Food Analyzer: DCNN Transfer Learning for Food Recognition and Ripeness Classification

Henry Bobeck[1], Sam Stazinski[2], Devarshi Bhadouria[3], Zane Snider[4]

Contact: [1]hsbobeck@alumni.iu.edu, [2]samstazi@iu.edu,

[3]zesnider@iu.edu, [4]dbhadour@iu.edu

Code: `https://github.com/hsbobeck/food-analyzer`

1 Introduction

The goal of our project is to detect and identify food from an image, and then proceed to give detailed information based on the image info, such as how long it will be until this food expires, nutritional info about the food, and/or general information about the food. From a user-side process, the Food Analyzer first takes in a provided input image of a piece of food. It then interprets the image and provides back information of 1) the name of the food, 2) its current ripeness/predicted expiration, and 3) basic and nutritional info about the food. In order to achieve this final goal, we provide solutions for the following subproblems: first, to design and train a neural network to correctly identify food in an image. Second, to design and train individual neural networks to correctly predict food ripeness for a specific food. And third, to write a script to retrieve basic nutritional information for a given food name. In this way, we approach the design structure of the project using a cascading style, in which the results for subproblems (2) and (3) above rely on the result from subproblem (1), the food recognition model. Through the efficacy of our resulting final product, we show that this method of combining cascading deep convolutional models and scripts into one fluid, intuitive user-side process is an effective approach for AI-based application development.

2 Background and Related Work

Image classification of food has existed before and has been done in various implementations of classification. Our project takes image classification further by classifying food items along with their expected expiration, or ripeness level, as well as providing nutritional info for the food. It is often difficult to predict ripeness of a food, as different foods display their ripeness and age in varying ways. Therefore, for most real-world purposes, food ripeness classification is still done manually. In this paper we seek to show that development of computer vision

and machine learning techniques can be used to do this automatically. We use a custom CNN model based on a state of the art CNN model using transfer learning. The data used consists of different food images sorted by food label as well as different ripeness stages. By limiting ripeness classification to fruits, we believe we are setting a reasonable goal to reach high accuracy on ripeness classification, as many other foods tend not to have significant visual cues for precise expiration date prediction. Furthermore, recent accomplishments in object recognition have already shown models based on deep CNNs perform well for the problem space; therefore, our efforts seek to show how the used techniques can be combined in a unique cascading style with polished user-side process in a real world-application. Related scientific papers are linked below:

- Banana Ripeness Classification Based on Deep Learning using Convolutional Neural Network
- Computer Vision and Image Analysis based Techniques for Automatic Characterization of Fruits – a Review
- Object identification from few examples by improving the invariance of a Deep Convolutional Neural Network
- Large Scale Visual Food Recognition
- Very Deep Convolutional Networks for Large-Scale Image Recognition

3 Development Process and Challenges

The first aspect that we worked on was the overall food recognition model, as we knew that without a highly effective solution to this subproblem, the rest of the project would suffer due to its cascading nature. In this regard, we underwent multiple different iterations of attempts to solve this problem. In each of those drafts the following methods have remained consistent: first, the use of a transfer learning approach to capitalize on proven models trained with millions of images for the general problem of image classification.

At first, we made use of a pre-trained VGG-16 network for this purpose, freezing the majority of the layers and training our own (specifically food) data on the last few. Second, in all iterations we have used some form of tensorflow's training and model construction patterns, including using the tf Dataset data structure to handle the processed inputs to the model. Other than this, our approach to specifically the image data pipeline has gone through multiple iterations as we have had to self-learn the best practices for structuring this as well as learning how to use both tensorflow and image processing packages. This learning process led to us encountering multiple roadblocks which revealed an error in our design structure and thus required a restructured approach. Because of this, perfecting our image data pipeline took a considerable amount of learning and development time.

The first roadblock involved the form of the dataset that we initially attempted to use for food recognition training. The most promising dataset of food images we found was

made available online in only the ActiveLoop Hub format, which is a dataset hosting service that attempts to redefine the commonly used image data pipeline by streaming all data whenever it is needed rather than downloading an entire dataset at once. While great in theory, however, as novice computer vision students who have not yet had experience with the standard form of image data pipeline management, this led to significant challenges in handling the processing and preparation of the data with such a novel method that has highly limited online documentation. When the given hub dataset was not even able to convert itself into the more well understood Tensorflow Dataset structure, for example, we found ourselves to be at a loss.

The second roadblock came largely hand-in-hand with our lack of experience using tensorflow in the past. Specifically, significant time was spent learning how the tf Dataset structure encodes its data into tensors, and how we could apply transformations and preprocessing functions to this data while it was in this state. In the end, we found that performing as much of the preprocessing (including one-hot encoding training labels, for example) as we could before enveloping the data into the Dataset structure was a much smoother experience than attempting to process the data afterwards, and having to deal with data manipulation with tensorflow running in graph mode.

In the earliest functional version of our project, our VGG-16 based transfer learning food recognition model was successful in its ability to be trained by the Food101 dataset, however after peaking at around 50% accuracy and 35% validation accuracy after 22 epochs as seen in Fig. 1, it was clear that more research and changes could be done to increase the model's performance. This task, we realized, would involve testing base models more proficient than VGG-16, further tuning the parameters of our custom model, as well as adding increased variation in the image data by means of random transformations in the image data pipeline, all of which we implement in (4) The Final Method.

4 The Final Method

4.1 Transfer Learning

The challenge of image-object recognition is by no means a novel problem. In fact, there exist a wide variety of solutions to this problem in the form of many different strategies and models. While there is no single best strategy to solve it, though, there are still highly effective and proven models which make use of deep convolutional neural networks (DCNNs) that have been tailored and pre-trained on millions of images to excel at decoding object information in images. Given our comparatively limited data and available training time, using these proven models is undoubtedly our best option. However, while these models are effective at object prediction, they aren't yet specialized to our specific food classes or ripeness stages. Therefore, by replacing the last few layers of a pre-trained model, freezing the deep layers, and then training on our specific data, we are able to make use of the powerful base model while also directing the predictions to our custom set of classes. This approach is called transfer learning, and the benefits of this approach include much less

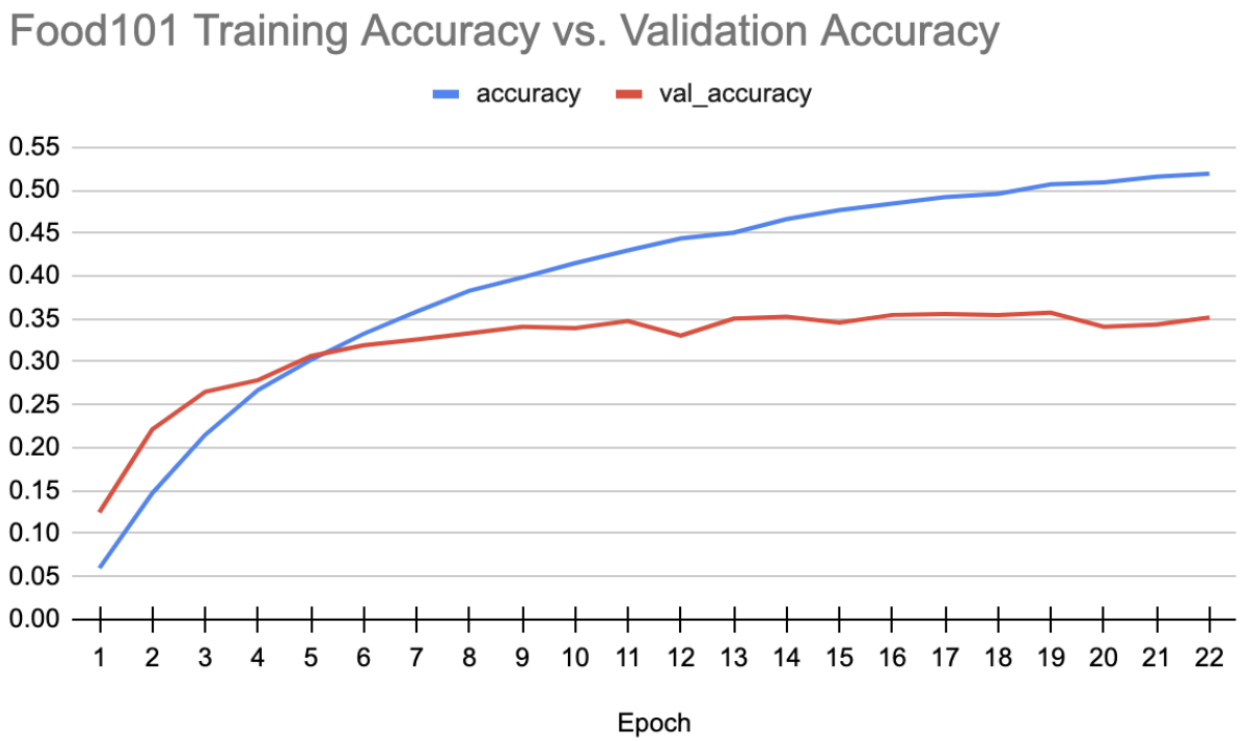

Figure 1: Training accuracy vs. validation accuracy while training the earliest version of our food recognition model with a VGG-16 base on the Food101 dataset, with all 101 classes included.

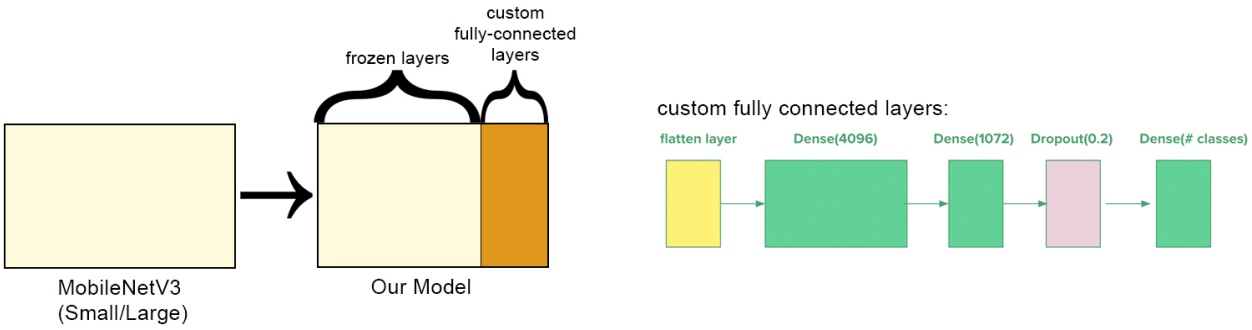

Figure 2: Our final custom model architecture using transfer learning with MobileNetV3 as a base model.

required data, much shorter training time, and much better results compared to designing and training a similar network from scratch.

4.2 Deep CNN Architecture

The architecture of our separate classification models are nearly identical, with only a couple small differences. As can be seen in Fig.2, the base for our food recognition model is MobileNetV3_Large, while the base for our food ripeness model is MobileNetV3_Small. To decide on these base models, we performed a series of tests comparing results across multiple state-of-the-art models including VGG-16, Xception, and MobileNetV3. The results can be seen in Section (5). For our custom final fully-connected layers, we performed a similar testing process to decide on (in order): a Flatten layer, a Dense layer of size 4096, a Dense layer of size 1072, a Dropout layer of 0.2, and finally a Dense prediction layer with size equal to the number of classes. With this construction we sought to strike a balance between model accuracy and reasonable training time.

4.3 Training Data

The majority of our food recognition training data comes from the openly-available dataset Food101. In the current version of our product, we utilize ten of the 101 given classes in order to reduce necessary training time, which is a decision we were forced to make given the runtime limits of the service Google Colab. We then supplemented this data with 256 images of a new class 'banana', which we pulled from google image results. We also sorted these banana images into three distinct ripeness classes based on the appearance of the bananas. This sorted banana data was then used as training data for the banana-specific ripeness detection model, which we use as a proof of concept for ripeness detection. To extend our application to also detect ripeness of additional foods, or to further specify additional ripeness stages for a specific food, all that is required is to collect and sort the necessary images and use our existing ripeness detection model architecture to train a new model for that specific food.

4.4 Image-Data Pipeline

In order to train our models, images are first sorted into folders labeled with the desired class. We then read these files into a pandas DataFrame with columns 'path' and 'class_name'. Using scikit-learn's train_test_split function, we split this dataframe into two, 80% towards training, 20% towards testing. From this point, we make use of keras' image preprocessing utilities to read the image data from the DataFrame into an ImageDataGenerator Dataset. As part of this process, all input images are sorted into batches of 32, converted to rgb, and resized to 224x224. The labels are one-hot encoded. Furthermore, in order to reduce potential for overfitting, we apply randomized transformations to the training images (not the testing images) regarding brightness, width, rotation, and flips. The images are then preprocessed to the desired input format of the base MobileNetV3 model, and a validation split of 0.15 is applied for the training images.

4.5 Food Information Scraper

Once we have a resulting prediction from an input user image, one of the goals of our project is to provide nutritional and general information regarding the detected food. To achieve this, we utilize the Python library Scrapy to collect information displayed by Google after a Google search of "what is an %FOODNAME%" in the form of a string. The resulting text for a specific food is then displayed to the user when that food is detected in an image, as part of our final user-side interface.

4.6 Flask User Interface

In order to provide an appealing user-side interface for our project, we developed a full-stack web application with Flask. In doing so, we set up an HTML/CSS webpage to communicate with a Python backend which handles the tensorflow models and outputs as well as the general information scripts. Through creation of an appealing and intuitive final user interface, it is our intention to show that our method of combining cascading deep convolutional models and scripts into one fluid, intuitive user-side process is an effective approach for AI-based application development.

5 Results

5.1 Food Recognition Results

To decide on the base model to be used for transfer learning, we performed a series of tests comparing results across multiple state-of-the-art models including VGG-16, Xception, and MobileNetV3. The results can be seen in Fig.3 and Fig.4. Note that the validation accuracy is significantly lower than the final testing accuracy; this is expected due to the randomized image transformations applied to the training set of images. The testing set is thus intended to represent the most realistic set of inputs, and therefore the testing accuracy of 82.07%

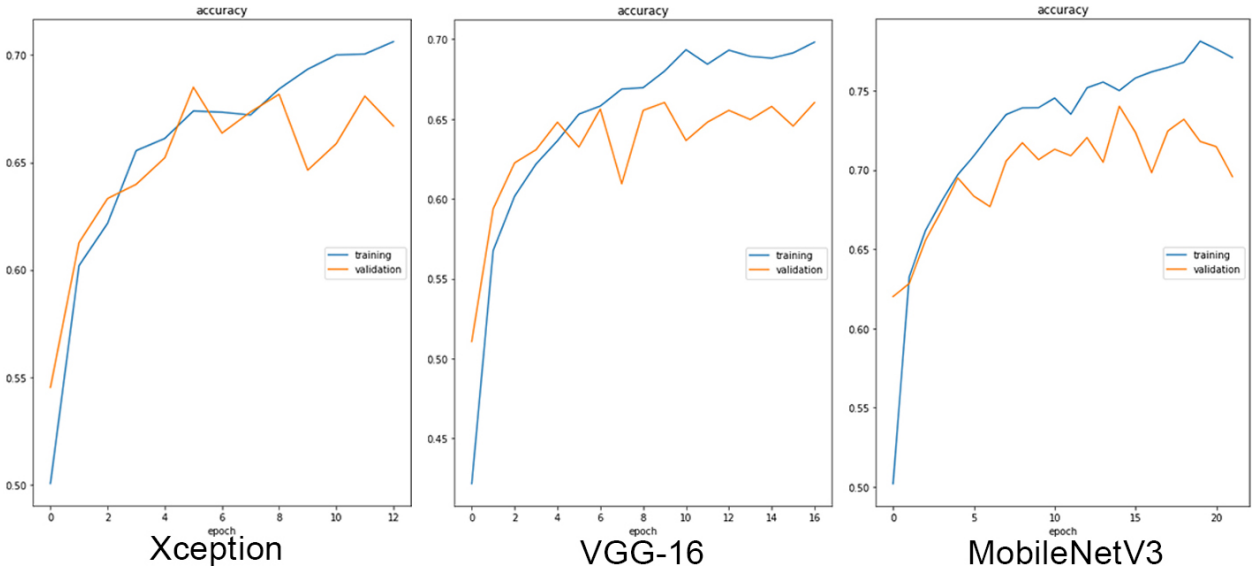

Figure 3: A comparison of the training and validation accuracy between varying base models during the training process for the food recognition model, with 11 classes included.

for the MobileNetV3 base is most representative of a real user's experience. Training was performed for a maximum of 25 epochs each, with the potential for early stopping after 7 epochs without improvement in validation loss. The saved model is decided from the epoch with the lowest validation loss. Fig.5 shows a set of example results from the model with a MobileNetV3 base, which is the base we select for our final model.

5.2 Banana Ripeness Results

For the final food ripeness model, we decided on a base model of MobileNetV3_Small, which is a smaller and more lightweight version of the large MobileNetV3 we use in the food recognition model. Despite our quite limited data for training this model, the model performs exceptionally well in testing due to the use of transfer learning. Training was performed for 25 epochs with batches of 8. The model achieved 96.15% for its testing accuracy, and an example of labeled image results can be seen in Fig.8.

5.3 Application User Interface Results

The application user interface involves a simple and intuitive introduction page (see Fig.9) as well as a results page (see Fig.10). The results page appears once the user submits an image file. The required models are loaded lazily. For example, once an image is first submitted, the food recognition model will load, and will stay loaded for the rest of the session. If a food is then detected that has a valid ripeness model available, the ripeness model will load at that point and remain loaded for the rest of the session. Once a model is loaded, the results page is shown nearly instantaneously.

Overall Categorical Accuracy: 80.12%

	Precision	Recall	F-Score	Support
banana	1.000000	1.000	1.000000	52.0
guacamole	0.931818	0.820	0.872340	200.0
pizza	0.841860	0.905	0.872289	200.0
hamburger	0.814286	0.855	0.834146	200.0
hot_dog	0.848168	0.810	0.828645	200.0
pancakes	0.792453	0.840	0.815534	200.0
donuts	0.725100	0.910	0.807095	200.0
waffles	0.876471	0.745	0.805405	200.0
sushi	0.852564	0.665	0.747191	200.0
tacos	0.660079	0.835	0.737307	200.0
nachos	0.692771	0.575	0.628415	200.0

Xception

Overall Categorical Accuracy: 75.97%

	Precision	Recall	F-Score	Support
banana	0.962963	1.000	0.981132	52.0
pizza	0.939227	0.850	0.892388	200.0
guacamole	0.821256	0.850	0.835381	200.0
donuts	0.835897	0.815	0.825316	200.0
sushi	0.837209	0.720	0.774194	200.0
waffles	0.843373	0.700	0.765027	200.0
hot_dog	0.775956	0.710	0.741514	200.0
hamburger	0.728205	0.710	0.718987	200.0
pancakes	0.594406	0.850	0.699588	200.0
tacos	0.698413	0.660	0.678663	200.0
nachos	0.598214	0.670	0.632075	200.0

VGG-16

Overall Categorical Accuracy: 82.07%

	Precision	Recall	F-Score	Support
banana	0.962963	1.000	0.981132	52.0
pizza	0.971429	0.850	0.906667	200.0
guacamole	0.828054	0.915	0.869359	200.0
donuts	0.905028	0.810	0.854881	200.0
sushi	0.848485	0.840	0.844221	200.0
hot_dog	0.890805	0.775	0.828877	200.0
hamburger	0.757447	0.890	0.818391	200.0
waffles	0.864407	0.765	0.811671	200.0
pancakes	0.694779	0.865	0.770601	200.0
nachos	0.715640	0.755	0.734793	200.0
tacos	0.776536	0.695	0.733509	200.0

MobileNetV3

Figure 4: A comparison of the overall and class-specific testing scores between varying base models during the testing stage for the food recognition model, with 11 classes included.

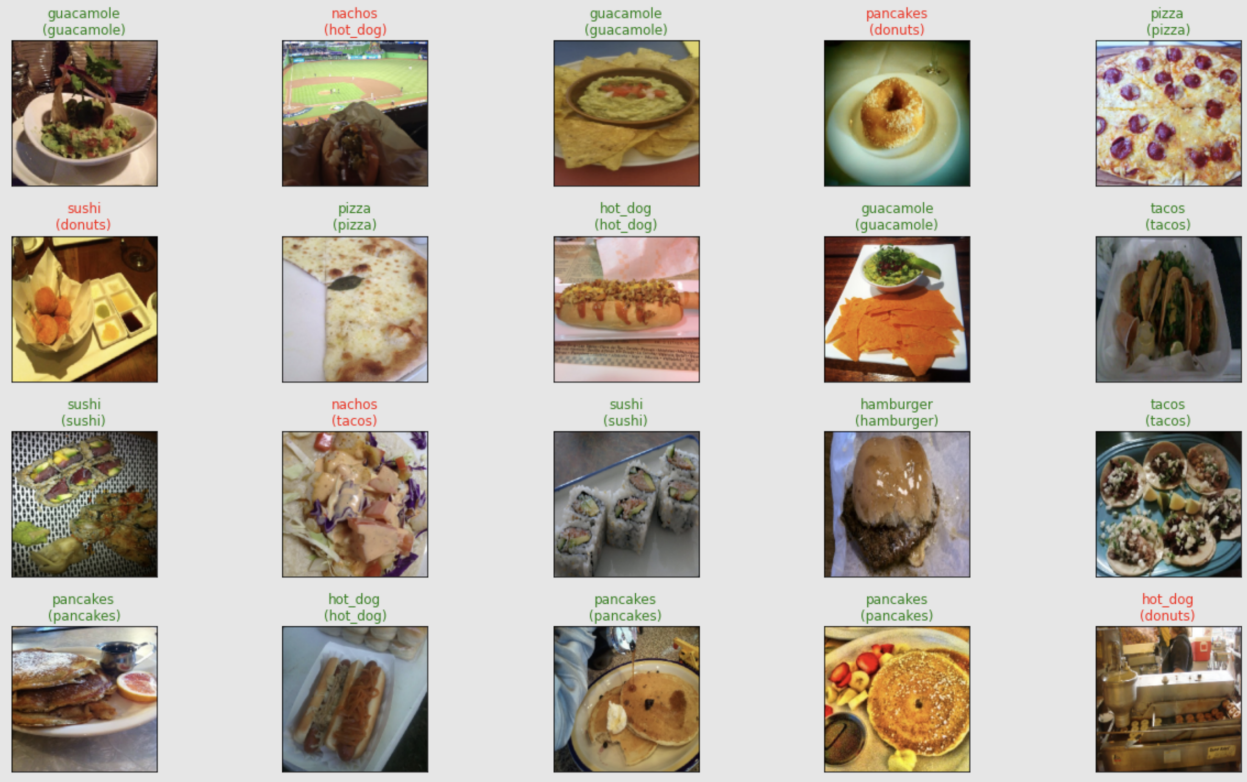

Figure 5: A set of example images and labels using our final trained food recognition model with MobileNetV3_Large as a base model.

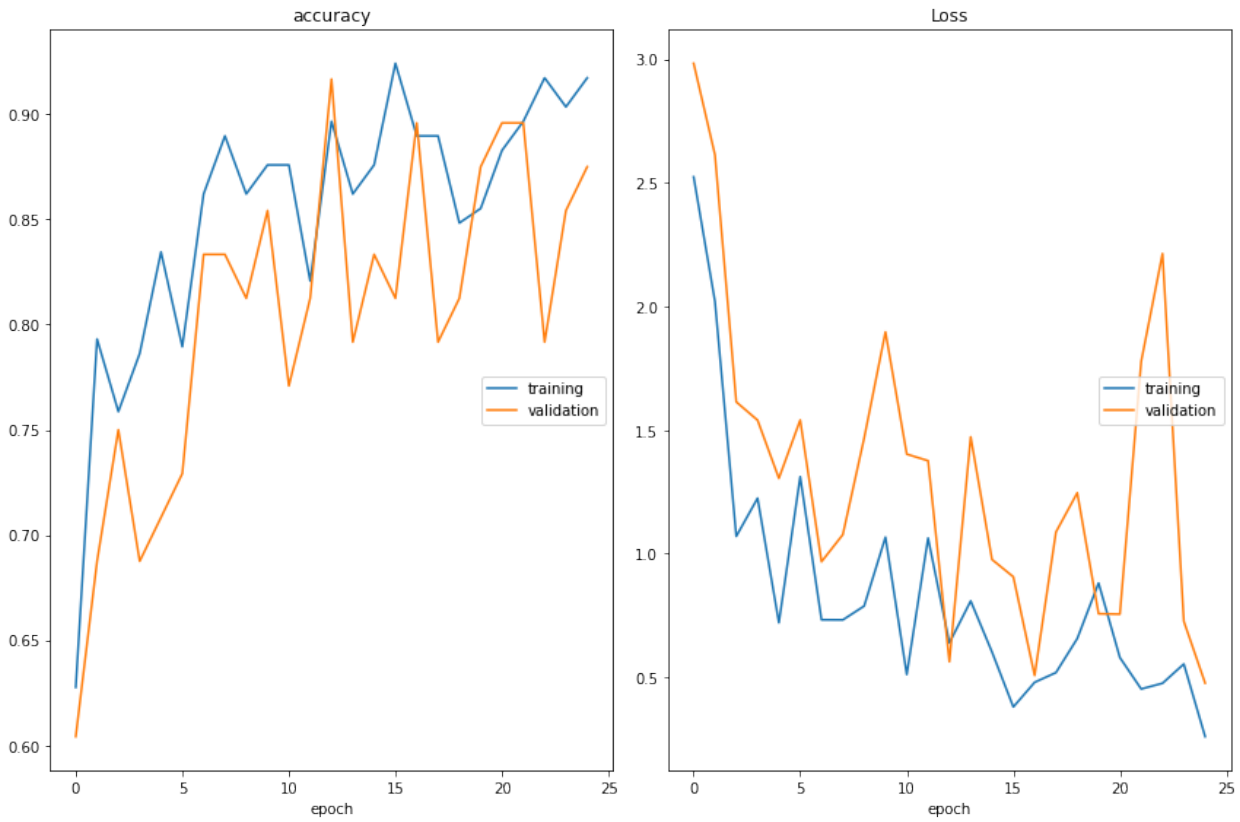

Figure 6: Plots for the training/validation accuracy and loss during the training process for the banana ripeness model with MobileNetV3_Small as a base model.

Overall Categorical Accuracy: 96.15%

	Precision	Recall	F-Score	Support
green	1.000000	1.000000	1.000000	24.0
ripe	0.947368	0.947368	0.947368	19.0
overripe	0.888889	0.888889	0.888889	9.0

Figure 7: A table showing the overall and class-specific scores in the testing stage of the banana ripeness model with MobileNetV3_Small as a base model.

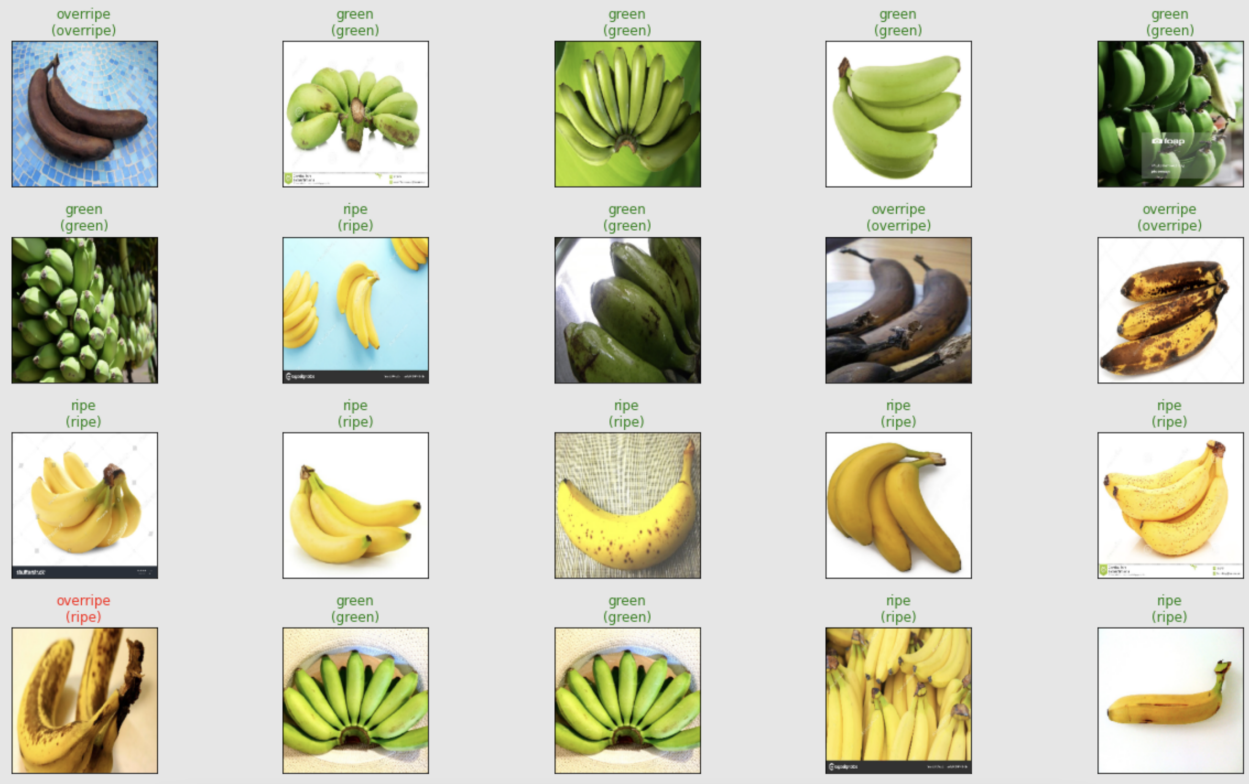

Figure 8: A set of example images and labels using our final trained banana ripeness model with MobileNetV3_Small as a base model.

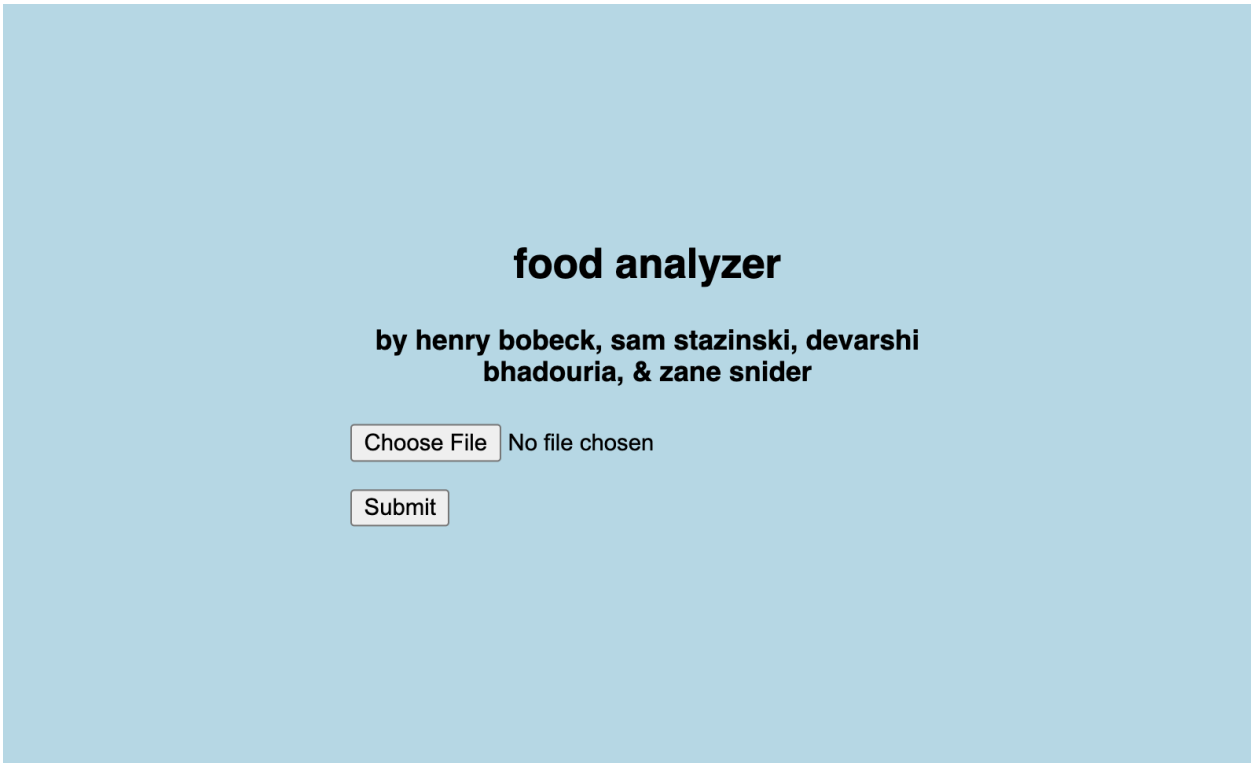

Figure 9: The introduction page for the final web application user interface.

This photo looks like **banana**!

Appears healthy to eat.

banana has 105 kcal, 0.4g fat, & 1.3g protein.

A banana is an elongated, edible fruit – botanically a berry – produced by several kinds of large herbaceous flowering plants in the genus Musa. In some countries, bananas used for cooking may be called "plantains", distinguishing them from dessert bananas.

Figure 10: The results page of the final web application user interface for an input of a ripe banana image.

6 Discussion

Through our time spent working on this project, we were able to delve deeply into the process of AI-based application development and explore through experience what it entails to produce a user-facing, commercially-minded product in the realm of computer vision. In doing so, we developed a greater intuition for not only the existing power of image recognition and deep convolutional neural networks, but also the vast amount of improvement that is still to be made in the field as a whole. While highly accurate models are indeed achievable for problems similar to our own, to train and produce these models takes countless hours of training, huge amounts of training images, and a lot of time spent tweaking and fine-tuning the model's accuracy. Such limitations have definitely played a part in our own development process, causing us to forego early intentions to test more models and configurations due to available time, as well as to collect enough data for more food classes and specificity in ripeness prediction. Part of this is also due to our training system, which we performed using the free version of Google Colab and thus were limited in training runtime before the session was terminated prematurely by the server. Furthermore, we struggled to install tensorflow on different OS versions, and the nature of how we run the trained models in the final product also means that the server running the user interface must have tensorflow installed and functional, which is not always possible. In terms of the overall development process we chose from the beginning, we believe we structured our design process well so as to allow for all of the individual parts to be brought together by the final web application. However, small parts of the design process, such as how we handled the web scraping as well as how we store tensorflow models in their full (very large file size) format, could undoubtedly be handled more efficiently.

7 Conclusion

Given the time constraints faced during our development process, we are highly satisfied with both the efficacy of the deep convolutional models constructed and the appeal and fluidity of our final user-side application. With a top-1 accuracy of 82.07% for our general food recognition model with 11 classes and 96.15% for our banana ripeness model, we feel that our approach to the image-data pipeline and transfer learning has been a success, especially considering the limited data we had access to for training purposes. The final web application works incredibly smoothly, and usage is highly intuitive and streamlined for a user which knows nothing about computer science or computer vision. While there are undoubtedly aspects to improve for future editions, such as the inclusion of far more training data, increased specificity in classes, more fine-tuned testing, and the inclusion of more specialized models for increased informational output, we believe our final product provides ample evidence for the efficacy and future potential of state-of-the-art image recognition techniques for use in consumer-facing apps.

8 References

Online resources used as references for learning and design structure include the following, sorted by category. For scientific papers, see section 2.

- Creating a basic web app with flask
  - https://roytuts.com/upload-and-display-image-using-python-flask/
  - https://www.youtube.com/watch?v=Z1RJmh_OqeA
- Comparison of existing image recognition models by size and accuracy
  - https://towardsdatascience.com/how-to-choose-the-best-keras-pre-trained-model-for-image-classification-b850ca4428d4
- Preprocessing with image augmentation to reduce overfitting + useful visualization of results
  - https://towardsdatascience.com/how-to-choose-the-best-keras-pre-trained-model-for-image-classification-b850ca4428d4
  - https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification/#ImplementationofaCNNinKeras
- Food images data
  - https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/
  - (Hub format) https://app.activeloop.ai/activeloop/food-101-dataset-train
  - (Hub format) https://www.kaggle.com/code/sainikhileshreddy/how-to-use-the-dataset/notebook
- Building an image data pipeline with tensorflow
  - https://cs230.stanford.edu/blog/datapipeline/#building-an-image-data-pipeline
  - https://www.youtube.com/watch?v=VFEOskzhhbc
- Looking into transfer learning with VGG-16, to get better results with less data
  - https://towardsdatascience.com/transfer-learning-with-vgg16-and-keras-50ea161580b4
  - https://medium.com/@1297rohit/transfer-learning-from-scratch-using-keras-339834b153b9
  - https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/