Milestone Report: Food Recognition with Deep Learning

Problem Statement

Food is one of the central aspects of our lives that computer software hasn't really revolutionized, and we still have a lot of problems when it comes to food. There's a growing concern about healthy eating habits -- we consume too much sugar, fats, and salt in our diets. There's also have a big problem with food waste - in America it's estimated that about 30-40% of food produced is wasted.

There have been technology solutions such as phone apps and trackers that help people manage their diets and food waste, but there's a disconnect between the physical realm of food and the interface provided by the technology. Solutions often require the user to bridge this disconnect manually (e.g. by entering nutrition information into a phone app). Image recognition can provide a smoother interface between food and technology, making it more convenient to use. The easier the product is to use for consumers, the more people we can reach and influence to make a change.

In this project, I use deep learning to build a model that can recognize food items. This kind of technology may be seen in phone apps or smart fridges. I also made a simple web application to demonstrate the type of application this model can be used in. The application recommends recipes based off ingredients in an image. We often have leftover ingredients that are thrown out because we don't know what to do with them, and I hope that this kind of technology can help with that problem.

Data Preparation

Collecting the Data

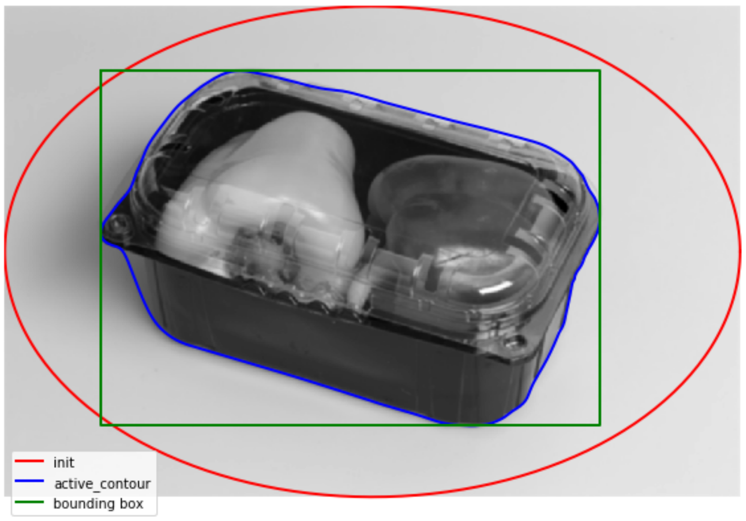
There are 52 classes of images with at least 20 images for each image with more samples for difficult classes (such as pasta). I chose some common ingredients as well as classes that could be difficult to distinguish (such as chicken leg, chicken breast, and chicken wing). The data came from two sources:

1. For fruit images, there's a publicly available dataset from the Visual Cognitive Systems Laboratory which I collected the fruit images from.
2. For all other classes, I manually collected the data using Google image search. The links to where I found each image can be found in the links.txt file.

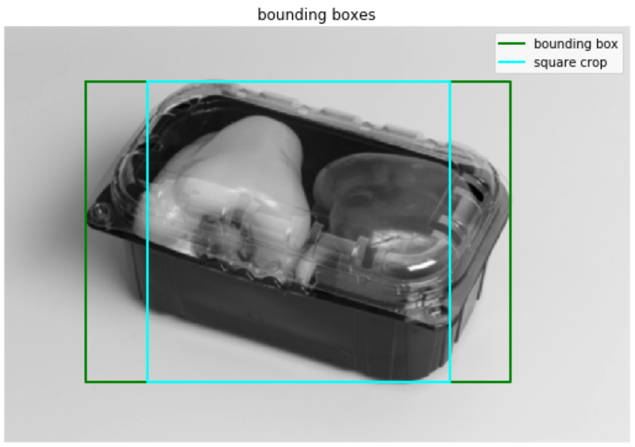
Here are some sample images taken randomly from the dataset:

Cropping and Resizing

In order to use these images as inputs to a neural network, I needed to make square crops of each image. Furthermore, these crops should be around the most important part of the image to reduce noise that won't help the neural network learn. I used an active contour model to get a bounding box for the image crop.



Using active contour to get a minimum bounding box around the object.

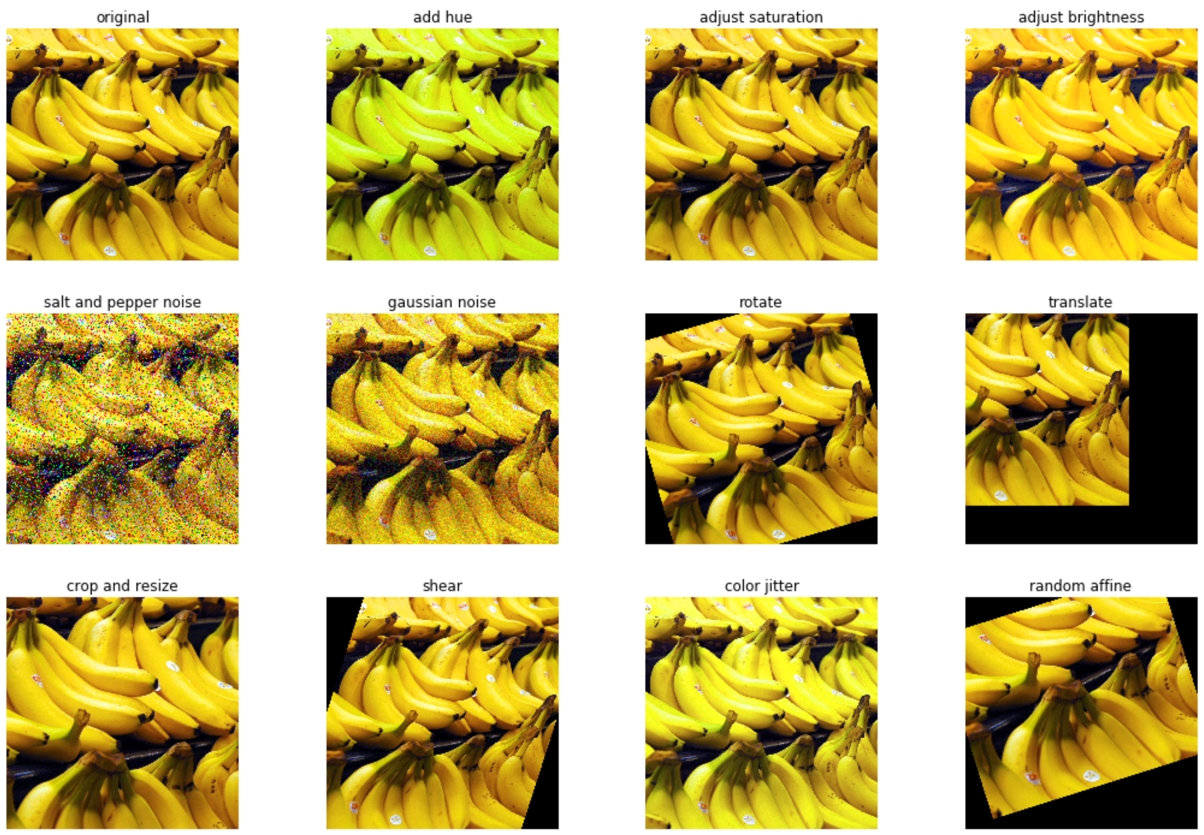


Reduce the minimum bounding box so that it is a square crop.

This process was repeated for all of the images in the dataset and saved to disk in a separate directory.

Data Augmentation

The dataset has relatively few samples for each class. In order to help alleviate this problem and prevent the model from overfitting, I created a pipeline to apply train time data augmentation for each image. These transformations included hue add, adjusting the saturation, adjusting brightness, salt and pepper noise, gaussian noise, rotation, translation, crop and resizing, shear, color jitter, and a random affine transformation.



Example of the data augmentation techniques applied during train time.

Architecture and Training

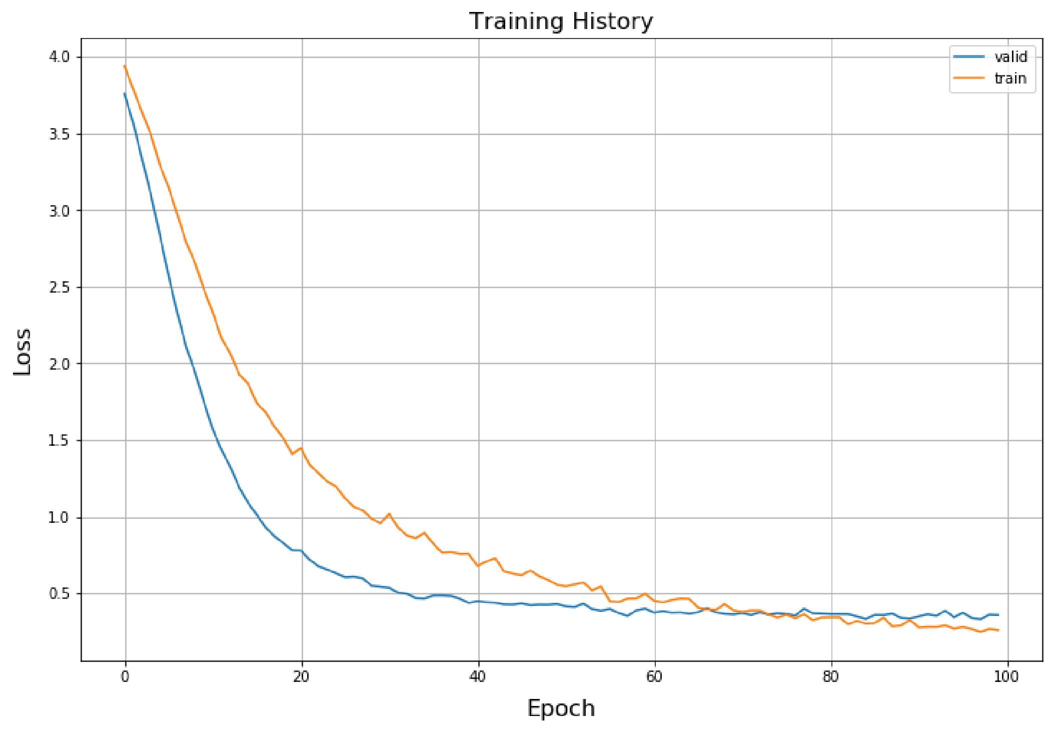
I used a transfer learning architecture to train the model. I used the resnet_50 model as a base and replaced the fully connected layer with one that has 52 outputs, one for each of our classes. I enabled updating of all layers during training (fine tuning). Furthermore, I used cross entropy loss as the loss function.

I split the dataset into a 80%-20% split for the training set and vaidation set.

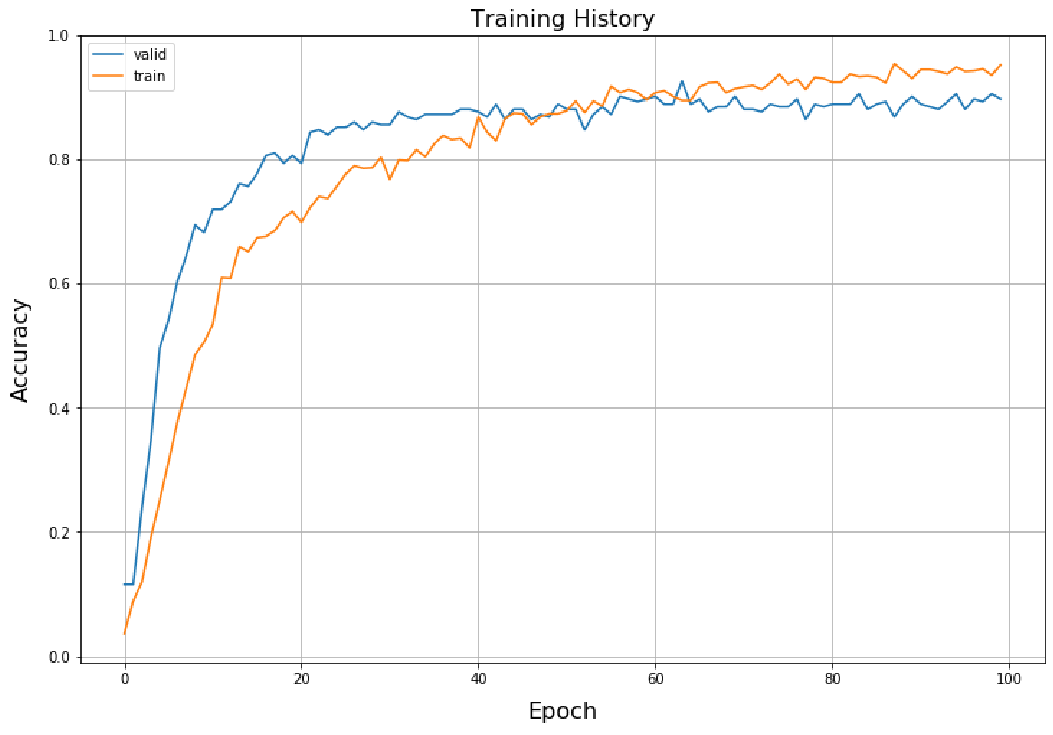
I used the gradient descent optimizer with an initial learning rate of 0.0001, weight decay constant 0.001, and a batch size of 8. I also included an exponential decay scheduler for the learning rate, reducing the learning rate by a factor of 0.1 for every 7 epochs. In total, the model trained for 100 epochs.

Results

Here are the results after training for 100 epochs:



Visualization of the change in the average loss for each epoch.



Visualization of the average accuracy for each epoch.

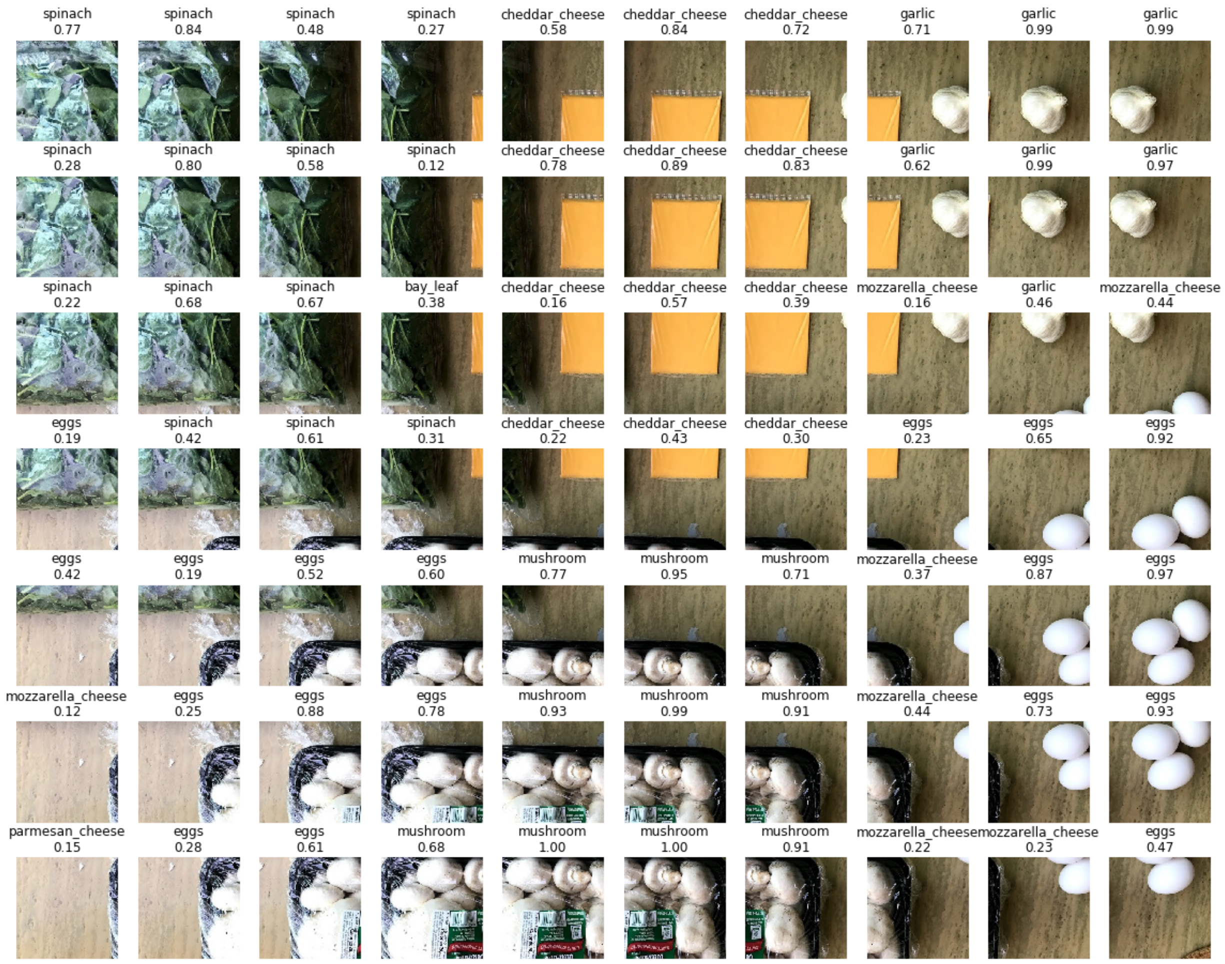
The best validation accuracy occurred at epoch 64 at 92.56%. We can see that this is also where the training loss starts to go under the validation loss, indicating some amount of overfitting.

The following are the classes that the model made misclassifications on:

Class	Validation Accuracy
celery	50.0000
garlic	66.6667
potato	66.6667
thyme	75.0000
bacon	75.0000
chicken_breast	75.0000
basil	75.0000
mushroom	75.0000
ginger	75.0000
carrot	75.0000
chickpeas	75.0000
eggs	80.0000
bell_pepper	80.0000
chicken_wing	83.3333
bananas	87.5000

Application

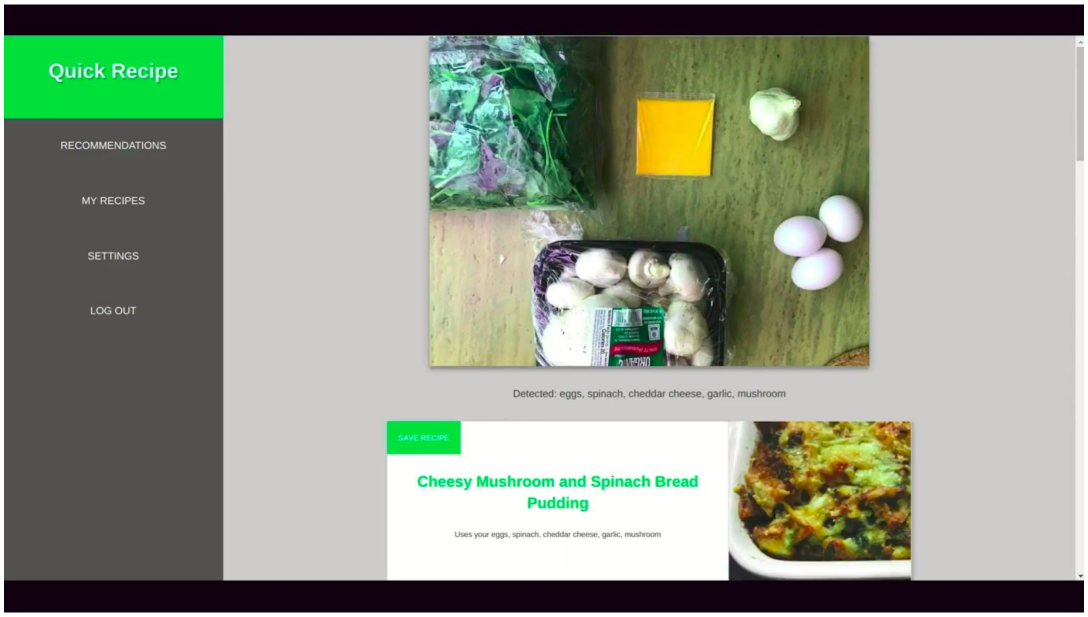
To recognize multiple objects in a single image, I used a simple sliding window method to make predictions on frames from an image, aggregating them into a single set of predictions. Here is an example:



A prediction is made for each frame. The class predicted is shown along with the probability for that class.

We choose only items that are above a probability threshold. When we use a 60% probability threshold, we get cheddar cheese, spinach, garlic, eggs, and mushrooms.

Finally, to demonstrate one application of this model, I created a simple web app that recommends recipes based on the ingredients detected in the image.



Detected: eggs, spinach, cheddar cheese, garlic, mushroom

Cheesy Mushroom and Spinach Bread Pudding

Uses your eggs, spinach, cheddar cheese, garlic, mushroom

A screenshot of the web application that recommends recipes.

I created a demo on YouTube to show how it works.