Tomato Leaf Disease Classification

Tony Zhang

December 13, 2023

1 Introduction

Tomatoes are one of the world's most popular vegetables, with 256 million tons of them being produced world wide in 2021 [fao]. Since they are a flavorful and nutritious vegetable, the ability to grow tomatoes on a large scale can reduce malnutrition by providing a cheap and healthy food source [fIN]. A common problem for farmers is that their tomatoes face many kinds of diseases such as early blight, yellow leaf curl virus, septoria leaf spot, etc. One of the common treatments farms use currently is a large amount of pesticide. The use of toxic chemicals in this kind of large quantity can pollute food and water supplies and harm the environment [SMWH11].

Many of the common tomato diseases show symptoms in the leafs of the plant. This project uses vision based approaches to detect and diagnose tomato plant diseases from looking at the plant's leafs. The hope is that this kind of technology can be used as part of a greater solution where farmers are able to deploy machines that can automatically diagnose and treat their crops. This would save the use of pesticide on healthy plants, and save the use of the incorrect type of pesticide on sick plants. Additionally this could also help farmers catch diseases early on so that action can be taken to minimize spread.

2 Experiment

2.1 Technical Problem

To classify the disease of the tomato leaf, a model must be able to spot certain patterns of discoloration, and learn which kinds of diseases cause the pattern. Due to the wide variation of ways that the disease can manifest itself, it is likely that more complicated techniques will have better results of classification, thus machine learning will be used in order to learn the complex problem.

2.2 Data Set Description

The data set used for this project was found on Kaggle at `https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf`. It contains pictures of individual leafs taken off of the tomato plant. Photos are taken with a light shined directly on the leaf, as well as just using the ambient light of the room. The background of the images is a consistent grey table. There are nine diseases documented in the data set, and one healthy class making this a ten category classification problem. There are 1000 pictures of each class in the training set and 100 of each in the test set.

2.3 Data Pre-processing

The initial data set already has some noise added to it. To further augment it, random transforms were applied to the images, specifically random horizontal flip and random vertical flip. This serves to increase the size of the data set and should increase the generalizability of the model to different leaf orientations.

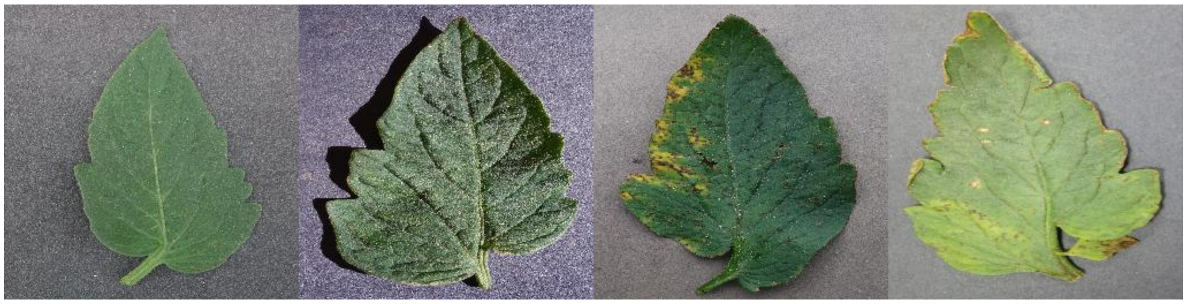

Figure 1: From left to right, 1: Healthy leaf no flash. 2: Healthy leaf camera flash. 3: Leaf infected with bacterial spot. 4: Another leaf infected with bacterial spot.

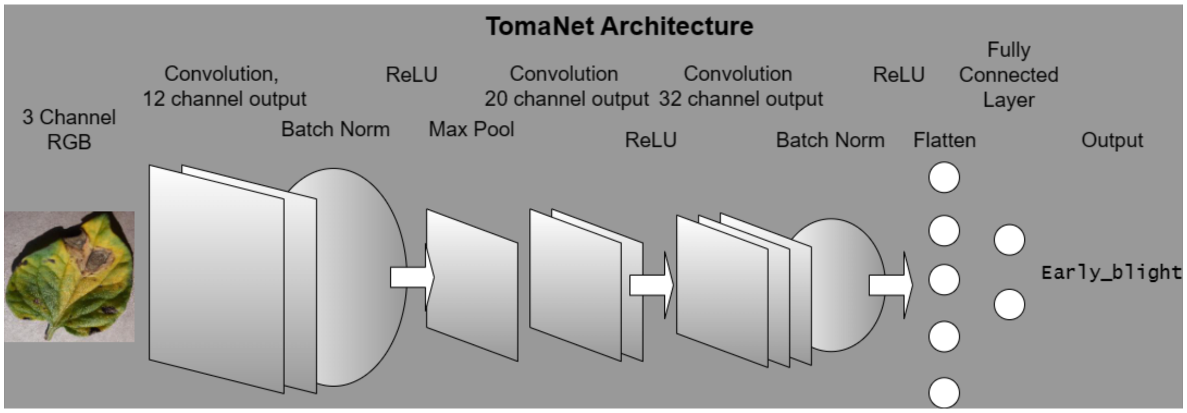

Figure 2: Figure depicting TomaNet Design

3 Algorithms Used to Learn

3.1 CNN TomaNet

The first algorithm that I trained on the data was a convolutional neural network (CNN). It seems like many papers like to name their models, so I will name my CNN TomaNet. TomaNet has a complex architecture consisting of several convolutional layers, batch normalization layers, and pooling layers. It uses ReLU as the activation function and has one fully connected layer before output. The specific design of the CNN is illustrated in Figure 2.

The loss function used was cross entropy. To optimize the parameters to minimize cross entropy, I created my own implementation of the ADAM algorithm. This algorithm combines SGD, momentum, and RMSProp to make a robust optimizer that effectively learn in many function spaces. To test the performance of this optimizer, it was compared against PyTorch's ADAM implementation. Each optimizer was ran for ten epochs.

3.2 Random Forest

To compare the performance of various machine learning methods, I also choose to use some non deep learning methods to classify the data. The random forest technique is an ensemble method where a large amount of decision trees are trained on subsets of the data. To predict the final output, a vote is conducted on all of the trees and the popular vote is the final output. This technique tends to have good results because many weak learners can combine to make a strong complex learner.

3.3 Logistic Regression

Finally logistic regression was also used to train on the data. Logistic regression is equivalent to a single layer neural network with a sigmoid activation function, and since this method is less complicated than the previous methods, I expect it to have lower performance. Due to the size of the data set, I had to use online learning methods, specifically stochastic gradient descent to train this model as to not run out of memory during computation. The model was trained for ten epochs.

3.4 Code

The code for this project was developed in Google Colab. It can make use of GPUs but certainly does not need them to run. The code is linked here and is also in the canvas submission. https://colab.research.google.com/drive/1qXx4-ShL8y-hBQvG984dH7TloP4xiHuQ?usp=sharing

4 Results and Discussion

After training, the performance of each model was collected by taking the accuracy of the model of the test set data. Additionally, the accuracy was broken down by class so that it could be observed how the models performed on a class by class basis. The information is displayed in table 1.

The best performing model was TomaNet trained with PyTorch's built in Adam implementation, reaching an overall accuracy of 0.86. This beats TomaNet trained with my custom implementation of Adam which only was able to achieve 0.74 overall accuracy.

Attempts were made to investigate why my Adam optimizer was not performing worse, and interesting findings were made. As the model trained, the loss went down for the most part but the test accuracy did not follow with it as seen in table 2. This means that my implementation is probably almost correct, but might be missing some small component which optimizes its performance. What is the most interesting to me is that even if my implementation is a little wrong, it still is able to get some pretty decent results for the problem. This is likely due to the fact that if the algorithm is moving in a direction similar to the gradient, then the loss function will reduce because that direction is still a descent direction.

The random forest and logistic regression models had lower accuracy than both of the TomaNet models, achieving 0.61 and 0.60 overall accuracy respectively. This is likely due to these algorithms being less complex than TomaNet. However, I was surprised by how well they could learn despite not having convolution layers or pooling layers to extract and consolidate visual features.

Some of the diseases like Mosaic Virus and Spider Mites were easier to recognise than others. Identifying that a leaf was healthy was actually more difficult than some of the diseases due to many diseases having very subtle visual cues which could have made the algorithms have a tendency to activate when the leaf just looked a little different or had a blemish.

5 TomaNet Applied to Real Images

Early in the semester I visited my mom and took several pictures of her tomato garden, and also asked if she could have any of her church friends take some pictures of their tomato plants. The TomaNet trained with PyTorch's Adam was applied to these images and the output can be seen in figure 3. In the left top image we see that TomaNet accurately classifies the disease even with a very active background. In the top right TomaNet incorrectly classifies the disease when the plant is healthy. This is probably because it sees the yellow fruit in the back. In the bottom image a leaf that is just beginning to be sick is classified as healthy.

6 Challenges

One of the challenges faced during this project was trying to figure out how to train a model on a very large data set. I discovered that some machine learning methods like support vector machines and k nearest neighbors do not play well with large data sets because they like to have the entire data set loaded into memory when they run for computational efficiency. This led me to investigate using online learning methods to train my models. I ended up picking random forest because it takes subsets of the data and trains decision trees individually so it never entirely fills up the memory. I picked logistic regression because I remember it being equivalent to a single layer neural network with sigmoid activation. This allowed me to use stochastic gradient descent to train the model and could be done in PyTorch.

Figure 3: TomaNet applied to images taken from tomato gardens

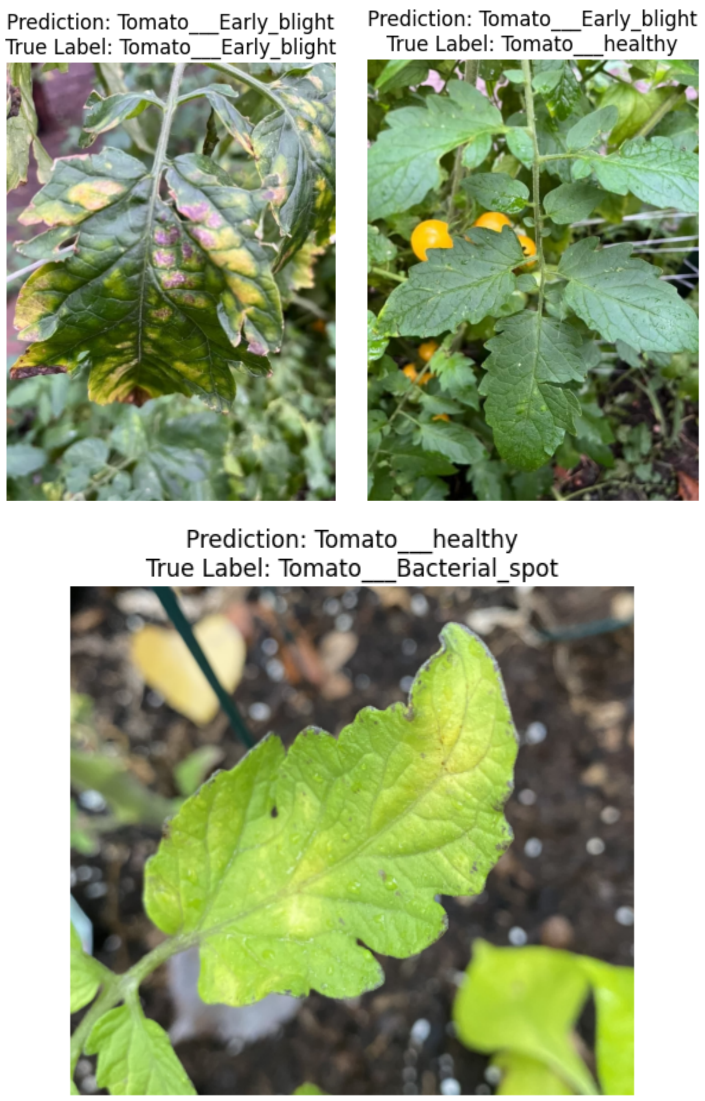
Prediction: Tomato___Early_blight
True Label: Tomato___Early_blight
Prediction: Tomato___Early_blight
True Label: Tomato___healthy
Prediction: Tomato___healthy
True Label: Tomato___Bacterial_spot

Table 1: Test set accuracy for each disease by algorithm

Accuracy by Algorithm	TomaNet PyTorch Adam	TomaNet Custom Adam	Random Forest	Logistic Regression
Healthy	0.83	0.76	0.66	0.36
Bacterial Spot	0.95	0.89	0.83	0.59
Early Blight	0.81	0.31	0.40	0.52
Late Blight	0.81	0.63	0.54	0.46
Leaf Mold	0.74	0.58	0.55	0.58
Septoria Leaf Spot	0.81	0.73	0.57	0.67
Spider Mites	0.85	0.82	0.59	0.67
Target Spot	0.89	0.75	0.46	0.76
Mosiac Virus	0.97	1.0	0.69	0.78
Yellow Leaf Curl Virus	0.95	0.92	0.79	0.58
Overall	0.86	0.74	0.61	0.60

Table 2: Training loss and test set accuracy by epoch for TomaNet trained with my custom implementation of Adam

	Training Loss	Test Set Accuracy
Epoch 1	57.431	0.731
Epoch 2	24.614	0.586
Epoch 3	4.892	0.739
Epoch 4	1.577	0.518
Epoch 5	1.425	0.546
Epoch 6	1.359	0.718
Epoch 7	1.201	0.693
Epoch 8	1.042	0.700
Epoch 9	1.114	0.692
Epoch 10	1.046	0.631

7 Next Steps and Feedback Incorporation

I received a lot of good feedback on my proposal and progress report which I Incorporated into the final project. First, it was suggested that I train multiple algorithms on the data to compare their performance and demonstrate how a CNN learns visual data more efficiently. Additionally, I added some data augmentation to the image loading part of the data pipeline as suggested in order to increase the generalizability and versatility of the model. Third, a great piece of advice I got was to create a visual diagram showing the architecture of TomaNet to clarify the design of the network.

A possible next steps of improvement on this project would be to find a way to collect more images of tomato leafs in their actual growing environment instead of cut off the plant. This would train the model to have to learn what visual information is important and what is background information. It also would help the model not get thrown off by seeing the fruit of the plant like in figure 3.

This may take a lot of time and money so another cheaper option would be to create an algorithm that augments the current data set by making the background of the leaf images include things like tomato fruits and other plants. This has a chance of simulating photos of tomato plants taken in their growing environment.

References

[fao] Gleaned from data provided by FAOSTAT https://www.fao.org/faostat/en/#home.

[fIN] Global Alliance for Improved Nutrition. Tomatoes, the world's most popular vegetable. https://www.gainhealth.org/sites/default/files/publications/documents/advocacy-brief-tomatoes-the-worlds-most-popular-vegetable.pdf.

[SMWH11] Doris Sande, Jeffrey Mullen, Michael Wetzstein, and Jack Houston. Environmental impacts from pesticide use: a case study of soil fumigation in florida tomato production. *International journal of environmental research and public health*, 8(12):4649–4661, 2011.