

Identifying Food Allergens from Images of Prepared Foods

Problem Statement

This project aims to create an image classification model to predict the likelihood of whether one or more of the major nine food allergies are present in a given image.

Context

In the United States, food allergies affect 4-6% of children and 4% of adults, and the number of people suffering from food allergies is on the rise.¹ For children, the prevalence of food allergies increased by 50% between 1997 and 2011 and again by 50% between 2007 and 2021, resulting in billions of dollars in medical treatment.² While sometimes symptoms can be mild, others can be life threatening, and can include things like vomiting, hives, and anaphylaxis which can impair breathing. According to the US Food and Drug Administration, there are nine major food allergies: milk, eggs, fish (like bass, flounder, and cod), crustacean shellfish (like crab, lobster, and shrimp), tree nuts (like almonds, walnuts, and pecans), peanuts, wheat, soybeans and sesame.³

Problem identification

In a cursory search of readily available datasets about food and allergens, one was identified, but each image only had one label associated with it, and many of the images were of raw ingredients as opposed to prepared foods. This seemed unrealistic and did not make a good business case for the creation of a model. Instead, the problem statement was to identify any number of allergens based on images of prepared foods.

This problem is a multi-label image classification problem because any one image could contain multiple allergens. While similar to most image classification problems, there are a few key differences which will be discussed later.

Data Collection and Processing

The first step of the problem was to collect the data. Prior to actually harvesting the images, a decision needed to be made, out of the millions of foods out there, which ones would be collected. A few rules guided that decision:

- Each allergen should be represented in the final data set

¹ "Food Allergies: Causes, Symptoms, Treatments," <https://acaai.org/allergies/allergic-conditions/food/>.

² "Facts and Statistics - Food Allergy.org", <https://www.foodallergy.org/resources/facts-and-statistics>

³ "The Faster Act: Sesame is the Ninth Major Food Allergen", https://www.fda.gov/food/food-allergies/faster-act-sesame-ninth-major-food-allergen?gad_source=1&gclid=CjwKCAiA0rW6BhAcEiwAQH28lrrSWFirWo9L_I8QVBFErfj8oxwvYCKbSz-wwKPowkusKDHT-Ms5hoC4hYQAvD_BwE

- There should be foods that contain no allergens
- Effort was made to select a diversity of food from around the world, and not just from North America.
- A few examples of some common foods were included. For example, several types of cookies and several types of pies (including quiche), were included to help guard against the model thinking all pies have pecans in them, for example.

A list of over one hundred foods was compiled, in some cases based on searches for foods with particular allergens, e.g. sesame. The original list was whittled down to 99 foods, eliminating those sets of allergens which were overrepresented. By far, wheat was the most common allergen, followed by eggs and milk.

As the list of foods was being assembled, simultaneously the “labels” for each food was recorded. For each food, a “typical” recipe or list of ingredients derived from a Gemini search was used to establish the labels. Labels were recorded for each ingredient as either “0” (not present) or “1” (present).

Once the final list of foods was codified, the process of collecting images began. A web scraper was created that would harvest images from a Google Image Search of the name of the food. Images were filtered so that only images of a minimum size were kept. One issue that came up was that some images contained additional allergens that were not present in the Gemini recipe. For example, images of bagels and cream cheese would return images of bagels and cream cheese with lox, bagels with cream cheese and sesame (seeds), or even bagels and cream cheese and sesame with lox. In most cases these images were manually separated out and added as their own group, unless there were very few images, in which case those images were simply deleted from the dataset.

EDA

Traditional EDA is not as relevant for image classification tasks. That said, some basic visualizations were performed including a display of 16 randomly selected images and their labels; a histogram of the number of images in the dataset per food and a histogram of the pixels in a single randomly selected image.

eggs, wheat, soybeans



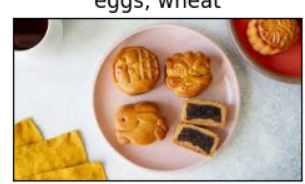
milk



milk, wheat, soybeans, sesame



eggs, wheat



milk, tree nuts, wheat



wheat



none



fish, shellfish, wheat, soybeans, se



milk



none



milk, wheat, soybeans, sesame



wheat



wheat, soybeans



none



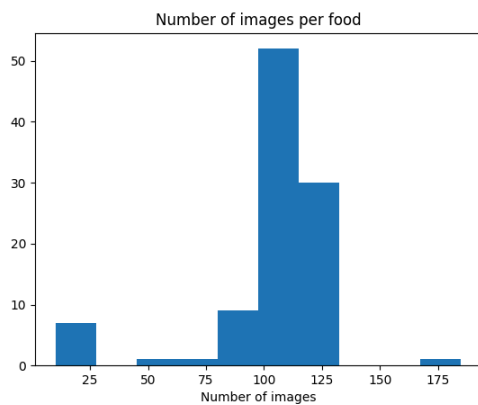
eggs



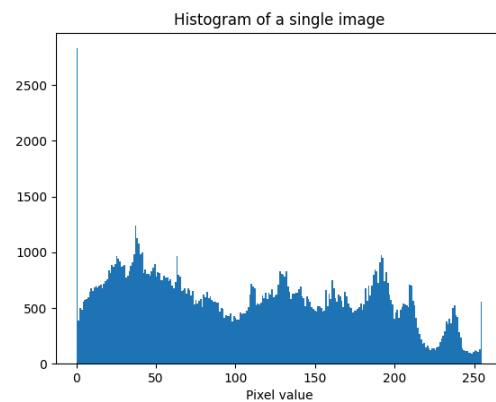
none



A selection of images and their labels from the dataset



A histogram of the number of images per food



A histogram of the pixels from one randomly selected image

File Manipulation

Before the models could be created, some file manipulation had to be done. This included several steps:

- Repeating the labels in the labels CSV file for as many images as existed in the respective folder
- Renaming the images consecutively and adding their file name to the labels CSV file
- Copying all of the images to a single folder
- Resizing all images to be the same size
- Converting the images to numpy arrays
- Normalizing the pixel values
- Shuffling the images to avoid the neural networks from learning from images of the same foods being grouped together.

Validations were conducted throughout the data preparation stages to verify that the labels correctly associated with the images.

Modeling

The next step was modeling. The modeling notebook was written in Google CoLabs in order to take advantage of the access to High-RAM runtimes which were necessary for processing the quantity of images in the dataset.

The first model created was a simple convolutional neural network (CNN) with four hidden layers in addition to the input and output layers. Batch normalization, max pooling and dropout were also used for each layer. The construction of the output layer for multi-label classification problems varies from regular multi-class classification problems because the number of filters must equal the number of labels, and the activation function must be “sigmoid” because the

probabilities for the occurrence of each label is independent of the others, and the sum of all of the label probabilities does not add up to one.

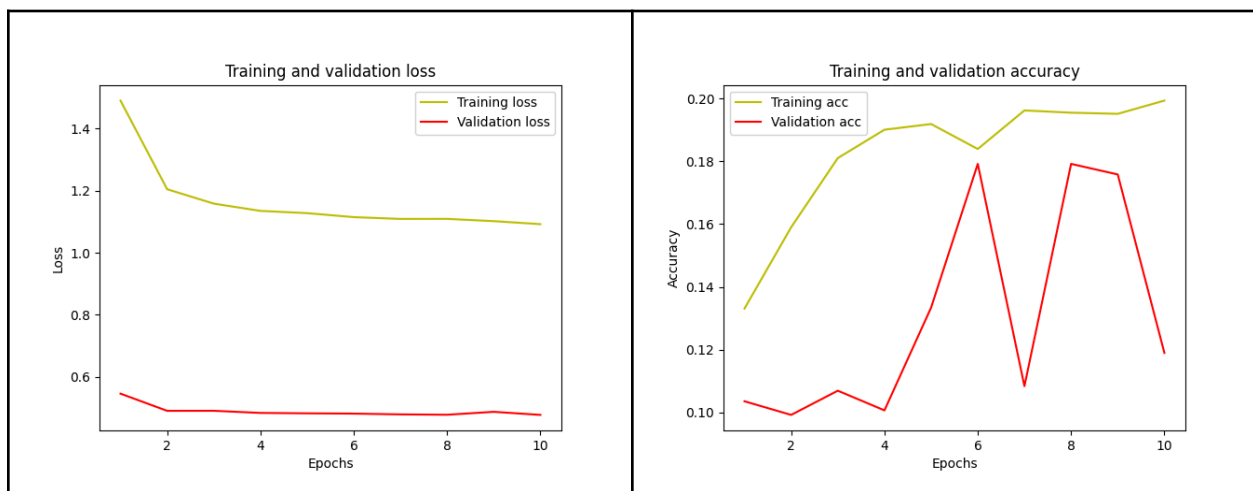
After each model, the validation loss and validation accuracy were calculated and the model was used to predict on a random image, in the case of the first two models, the image was of baklava.


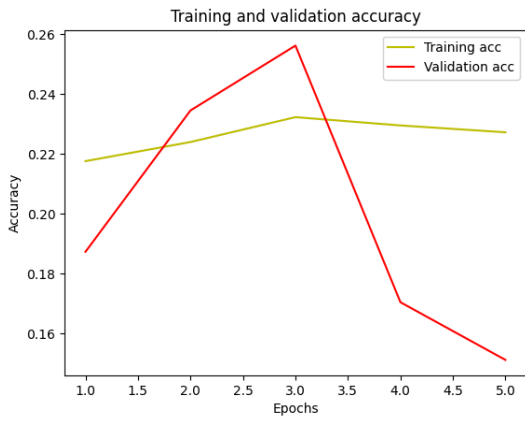
A table of the loss and accuracies for each model is below.

The second model was created similar to the first, but with the addition of two callbacks: Early Stopping and Model Checkpoint. A callback is a function that is executed after an action or another function takes place and is passed as an argument to the model.fit statement. Early Stopping tells the model to stop if a specified number of epochs go by without improvement in the model accuracy (in this case 5). The Model Checkpoint callback serves to save the model or weights at an interval for retrieval later. In this case the “save_best_only” argument was set to True.

Unfortunately, the results from the models were not tremendously successful. Accuracies stayed below 0.20 and loss above 0.4. When the model was used to predict the labels for a food like baklava, the probabilities of the top labels were in the range of 0.4-0.6. The model did tend to predict most of the correct labels with higher probabilities than the incorrect labels.

Table of validation loss and accuracies for each model		
	Validation loss	Validation accuracy
Initial model	0.4808	0.1238
Model with callbacks	0.4627	0.1619
Final model	0.4444	0.1905



<i>Losses for initial model</i>	<i>Accuracies for initial model</i>																																				
 <p>Training and validation loss</p> <table><tr><th>Epochs</th><th>Training loss</th><th>Validation loss</th></tr><tr><td>1.0</td><td>1.05</td><td>0.45</td></tr><tr><td>2.0</td><td>1.05</td><td>0.45</td></tr><tr><td>3.0</td><td>1.06</td><td>0.48</td></tr><tr><td>4.0</td><td>1.05</td><td>0.45</td></tr><tr><td>5.0</td><td>1.08</td><td>0.47</td></tr></table>	Epochs	Training loss	Validation loss	1.0	1.05	0.45	2.0	1.05	0.45	3.0	1.06	0.48	4.0	1.05	0.45	5.0	1.08	0.47	 <p>Training and validation accuracy</p> <table><tr><th>Epochs</th><th>Training acc</th><th>Validation acc</th></tr><tr><td>1.0</td><td>0.218</td><td>0.188</td></tr><tr><td>2.0</td><td>0.225</td><td>0.235</td></tr><tr><td>3.0</td><td>0.232</td><td>0.255</td></tr><tr><td>4.0</td><td>0.230</td><td>0.170</td></tr><tr><td>5.0</td><td>0.228</td><td>0.150</td></tr></table>	Epochs	Training acc	Validation acc	1.0	0.218	0.188	2.0	0.225	0.235	3.0	0.232	0.255	4.0	0.230	0.170	5.0	0.228	0.150
Epochs	Training loss	Validation loss																																			
1.0	1.05	0.45																																			
2.0	1.05	0.45																																			
3.0	1.06	0.48																																			
4.0	1.05	0.45																																			
5.0	1.08	0.47																																			
Epochs	Training acc	Validation acc																																			
1.0	0.218	0.188																																			
2.0	0.225	0.235																																			
3.0	0.232	0.255																																			
4.0	0.230	0.170																																			
5.0	0.228	0.150																																			
<i>Losses for model with callbacks</i>	<i>Accuracies for model with callbacks</i>																																				

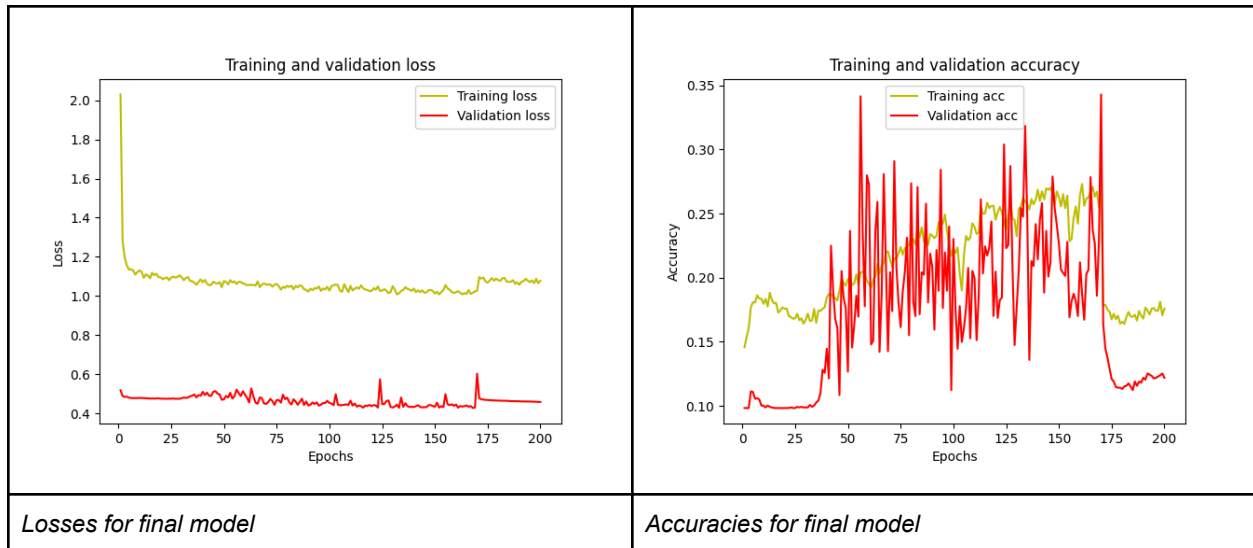
Hyperparameter Tuning

Two rounds of hyperparameter tuning were conducted. The first tuning included the optimizer, the epochs, the batch size and the activation function for the hidden layers. In the second round, the number of hidden layers and the number of filters were tested. The values tested for each parameter are in the table below. The values the tuning selected are in bold.

Parameter	Values tested
optimizer	sgd, adam
epochs	3, 100, 200
batch size	16, 32, 64
activation function	relu , tanh
number of hidden layers	1, 2, 4 , 8
number of filters	16 , 32, 64, 128

Final Model

Using the values for the hyperparameters determined in the hyperparameter tuning, a final model was created and run.



Despite the hyperparameter tuning, the model loss and accuracies did not improve much. In addition, it seems that the accuracy increases around 172-174 epochs before plummeting. The final model loss was 0.4444 and the accuracy was 0.1905.

Further research and next steps

While ultimately not particularly predictive, the model did improve slightly over time. That said, it would be useful to conduct additional hyperparameter tuning to potentially discover a better performing model. In addition, the images should be used with pretrained models like Yolo or MobileNet, to see if leveraging those model's intelligence could help improve the accuracy on this dataset.

Conclusions

It is clear that the objective of predicting food allergens through images of prepared foods is complicated for several reasons. These include the fact that some ingredients are not visible (e.g. butter in cookies), and the sheer number of, not only the kinds of prepared foods, but the different ways prepared foods might look when prepared by different people. That said, there is still a case for continuing to work on this model as it could have real value in helping to identify potential allergens in food.