Erik Truong, Justin Chang
Professor Parijat

Identifying Freshness of Fruits and Vegetables

Project milestones (breakdown of major steps):

Part 1: Process data

First step is to standardize resolutions and make sure they are fit for use in NN. We chose an image resolution of 224 x 224 for now as this seemed to be a good rough number to choose based on eyeballing image resolutions in the data. This may change this later based on image resolution data from the sample images for optimization in the future.

Label the data based on class. This has been done by creating labels for the data based on folders, 2 types of labels, one for freshness, other for type of fruit/vegetable. Based on folder names, we have created 20 different class labels based on a combination of fruit/vegetable (10 classes) and freshness (2 classes) to end up with a total of 20 classes. The data has been normalized as well by dividing the pixel color values by 255.

(If time permits). Need to create 3 different datasets for 3 different labelings: fruit/veggie, freshness, and both. May also figure out a more elegant way of doing this. The idea is to test how each model does based on different classification tasks. In essence how well it can classify food freshness, type of food, or both at the same time. We will be exploring the other classification cases in the future after managing to classify the data on all models for the case of both classes. See bottlenecks sections for issues we ran into during this part.

Part 2: Determining and Testing Architectures (about 6 of them)

Figure out 6 architectures that we can compare results. We plan to have 4 simple architectures ResNet, GoogLeNet, LeNet 5, VGG-16. And 2 stacked architectures EnsembleNet, and Inception-v4. SiameseNetworks will be a backup architecture in case one of the 2 stacked ones don't end up working out during testing.

After figuring out what architectures to test, run the data through the architectures and test performance initially and compare performance with several metrics for classification, but mainly judge model performance based on precision to reduce false negatives. Make sure we are testing the data properly and have done a proper train test split.

For the stacked networks after getting them working with the data, see if the order of the models within the stacked networks affects the performance (accuracy and precision) of the overall end classification. And potentially on the different classification tasks as well.

Erik Truong, Justin Chang
Professor Parijat

Part 3: Hyperparameter tuning

For all the different architectures we are using, mess with hyperparameters after all architectures are working for the initial classification task. This task may be very time consuming. Additionally if time permits test out optimal hyperparameters for different classification tasks and see if they differ based on the classification task as well.

Look into using external hyperparameter optimizers such as weights and biases and whatnot and see if the use of them are helpful for this part. Make sure to graph performance for comparison to see which combination of hyperparameters give the best performance and choose the best ones.

Part 4: Reporting on results and visualizing

Make sure to graph out final results using matplotlib, and other visualization software/libraries for demonstration and comparison of results in the final report.
Final Report should be done in word and be converted into a pdf once it is done.

Milestones completed and results:

The data has been processed and train test split, ready for use in neural networks. More specifically the data has been labeled, resolution has been standardized, and color values have been normalized.

The architectures to test have been decided and implemented within python to the point where we can run data through *most* of the models. There are a few limitations we ran into later that may require changing this up. These limitations are described below in the next section.

Currently we have tested the default models accuracy and all the models have an accuracy of slightly below 0.05 (mostly around 0.045-0.05) which is slightly worse than blind guessing from 1 out of 20 classes. These are obviously not great results but we plan to optimize hyperparameters and the data itself in the future to have much better accuracy and precision.

Overall we are finalizing the phase of initial model testing and beginning the phase of hyperparameter tuning.

Any bottlenecks in completing remaining milestones:

We ran into some issues using Pyspark at first. The idea was to use PySpark as a platform to separate the different fresh and rotten fruits/vegetables images into different datasets. The reason behind this was to use these different partitions of data to see if model performance is

affected if the classification task was different. Additionally, one of our initial questions was whether the classification of freshness first would affect the performance of classifying what type of fruit/vegetable it is and the other way around: fruit/vegetable classification first into freshness. However, the implementation of this did not work due to hardware limitations (RAM issues). We are currently working on another approach to partitioning the data manually.

Another problem we ran into is with the data itself as some of the images in the data are very similar or duplicate. They also contain watermarks such as big black bars at the bottom and we even found one case of a misclassified image where there was an onion in the rotten potatoes section. We plan to mitigate this issue through testing data transformation while optimizing performance.

One other problem is when trying to run GoogleNet, we ran into hardware issues again through RAM limitations, we may either replace GoogleNet with another architecture or down scale the image quality, since this may be based on the sheer volume of the data being 12000 images.