Classifying Tomato Leaf Health

Tyler Osterberg

Regis University

MSDS 686: Deep Learning

12/11/2022

Classifying Tomato Leaf Health

Agriculture, and specifically, crop health monitoring is becoming an interesting class of problems for machine learning and data science. These problems tend to involve computer vision, and classification of images that are remarkably similar, and the models are likely to be deployed on resource-constrained devices. This could mean limited, to no, internet connectivity or RAM measured in megabytes or kilobytes. The combination of computer vision and small deployment models drew me to choose a Kaggle dataset for classifying diseases from photographs of the tomato plants' leaves. I chose a dataset with a few current code samples on it so that my contribution could be more meaningful.

Overview

The dataset chosen for this project was the "Tomato Leaves Dataset" published by Ashish Motwani. This dataset provides images of tomato leaves for classification into a range of classes of different diseases and healthy tomato plants. In addition, the brief for this dataset specifies a goal of developing a lightweight model that can predict tomato leaf disease that can be deployed to an offline mobile application. From this dataset and description of the problem, I decided to create a higher accuracy small model for limited memory and offline mobile devices. This means that the bulk of the work would be in trying to make a small model more accurate. My strategy was to see what a small model's accuracy would be when trained on the data, then test if data augmentation improved that model. After that, I would attempt to generate a highly accurate model using transfer learning from a pretrained model and well-known architecture. Assuming I get a higher accuracy model from the transfer learning model, I will try to make a more accurate small model using a process called knowledge distillation.

Description of the Data

The tomato leaf dataset is a library of images with 25,848 in its training folder, and 6,683 in its test folder. These images were bucketed into their classifications folder, of which there were 11 classes. These are: bacterial spot, early blight, late blight, leaf mold, powdery mildew, Septoria leaf spot, spider mites, target spot, tomato mosaic virus, tomato yellow leaf curl virus, and healthy. The images of the leaves come from both natural and lab environments, so we should have a few diverse backgrounds present in our dataset. The counts across our categories are also balanced, with a smaller representation of powdery mildew but still with many examples.

Methods

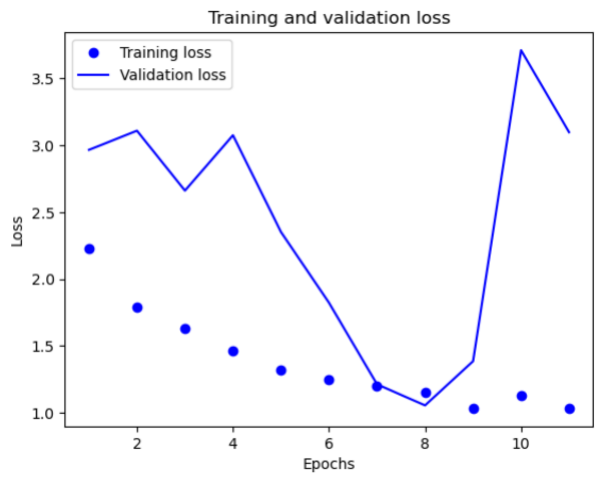
The first method that we applied was a normalization of our images color channels to a range between 1 and 255. This would limit the range of values that could be fed to the model and skew the results. The second method was to resize our images so that they were all the same size going into the model. We found that the mode size of our images was 256 by 256 and chose to use that as our target size for images being input. Third, we added data augmentation, randomly rotating, shifting, shearing, zooming, and shifting the input images both to increase the number of available training images but also to avoid overfitting on well centered tomato leaf images. Finally, we used a method of training called knowledge distillation which takes two models as input for training, with one large pre-trained model acting as a teacher and the output model acting as the student whose accuracy we want to improve.

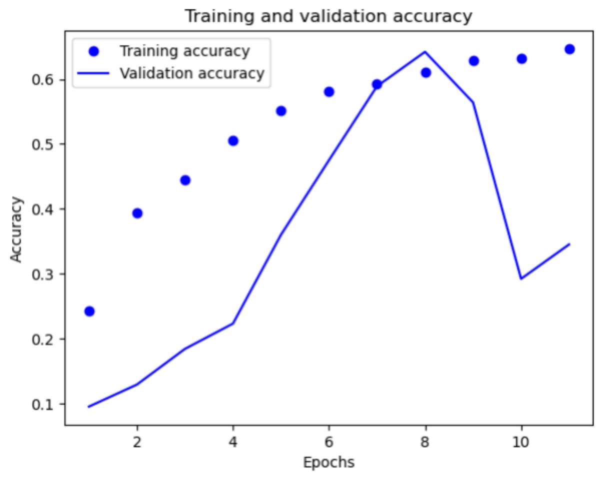
Models

Making our models started with a convolutional neural network architecture consisting of a sequential model, with four iterations of a convolutional layer, max pooling layer, and batch normalization layer then flattened into three dense fully connected layers with dropout finally outputting a classification using a dense eleven output layers using a SoftMax activation function. The second model was a transfer learning model using pre-trained image classification models with a three-layer dense fully connected top into the same SoftMax eleven output layer, which from here on I will just refer to as the top layer. This was tested using multiple models, unfreezing the last module for fine-tuning, as well as the top layer. The models experimented with were the ResNet152V2, the ResNet50, the VGG16, the EfficientNetB0, the EfficientNetB7, and the DenseNet201. Our last models were the resulting knowledge distillation using the best accuracy transfer learning model, and our small convolutional neural network. Our goal was to improve the small convolutional neural networks performance, and hopefully to find something that would be both small and accurate for the datasets stated model deployment goal.

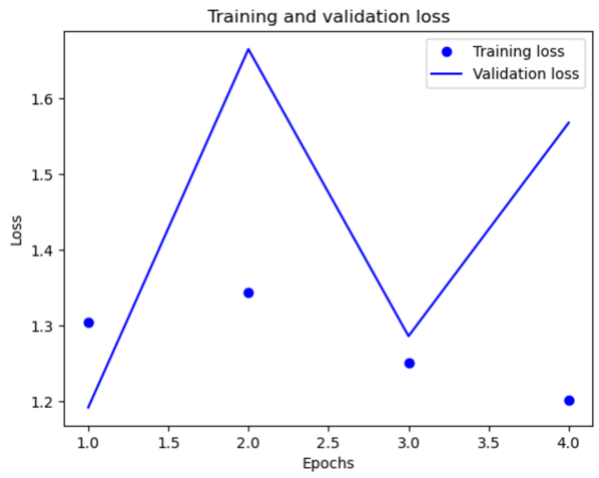
Results

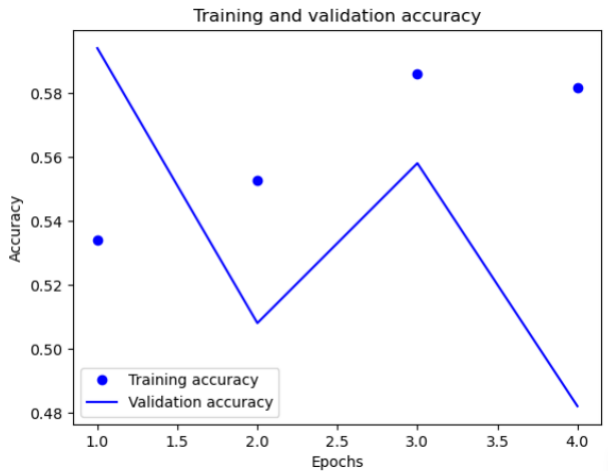
The training of our first model, the small convolutional neural network ran for a total of eleven epochs with a resulting test accuracy of 64.75%.



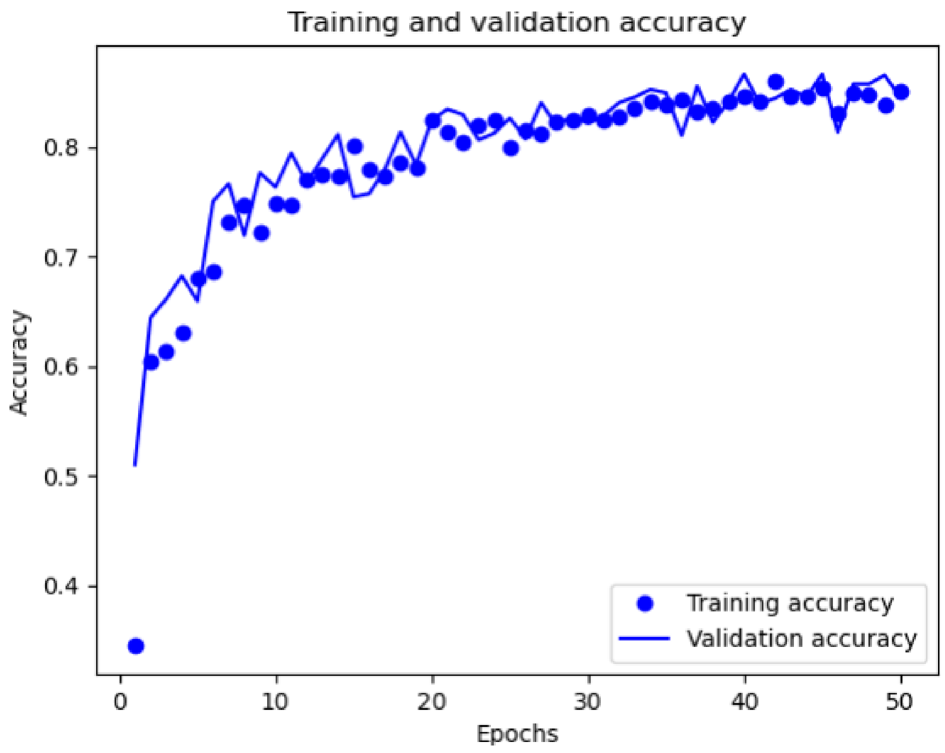


The model was then trained using augmented data resulting in a test accuracy of 64.45%. This was after just four epochs which means that our dataset was sufficiently large to not need a bunch of additional training data. However, we want to continue using data augmentation to avoid overfitting on our first model training.

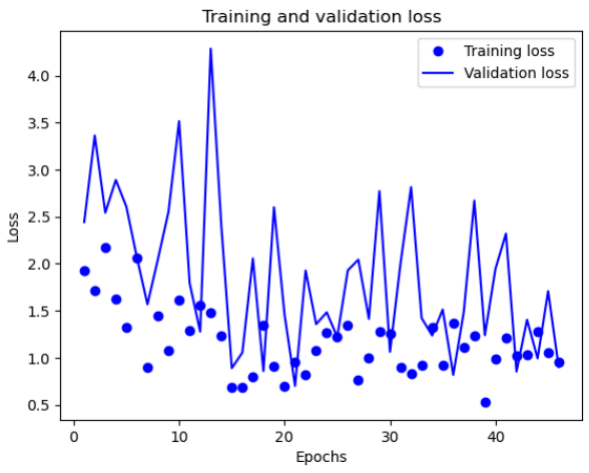


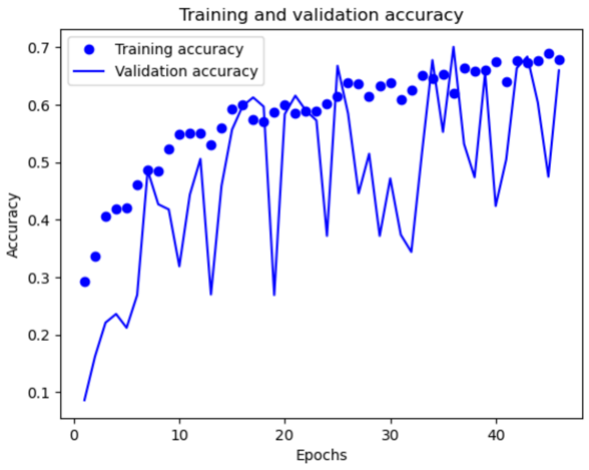


Moving onto the transfer learning models we found our best accuracy with the DenseNet201 model with an accuracy of 87.01%. Unfortunately, it also had a memory footprint of 836.17 MB which is likely to be far too large for our target devices for the model to run on.

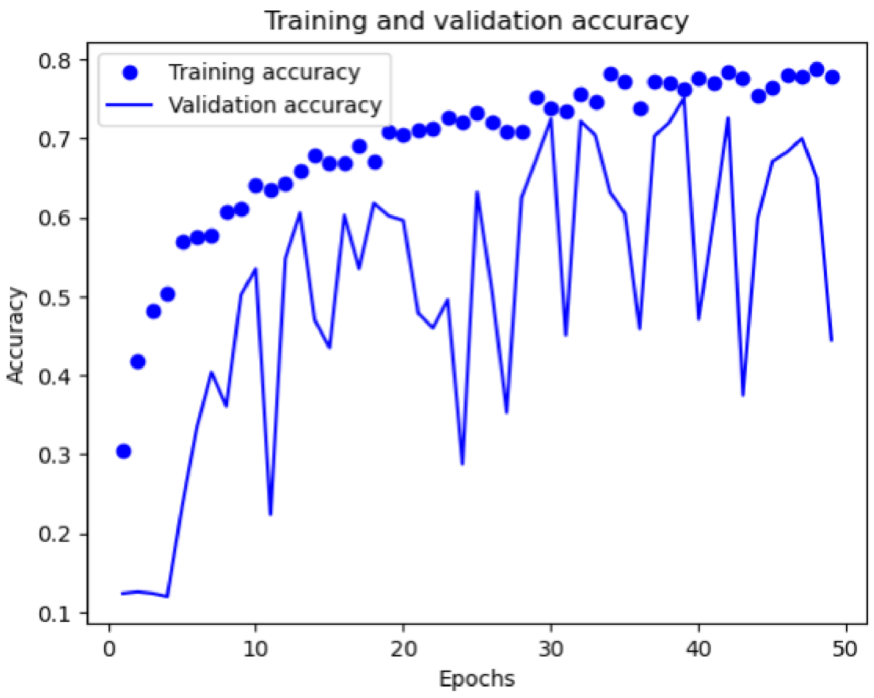


Using knowledge distillation, we then trained our original convolutional neural network with a resulting accuracy of 71.15%, well above the 64.75% the model originally trained to. Additionally, the model was just 1.82 MB's.





Continuing to experiment using our knowledge distillation techniques, we made a larger version of our convolutional neural network to see if we could keep a small model but continue to improve the accuracy.



The resulting model was a 7.07 MB model with 78.65% accuracy. These results showed a clear pattern of improvement and needs further experimentation to see if some other small and accurate implementation can be found.

References

Motwani, A. (n.d.). *Tomato leaves dataset*. Retrieved December 11, 2022, from https://www.kaggle.com/datasets/ashishmotwani/tomato

Team, K. (n.d.). *Keras documentation: Knowledge distillation*. Retrieved December 11, 2022, from https://keras.io/examples/vision/knowledge_distillation/