Food Image Classification using Convolutional Neural Networks

Richa Sethi
March, 2021

Introduction

Food is central to the human being's sustenance and pleasure. The significance extends beyond nutrition and health, food is a form of social currency that shapes our lives as it provides us with comfort, relaxation and reward. And now with the continued proliferation of social media platforms, avenues for experience sharing and marketing have dramatically increased, and our digital experience is becoming more and more photo-driven Taking pictures of one's food and posting them on social media has, in recent years, become both a wildly popular and much maligned pastime. This project explores food image classification with convolutional neural networks (CNNs) for better image labeling and clustering by dish, which in turn may improve the recommendation and search flows for a better digital food user experience overall. The ability to properly label and classify food images could lead to better recommendation systems, matching food based on an individual's taste and preferences, or diet. The goal of the project is to, given an image of a dish as the input to the model, output the correct label categorization of the food image.

A food image classifier was built using a deep learning library, fastai. The fastai library sits on top of pytorch, an another hugely popular deep learning library developed by Facebook.

Dataset Overview

The Food-101 dataset for this project was obtained from https://www.kaggle.com/kmader/food41. It consisted of 101 classes of food with 1000 images for each class, leading to a total of 101,000 images. Of the 1000 images for each class, 250 were manually reviewed and cleaned test images, and 750 were intentionally noisy training images, for a total training data size of 75,750 training images and 25,250 test images. However, due to limited GPU and memory even on Colab Pro, of the 101 classes of food, 10 were chosen. To keep the project interesting and challenging, these 10 classes were chosen based on their diverse origin, similar content, ingredients and presentation. The 10 food classes are listed as follows:

1. Bibimbap
2. Chicken curry
3. Macarons
4. Pad thai
5. Paella
6. Peking duck
7. Pho
8. Ramen
9. Samosa
10. Takoyaki

The dataset consisted of images that were very dissimilar in lighting, coloring, orientation and size, and also contained mislabeled images, which were left as is to encourage models to be robust to labeling and lighting anomalies. Figure 1 shows some of the anomalous images from the dataset.

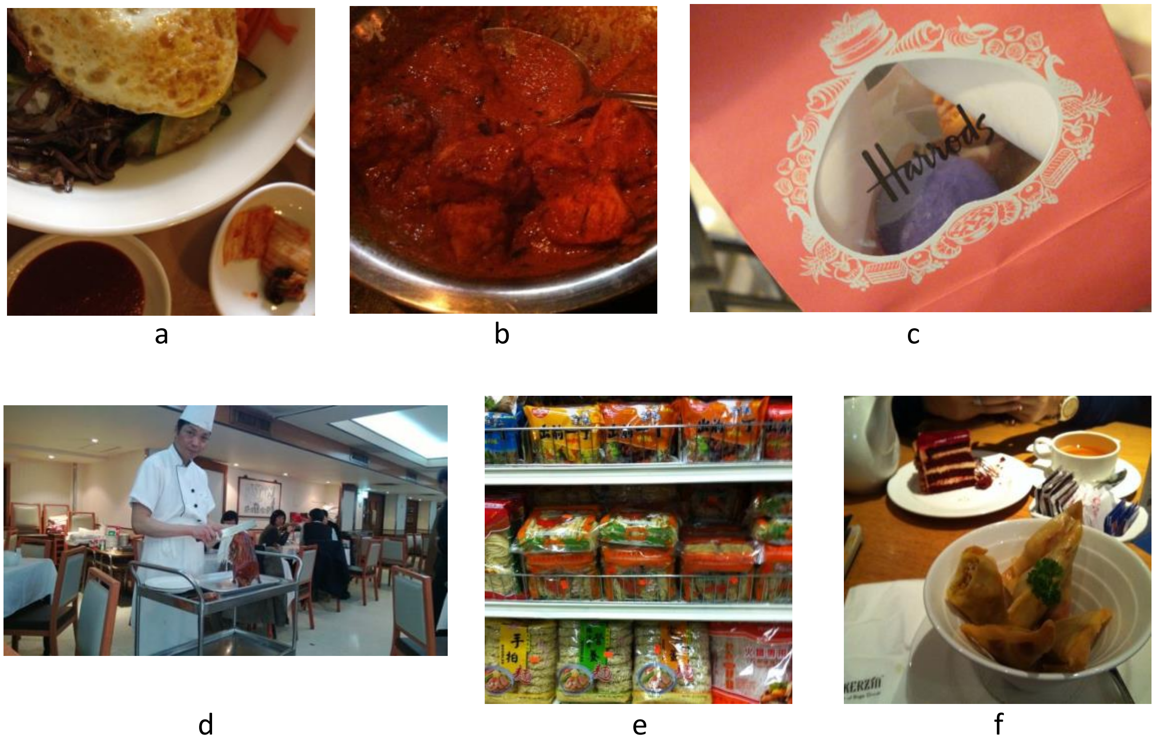


Figure 1: Mislabeled/anomalous images from the train dataset: a) Bibimbap, b) Chicken curry, c) Macaron, d) Peking duck, e) Ramen, f) Samosa

Data Loading

As the first step, fastai Data Block API was used to build a Data Loader from the raw data set to feed into the model. The DataBlock API customizes the creation of a DataLoader by isolating the underlying parts of that process in separate blocks. Here, the inputs are images and the targets are categories, which are represented as blocks. Also, a random split was performed to convert 20% of the train data into validation set.

Data Transformation and Augmentation

As a part of transformation, each image was first resized to the same size. This was done on CPU to prepare for batch transformation. Further transformation was performed on the GPU on a batch size of 64 images. Data augmentation was also performed to help the deep learning algorithm to significantly increase the diversity of data available for training models, without actually collecting new data. Resizing and various data augmentations such as horizontal and vertical flipping, zooming, rotating, lighting, warping, etc. were performed on the training set. Both default and custom values were tried during model building.

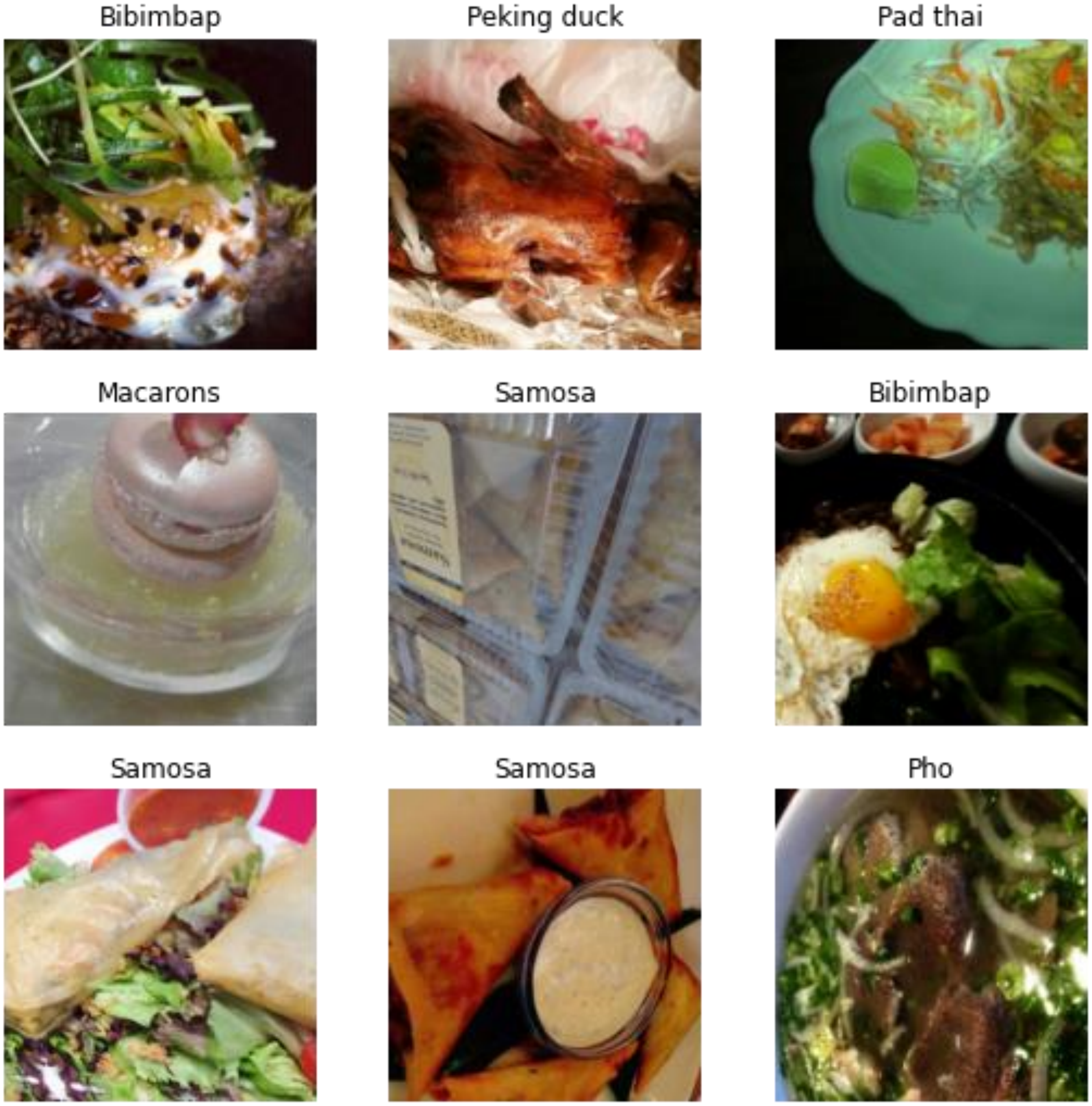


Figure 2: Images after data transformation and augmentation

Transfer Learning

Instead of training the model from scratch, transfer learning was implemented by loading the ImageNet weights into each model and freezing the base layers of each model while removing the top layers that were trained specifically on the ImageNet classes. These top layers were then replaced with trainable layers meant to learn classification on the food classes.

The used pretrained model was ResNet50. The ResNet models are trained on the ImageNet dataset and take advantage of the features learned by those models using deeper architectures and with more training time. In order to use the weights from the pretrained model, the data was normalized to equalize the pixel intensities of RGB channels, i.e. setting the mean of all three channel pixel values to 0 and standard deviation to 1.

Model Training

Model training was performed by first training the network using Leslie Smith's 1cycle policy. However, to use the 1cycle policy, an optimum learning rate was required. To find this learning rate, learning rate was called by using lr_finder. lr_finder works by performing a mock training by going over a large range of learning rates, then plotting them against the losses. A value a bit before the minimum, where the loss still improves is chosen. Our graph for lr_finder on our chosen ResNet 50 model is shown in Figure 3:

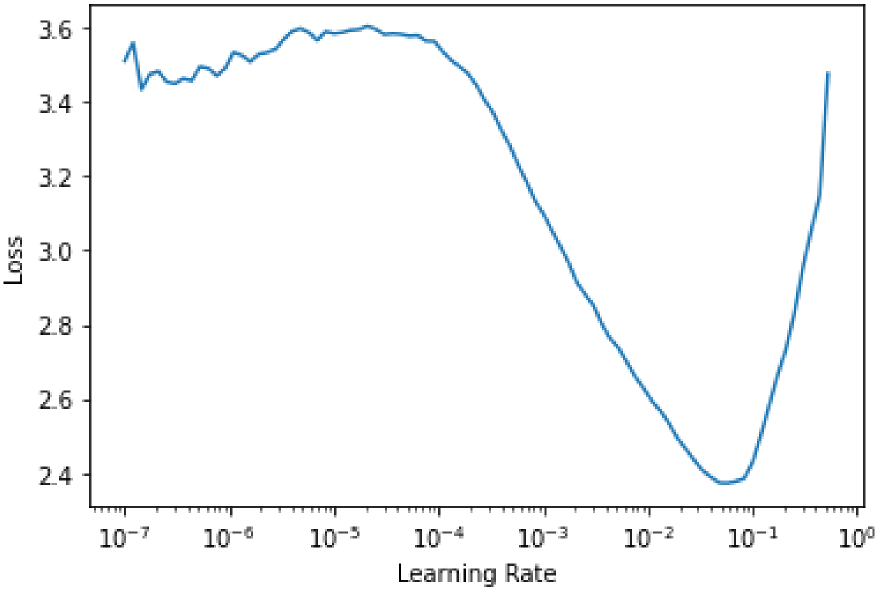


Figure 3: Learning rate finder

Here a learning rate of 3e-3 was chosen for total of 8 epochs was chosen. Next 1cycle policy was applied with the chosen learning rate as the maximum learning rate.

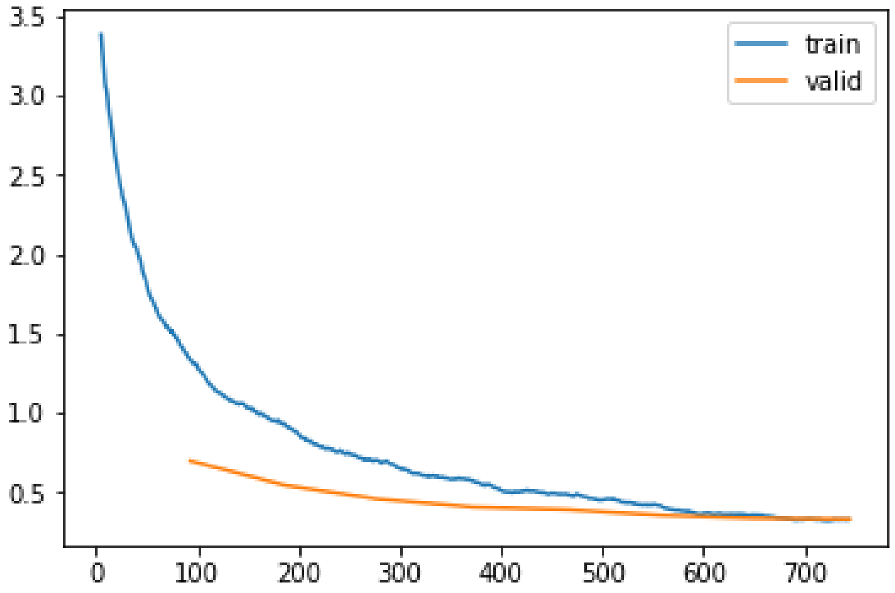
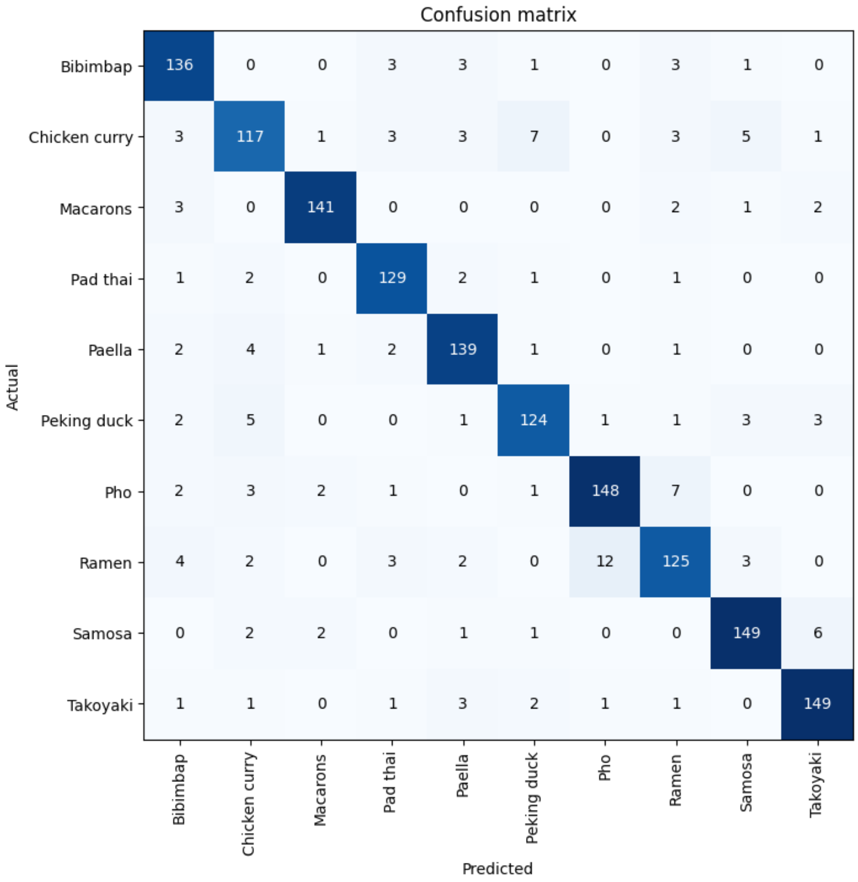


Figure 4: Loss for training and validation set for the batches processed

Figure 4 shows the learning rate for both train and validation set with the number of batches processed. As can be observed, just after 8 epochs, the train and validation set learning rates converged, implying that model should be generalize well to unseen data. The accuracy for the validation set at this point was ~90%. The usual model fitting for transfer learning works like this: train the weights that are closer to the output and freezes the other layers. It is important that for transfer learning, one uses the same 'stats' that the pre-trained model was applied with, eg. correcting the image RGB values with a certain bias.

Figure 5 shows the confusion matrix for all the classes.



Confusion matrix

Actual \ Predicted	Bibimbap	Chicken curry	Macarons	Pad thai	Paella	Peking duck	Pho	Ramen	Samosa	Takoyaki
Bibimbap	136	0	0	3	3	1	0	3	1	0
Chicken curry	3	117	1	3	3	7	0	3	5	1
Macarons	3	0	141	0	0	0	0	2	1	2
Pad thai	1	2	0	129	2	1	0	1	0	0
Paella	2	4	1	2	139	1	0	1	0	0
Peking duck	2	5	0	0	1	124	1	1	3	3
Pho	2	3	2	1	0	1	148	7	0	0
Ramen	4	2	0	3	2	0	12	125	3	0
Samosa	0	2	2	0	1	1	0	0	149	6
Takoyaki	1	1	0	1	3	2	1	1	0	149

Figure 5: Confusion matrix for the validation set

The top classes with maximum loss are as follows:

1. Ramen, Pho: 12
2. Chicken curry, Peking duck: 7
3. Pho, Ramen: 7
4. Samosa, Takoyaki: 6
5. Chicken curry, Samosa: 5
6. Peking duck, Chicken curry: 5

Some of the top losses on the images are shown in Figure 6.

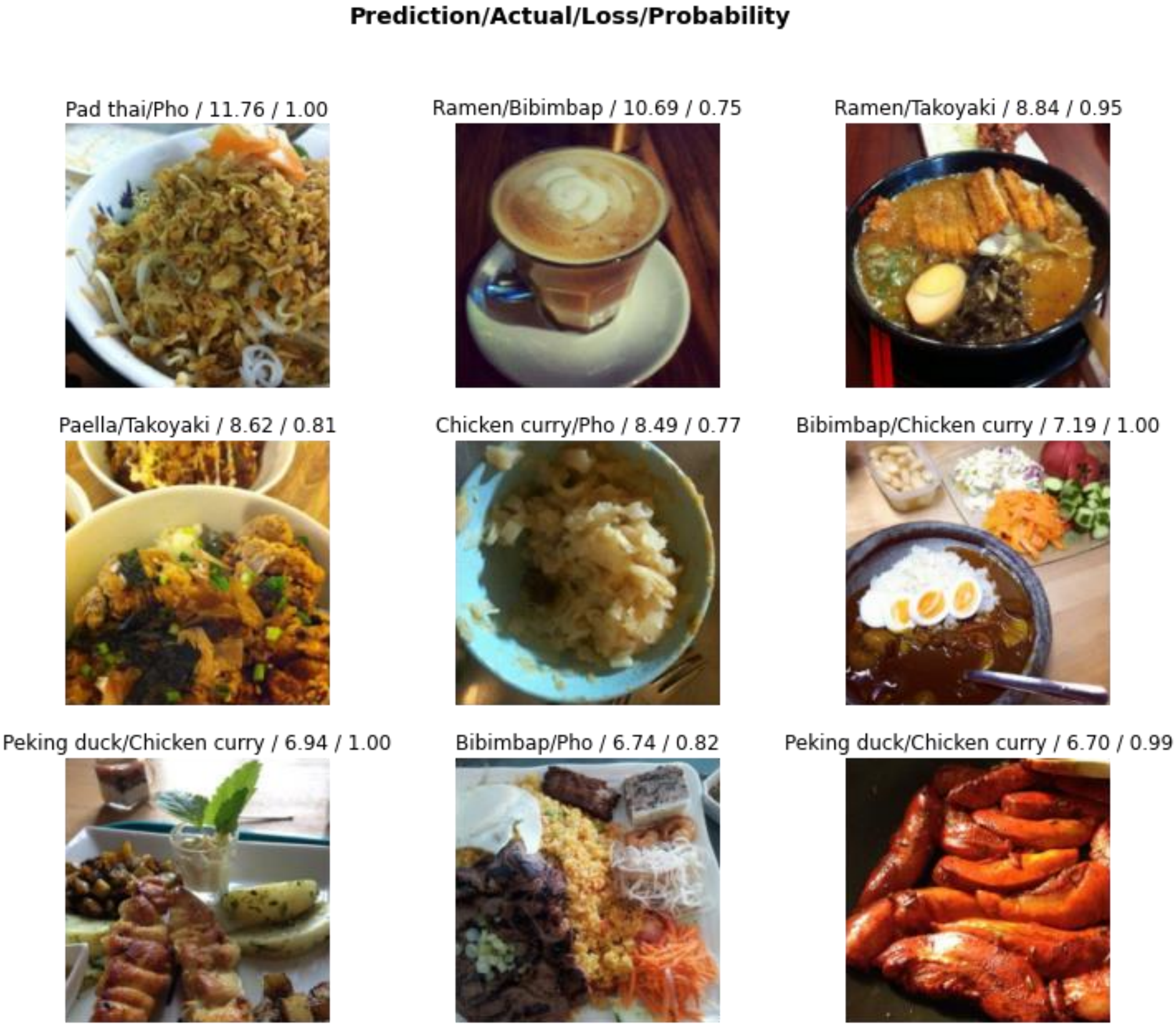

Prediction/Actual/Loss/Probability
Pad thai/Pho / 11.76 / 1.00
Ramen/Bibimbap / 10.69 / 0.75
Ramen/Takoyaki / 8.84 / 0.95
Paella/Takoyaki / 8.62 / 0.81
Chicken curry/Pho / 8.49 / 0.77
Bibimbap/Chicken curry / 7.19 / 1.00
Peking duck/Chicken curry / 6.94 / 1.00
Bibimbap/Pho / 6.74 / 0.82
Peking duck/Chicken curry / 6.70 / 0.99

Figure 6: Images with top losses

As can be observed, all the pictures that were predicted incorrectly were actually either mislabeled or are poor quality pictures.

Visualizing activation layers

In order to understand how the model is actual working, it’s important to visualize all the intermediate activation layers. Let’s take an example of the following image (Figure 7) in the class Samosa which is classified correctly by the model.

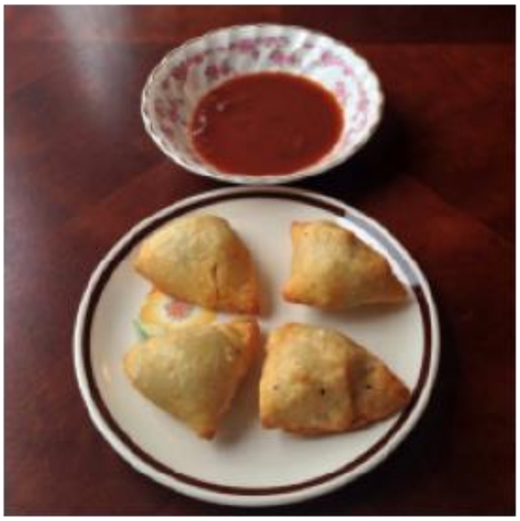


Figure 7: Image of Samosa

All the activations across the key layers for this image can be visualized in Figure 8. As can be observed, the initial layers are creating the shape of the food although there are several filters that are not activated and are left blank. At that stage, the activations retain almost all of the information present in the initial picture. As we go deeper in the layers, the activations become increasingly abstract and less visually interpretable. They begin to encode higher-level concepts such as single borders, corners and angles. Higher layers carry increasingly less information about the visual contents of the image, and increasingly more information related to the class of the image.

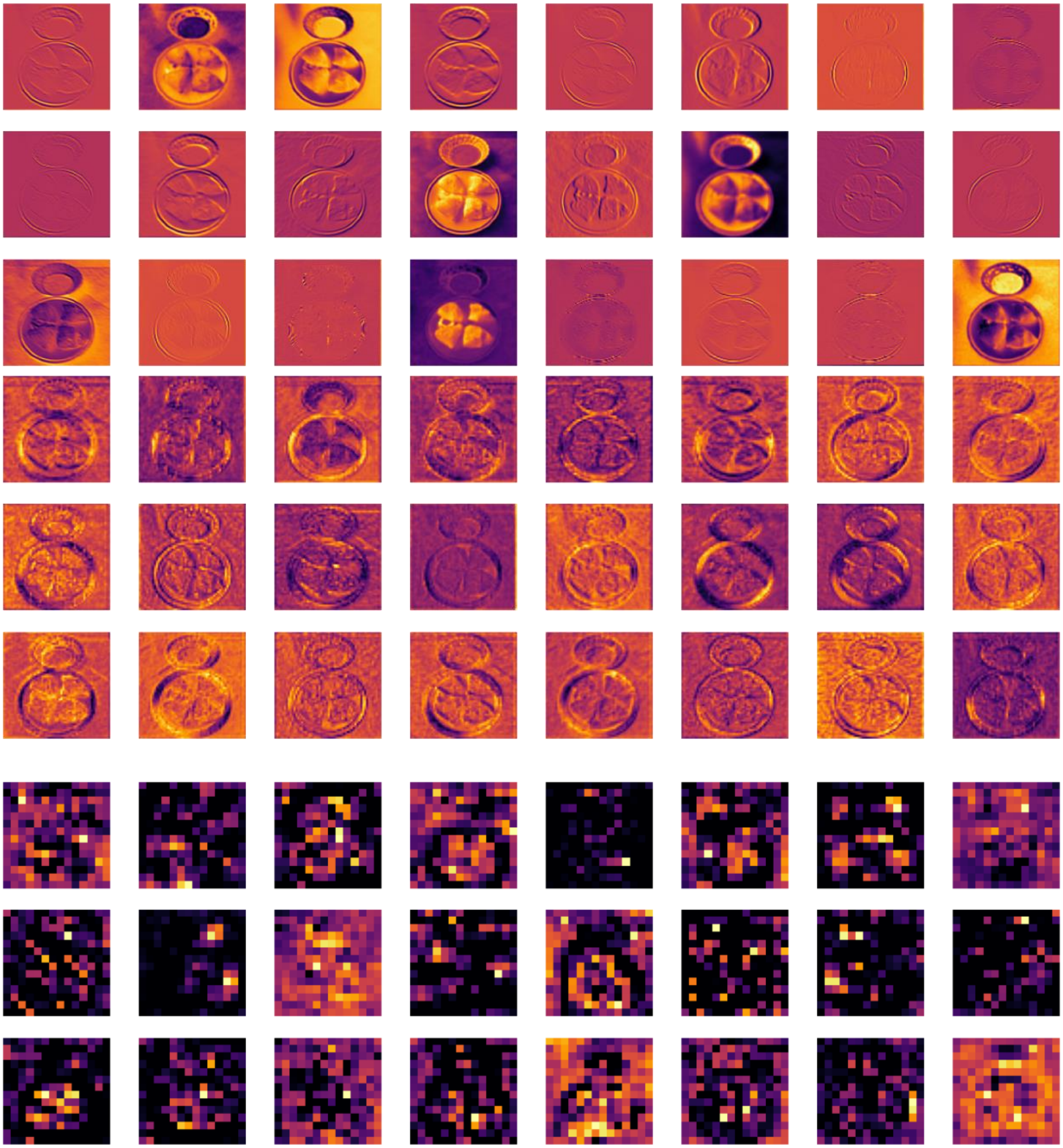


Figure 8: Activation layers of the image

Figure 9 shows the heatmap for the image, this gives us an understanding of where the model is looking to classify the image.

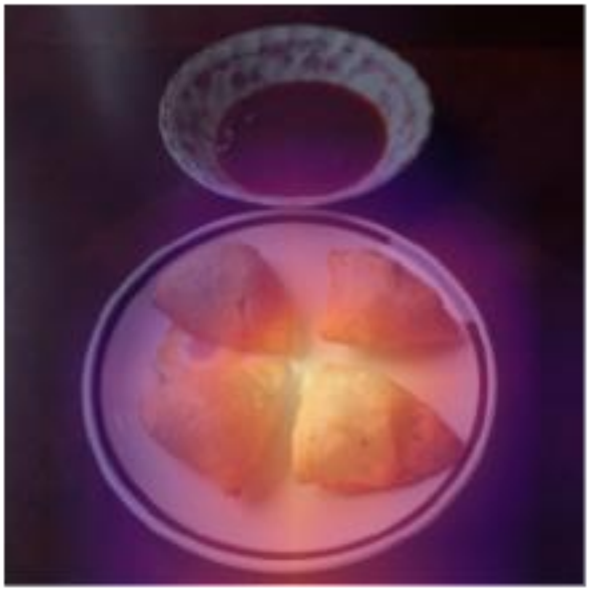


Figure 9: Heatmap of the image

Inference

At this point, we had a trained model showing ~90% accuracy, however, the process does not stop here. The model needs to be put to work by feeding new data, then do the predictions. This is the inferencing stage. The test set containing 250 images from each class was fed into the model, which showed ~94% accuracy. The results from the test set with the actual and predicted names are shown in Figure 10.

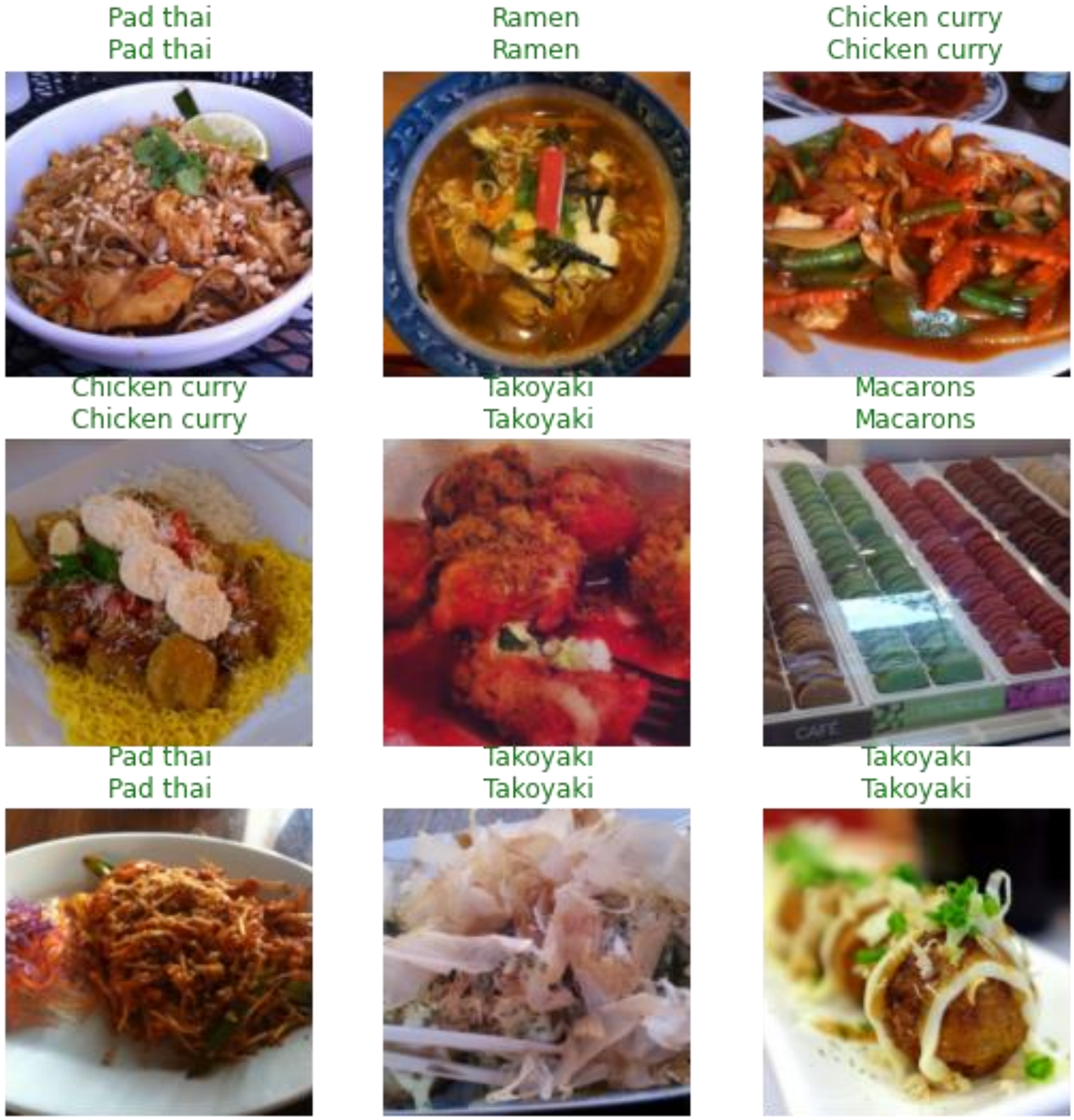
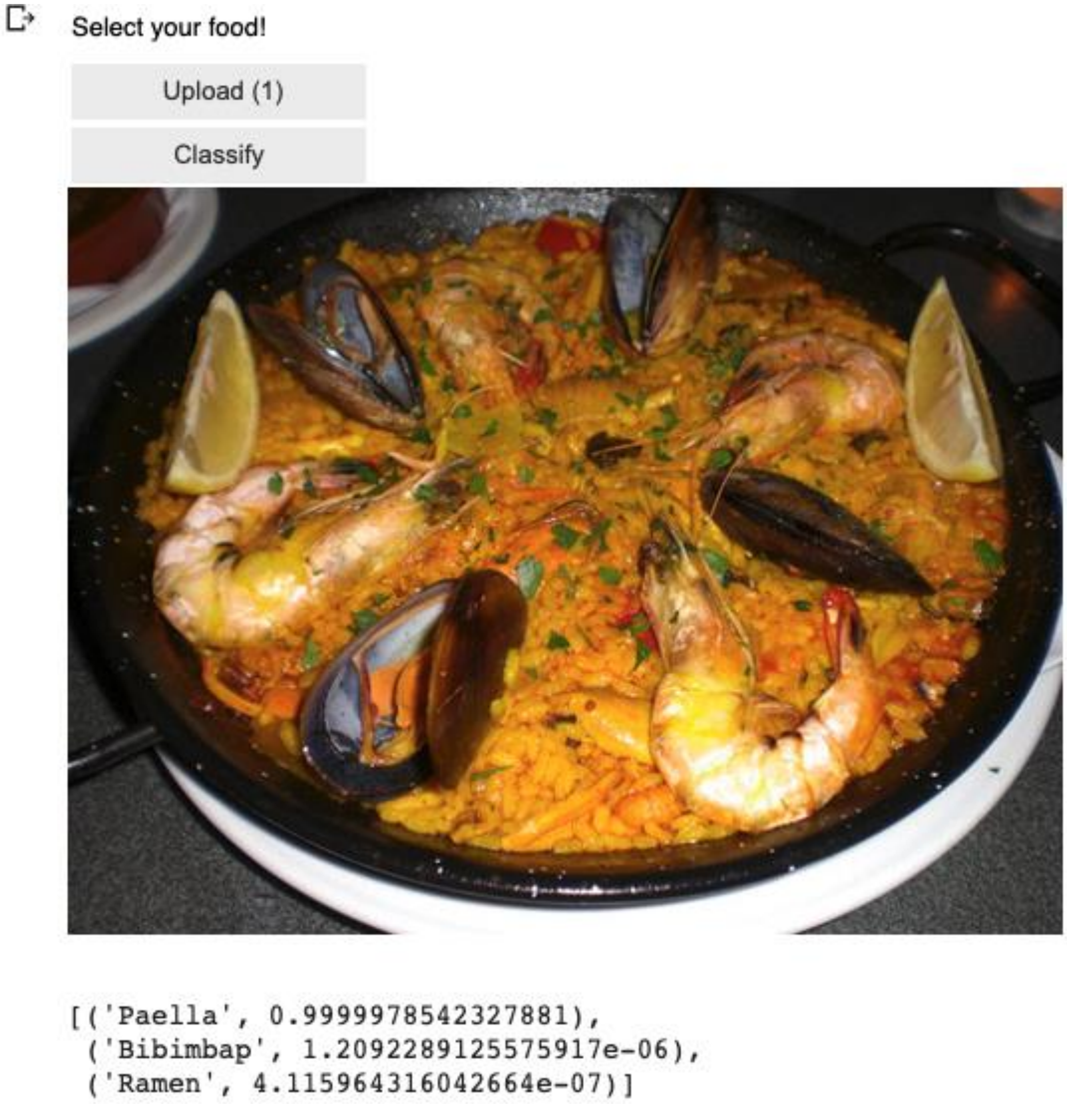


Figure 10: Model inference on test data

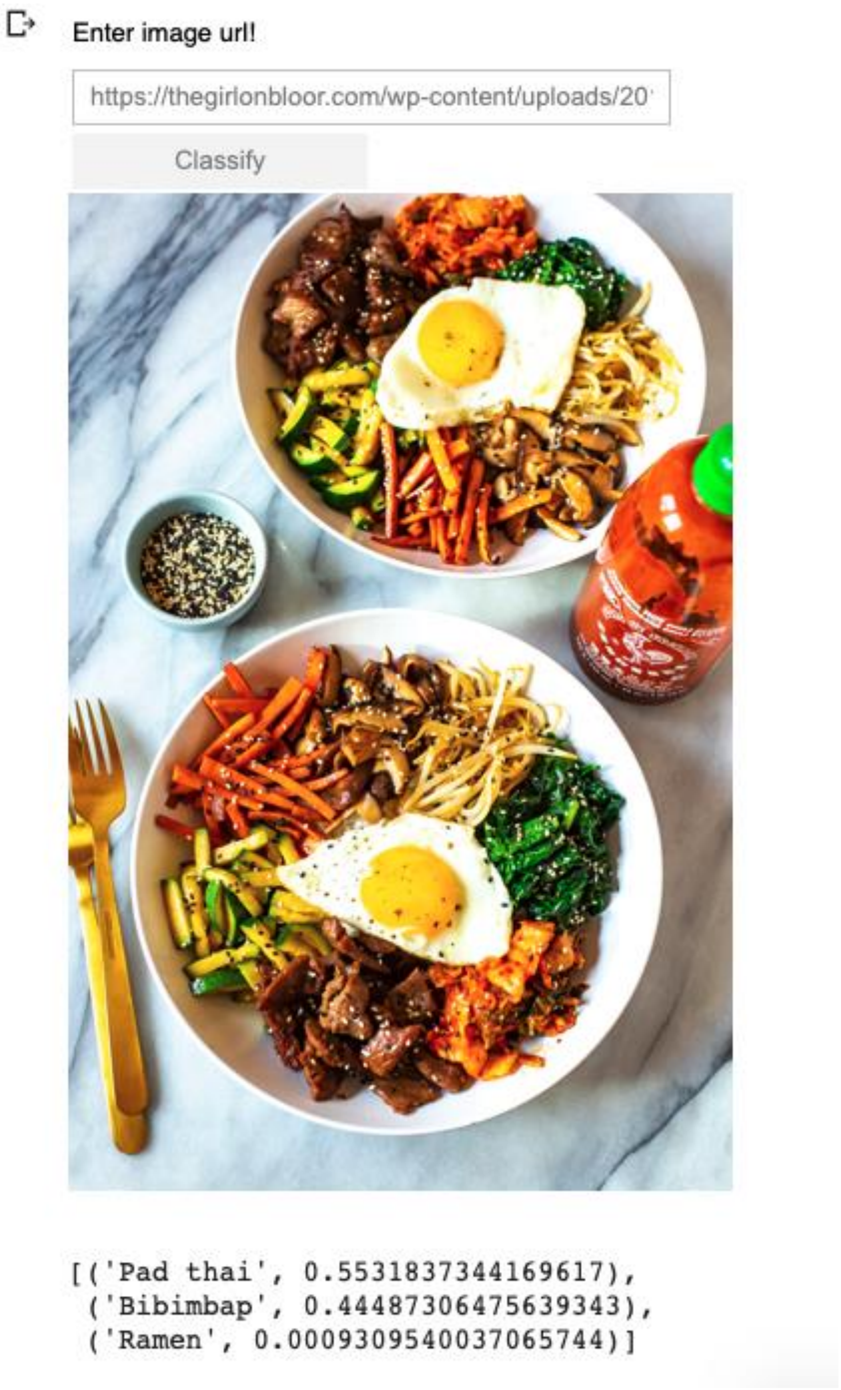
Graphical user interface (GUI) were also created to help users upload their image from the local machine or web to get top-3 predictions on their food image. The GUI imagers are shown in Figure 11 and 12.



Select your food!

```
[('Paella', 0.9999978542327881),
 ('Bibimbap', 1.2092289125575917e-06),
 ('Ramen', 4.115964316042664e-07)]
```

Figure 11: GUI to upload an image from local machine



https://thegirlonbloor.com/wp-content/uploads/20

```
[('Pad thai', 0.5531837344169617),
 ('Bibimbap', 0.44487306475639343),
 ('Ramen', 0.00093095400037065744)]
```

Figure 12: GUI to upload an image from the web

Results

Transfer learning was used to identify and classify 10 classes of food taken from Food-101 dataset. ResNet 50 with image transformation and augmentation showed the best results with accuracy as high as 94%. Transfer learning was successful because the earlier pre-trained layers had already learned a lot of the general features needed to identify food images. Additionally, the intermediate activation layers were visualized to understand how the model works on a granular level and which features help the model identify the food class. Lastly, GUIs were build around to result to predict the food class along with the probability from the local machine as well as from web.

Future Work

1. Use more deeper network or newer structure, such as ResNet200 or EfficientNet to train the model
2. An exhaustive hyperparameter search would have been a more empirical approach but due to computational and time constraints, it wasn't performed for this capstone and has been left to do as future work.
3. Model performance could be further improved by adding bounding boxes to the images
4. Train models to recognize images within a subset of food (e.g. vegetables vs fruits vs bread vs noodles vs. cakes etc.), since many of the errors from the model are a result of confusing similar food items with each other (e.g. pho vs ramen).
5. Extend the model to out ingredients, recipe and nutritional information along with the food prediction