing
 Layer (type)                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 (MaxPooling  (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
_________________________________________________________________
```

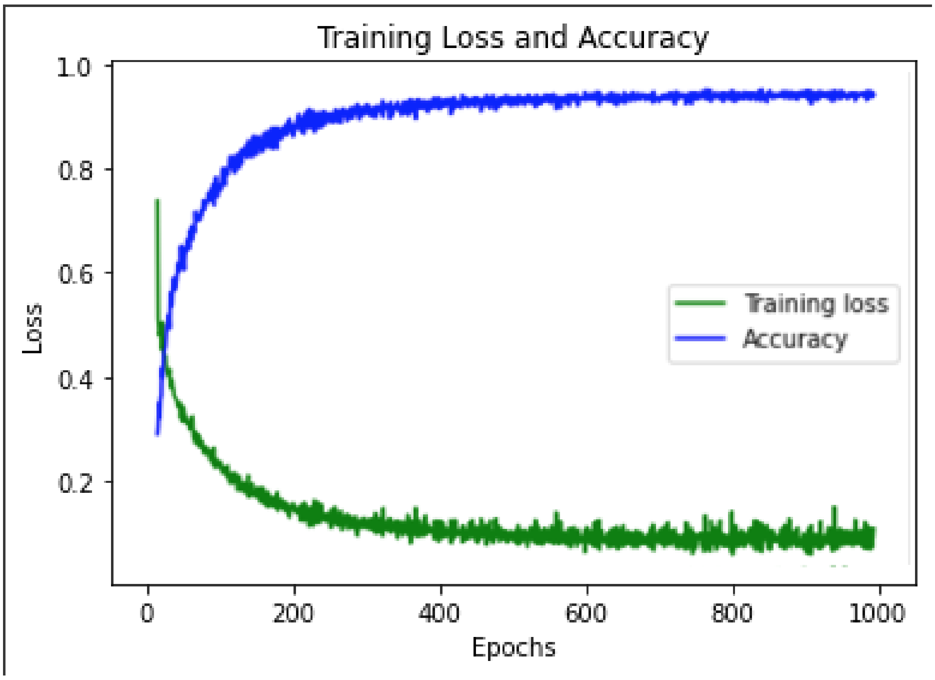

Figure 3: Model 2 Training Accuracy and Loss

Epochs. Additionally, it seemed to train without the immediate jump in performance. All through the training process, it seemed unsure if it could learn enough of the data.

4 Graduate Student Requirement Code

For the Graduate Student portion of the project, a completely separate Google Colab project was made. This was because the Image Generator was to be changed so substantially.

In the end, the Generator was changed to use the imageDatasetFromDirectory functionality as it more easily allowed another dimension to the input tensor.

The model that seemed to work best was a very simple Convolution Network as seen in Model OutputAsInput Listing.

A couple of interesting things about this. First, the model has over 3.5M parameters! That is much higher than would have been imagined.

Additionally, the model quickly went to 100% accuracy, as would be expected.

Model 3 Listing

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 150, 150, 3)]     0
 sequential_2 (Sequential)   (None, 150, 150, 3)       0
 conv2d_6 (Conv2D)           (None, 148, 148, 64)      1792
 max_pooling2d_4 (MaxPooling  (None, 37, 37, 64)       0
 conv2d_7 (Conv2D)           (None, 35, 35, 32)        18464
 max_pooling2d_5 (MaxPooling  (None, 8, 8, 32)         0
 conv2d_8 (Conv2D)           (None, 6, 6, 16)          4624
 flatten_2 (Flatten)         (None, 576)               0
 dropout (Dropout)           (None, 576)               0
 dense_3 (Dense)             (None, 5)                 2885
=================================================================
Total params: 27,765
Trainable params: 27,765
Non-trainable params: 0
_________________________________________________________________
```

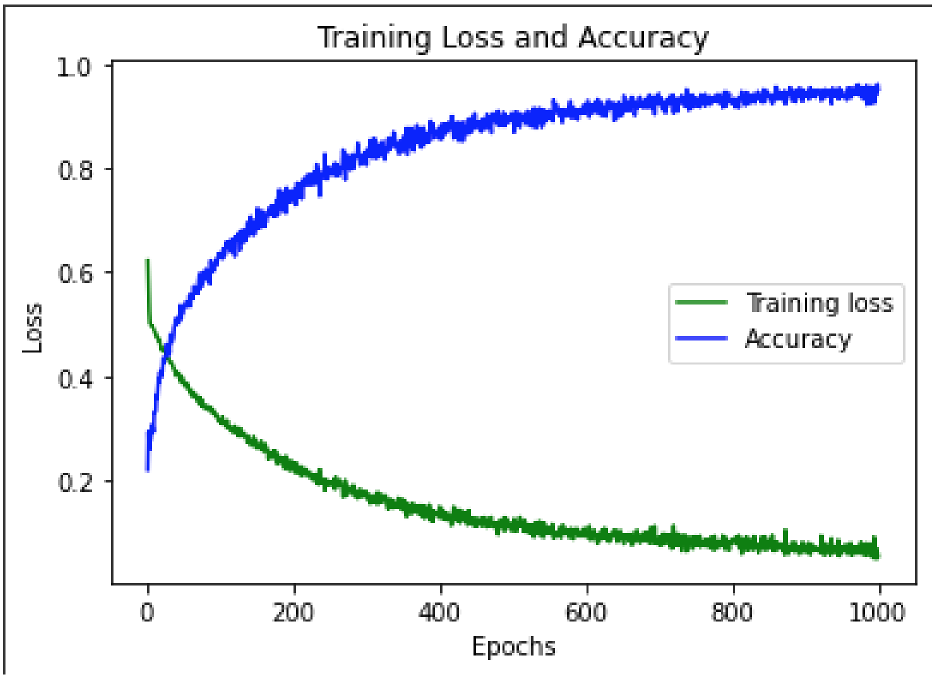

Figure 4: Model 3 Training Accuracy and Loss

```
Model OutputAsInput Listing
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 150, 150, 4)]     0
 conv2d_11 (Conv2D)          (None, 148, 148, 32)      1184
 flatten_5 (Flatten)         (None, 700928)            0
 dense_9 (Dense)             (None, 5)                 3504645
=================================================================
Total params: 3,505,829
Trainable params: 3,505,829
Non-trainable params: 0
_________________________________________________________________
```

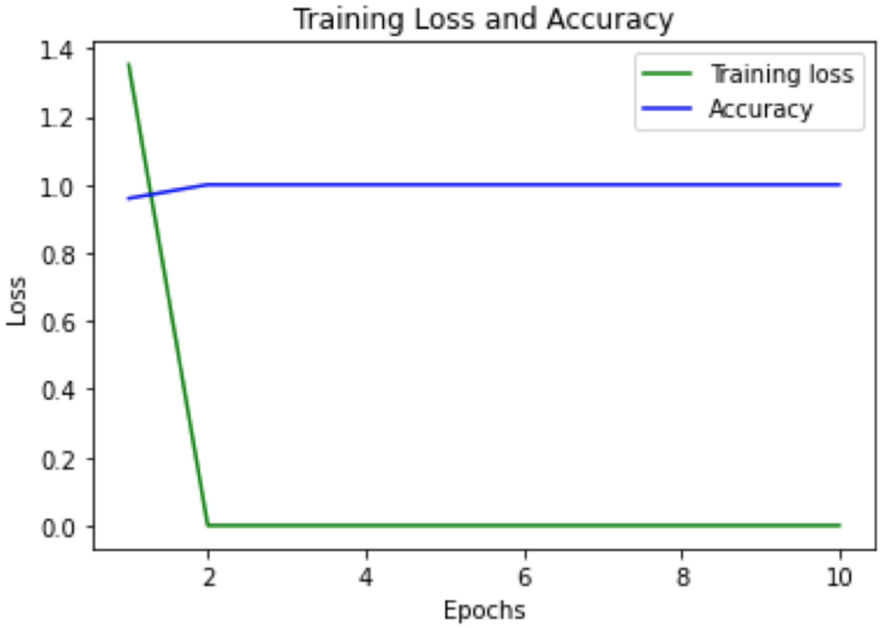

Figure 5: Output As Input Model Training Accuracy and Loss

References

[Cho21] Francois Chollet. *Deep learning with Python (2nd ed.)*. Manning Publications, 2021.

[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`, 2022. Invasive Honeysuckle Species Description.