CS230

Complex Food Classification

I-Sheng Yang, Bryce Long, Priyal Mehta
Department of Computer Science
Stanford University
ishengy@stanford.edu, molohov@stanford.edu, priyalm@stanford.edu

Abstract

Food image classification has many uses in everyday tasks, ranging from search to tagging. In our project, we explored the complexities associated with training a neural network to perform food classification. We experimented on state of the art deep convolutional neural network architectures before settling on the InceptionV3 architecture. Our final model uses a weights pre-trained on ImageNet and incrementally trains it on our dataset. The dataset is based on Food-101 and has an additional 4000 images added from a web crawling. To reduce the large variance gap in our model, we tuned our hyperparameters and added on-the-fly data augmentation, which resulted in some variance reduction. Based on visualizations of the network's filters and error analysis per class, we conclude that the network's learning capacity is too high for our dataset. It needs a lot more data and training to be able to classify our validation set with high accuracy.

1 Introduction

Our project tackles the problem of classifying food dishes using image classification neural networks. Based on an input color image, a neural network predicts the probability of predefined output food classes, and selects the one with the highest probability as the predicted class. Food classifying is useful in many everyday applications. Photo search software like Google Images or Photos can use food classification to search for specific dishes in a large dataset of images. In social media, pictures of food are plentiful, and being able to predict the dish in an image could allow faster tagging of the image. Finally, dieting and nutrition tracking applications can benefit from food image classification by making the tagging process for each meal as simple as taking a photo, rather than looking up dishes in a large database.

Classifying images of food is difficult because of the complexity of food's appearance, as the same dish can have many different appearances due to presentation, size, and ingredients. To complicate matters more, images of the same dish can look drastically different depending on the lighting conditions of the room and the angle at which it was taken.

2 Related work

We have seen many implementations of food classification in the literature. The basis for most of them is a research project named Food-101 [1]. The project introduced a novel computer vision method and also published a challenging dataset of 1000 images for each of 101 popular food dishes (which is what our dataset is based upon). The novel computer vision technique in Food-101 is described as a "weakly supervised mining method", which is able to beat other classification methods (except for convolutional neural networks), with an average accuracy of 50.76%.

Afterwards, other projects have used Food-101 as a dataset to train and compare performance. These projects almost all use CNNs to do food classification. A previous CS230 project named DeepMeal [2] used an Inception CNN and YOLO algorithm to perform image classification and localization, achieving better accuracy than Food-101. Other projects using the Inception CNN architecture for classification such as Rodriguez [3] and Hassannejad et. all [4] have achieved 86.97% and 88.28% accuracy respectively. Our project also uses the Inception CNN architecture for classification, albeit the newer V3 model. Other CNN architectures like residual networks achieve even higher accuracy on Food-101's dataset. Lee et. all [5] achieved 90.14% accuracy with their ResNet200 network, while Martinel et. all [6] achieved 90.27% accuracy with their Wide-Slice Residual Network. These results demonstrate that CNNs (particularly deep CNNs) are the best method currently known to tackle food classification, due to their ability to learn complex image features of the dataset.

CS230: Deep Learning, Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Features

We started with the Food-101 dataset [1], which contains 101 classes of food with 1000 images per category. We split the images such that, for each category, 900 images are used for training, and 50 each are used for validation and test (90/5/5 split). We supplemented the training data set with additional images downloaded using Google image search. After supplementation, the dataset had 99802 training images and 5050 images for validation and test. Additionally, we cleaned our supplemented dataset to ensure they only contained color images in the JPEG format. Food-101, however, was not cleaned and does contain some irrelevant images per class. This was noted by the creators of the dataset.

The images in our combined dataset did not have a standardized resolution, therefore we preprocessed them such that all images had a resolution of 150x150x3. We also normalized the RGB values by dividing the values by 255. Three sample images are included below in Figure 1. There are no additional features fed into our neural network.

(a) garlic bread

(b) paella

(c) strawberry shortcake (not cleaned)

Figure 1: Sample images from our dataset.

4 Methods

4.1 Deep Convolutional Neural Networks

Convolutional neural networks (CNNs for short) learn filters of varying size that, when convolved with an input image, detect features of the image. Cascading many of these filter layers together (in conjunction with pooling layers to help reduce dimensionality) can result in a network that learns very complex image features. Deep convolutional neural networks such as the Inception architecture contain many layers, motivating the 'deep' descriptor. For example, InceptionV3 is over 300 layers deep and contains 22 million trainable parameters, as shown in Figure 2.

4.2 On-The-Fly Data Augmentation

To decrease over-fitting and to get the most out of our limited dataset, we employ on-the-fly data augmentation. In each epoch, we augment the original image by applying any number of these modifications: noise, random rotation, shear, horizontal/vertical shifts, zooming in, and flipping. Additionally, to improve compute performance (especially as the dataset size increases), we created a data generator that only loads the images that are required for the batch being trained. The data generator contains the augmentation code and runs on multiple cores in real time and feeds the data directly to the model when it is ready to train on a new batch.

5 Experiments/Results/Discussion

5.1 Metrics, Baseline Model and Architecture Search

The main metric for evaluating the performance of our network is accuracy. We always evaluated the models on the same test set created during the initial 90/5/5 dataset split.

Our baseline model is a InceptionV3 CNN connected to a fully-connected layer to match the number of food classes of our dataset. The network was pretrained on ImageNet. The goal of the baseline was to try and match or improve the results of Rodriguez [3] and DeepMeal [2], both of which used Inception CNNs. The best performance on the baseline model was 61.67% test accuracy, with the model able to severely overfit the training set after 30 epochs. In the baseline metric, we used only half the training set without any additional downloaded images and did not employ any data augmentation techniques.

Based on the related work of [3] [4] [5] [6] , we explored other deep CNN architectures and briefly evaluated them. Inception-ResNet-V2 showed promise in its performance compared to older Inception networks and other high performance CNNs [7]. Additionally, the combination of Inception modules and residual networks into a new architecture seemed promising for our

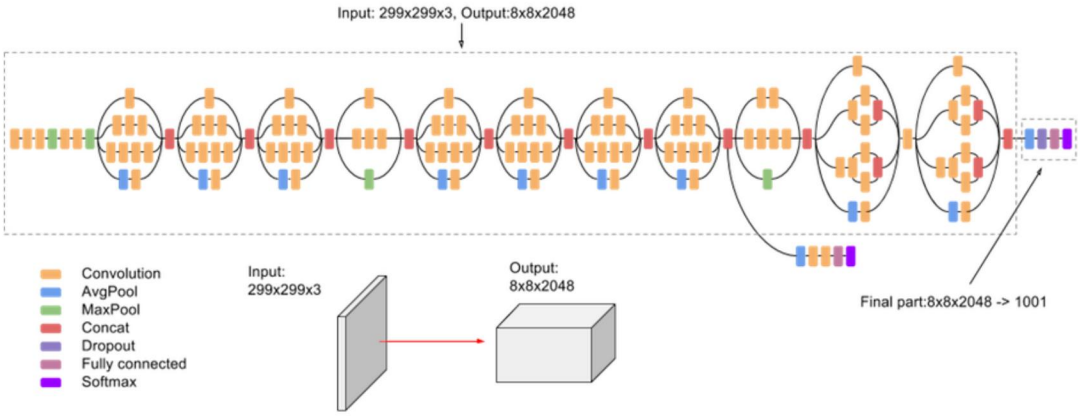


Figure 2: Inception V3 architecture

project, given the success of both in the literature. However, the accuracy of Inception-ResNet-V2 after 30 epochs of training was not any better than our baseline model. Further, Inception-ResNet-V2 was far slower to train because it has over twice as many parameters as InceptionV3 (54 million compared to 22 million). Thus, for the final model, we continued with InceptionV3.

5.2 Hyperparameter Tuning

At our input image resolution (150x150x3), the input feature set is quite large; thus, our training time was also long. We found that it takes 15 to 20 minutes per epoch on AWS if we trained with the entire dataset. Thus, for hyperparameter tuning, we used only half of the images from Food-101 to iterate faster. Our results are summarized in Table 1.

Table 1: Select highlights from our hyperparameters search.

Optimizer	**L2**	**Minibatch Size**	**Epoch**	**Val Acc**
SGD, LR 0.01, Momentum 0.8	0.005	64	30	0.6167
RMSprop, LR 0.01	0.005	64	30	0.2380
Adam, LR 0.01, β_1 0.9, β_2 0.999	0.1	64	30	0.2916
SGD, LR 0.01, Momentum 0.8	0.2	64	30	0.5720
SGD, LR 0.01, Momentum 0.8	0.005	32	30	0.6196
SGD, LR 0.01, Momentum 0.8	0.005	128	30	0.5805
SGD, LR 0.01, Momentum 0.9	0.005	32	30	0.5967
SGD, LR 0.2, Momentum 0.8	0.005	32	30	0.6012

During the search process, RMSprop and Adam optimizers showed low rate of learning, causing us to favor SGD instead. With SGD, our model was overfitting the training data severely, with most of them achieved over 99% training accuracy. We also identified a variance problem, as our validation accuracy hovered around 60%. Tweaking the L2 regularizer value had no pronounced impact on the variance. Thus, we opted to take the most promising hyperparameter combination and increase our training dataset, along with data augmentation, to try to improve variance. We noticed the training accuracy decreased after we started employing data augmentation techniques. We believed this was due to the input features becoming less standardized, so we needed to decrease the momentum for the neural network to quickly adapt to the changing features. We then redirected our optimization effort towards tuning the momentum value. Table 2 contains highlights from this process.

Our variance decreased when we started training with the full Food-101 dataset, but not by much. We then attempted to supplement the training image database with additional images downloaded off Google image search, which brought the validation accuracy up by another 1%. With a lowered momentum value, train and validation accuracy increased by a few percentage points, again overfitting the training set. We concluded that further tweaking of the momentum parameter offered little promise. Also worthy of note is the L2 Regularizer value. When using half-sized training sets, we observed having a low L2 value resulted in higher validation accuracy. However, when using full training set, having a low L2 value decreased validation accuracy relatively to using a higher L2 value. It is likely that our half-sized training set was not big enough for L2 regularizer to have a noticible impact on reducing variance.

We then tried freezing the weights at certain layers of the Inception network when performing training, hopefully forcing the network to learn to utilize generic set of features inherited from ImageNet weights in its decision making. For reference, our

Table 2: Selected highlights from our hyperparameters fine-tuning. Optimizer was SGD with learning rate 0.01.

Data Supplement	PreTraining	Momentum	L2	Epoch	Val Acc
None (Baseline)	ImageNet	0.8	0.2	30	0.6090
Augmentation	ImageNet	0.8	0.2	30	0.6130
Augmentation	ImageNet	0.8	0.2	30	0.6190
Download + Aug.	ImageNet	0.8	0.2	30	0.6290
Download + Aug.	from 0.6290 run	0.7	0.2	5	0.6495
Download + Aug.	from 0.6495 run	0.6	0.2	5	0.6531
Download + Aug.	ImageNet	0.6	0.2	30	0.6569
Download + Aug.	ImageNet	0.6	0.005	30	0.6332

model contains 314 layers in total. The results show that food images are sufficiently different from ImageNet images such that low level features need to be tuned to our dataset. All our experiments, performed using SGD with learning rate=0.01 and momentum=0.8, resulted in sub-par validation accuracy, as shown in Table 3.

Table 3: Selected highlights from our weight freezing experiments.

Data Supplement	Frozen Layers	Epoch	Val Acc
Download + Aug.	1-250	30	0.2868
Download + Aug.	1-80	30	0.4415
Download + Aug.	51-130	30	0.5263

At this point, we decided that we could not decrease our variance without significantly more data. We consulted with our mentor and concluded that the Inception network's learning capacity is too high for our input dataset. We then turned our attention to visualizing the network in our attempt to confirm our theory.

5.3 Visualizing Network Outputs

To try and understand the trained values of the network, we used deconvolution techniques to visualize the filters of intermediate layers learned by the network, as well as visualizing what the network thought a certain output class should look like. All images were generated using the keras-vis package.

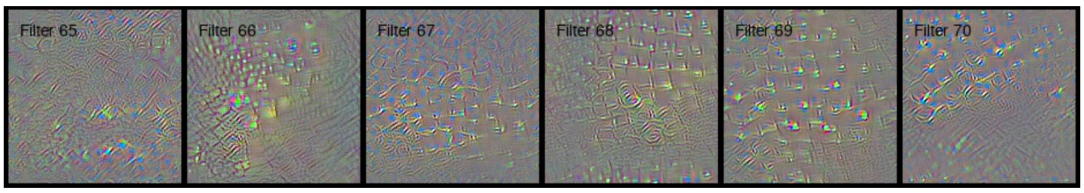


Figure 3: Visualization of filters in the last convolutional layer

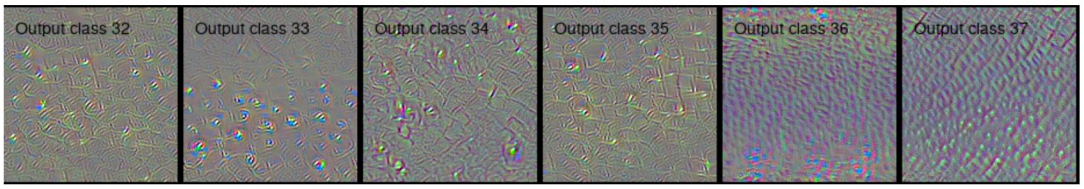


Figure 4: Visualizations of output classes

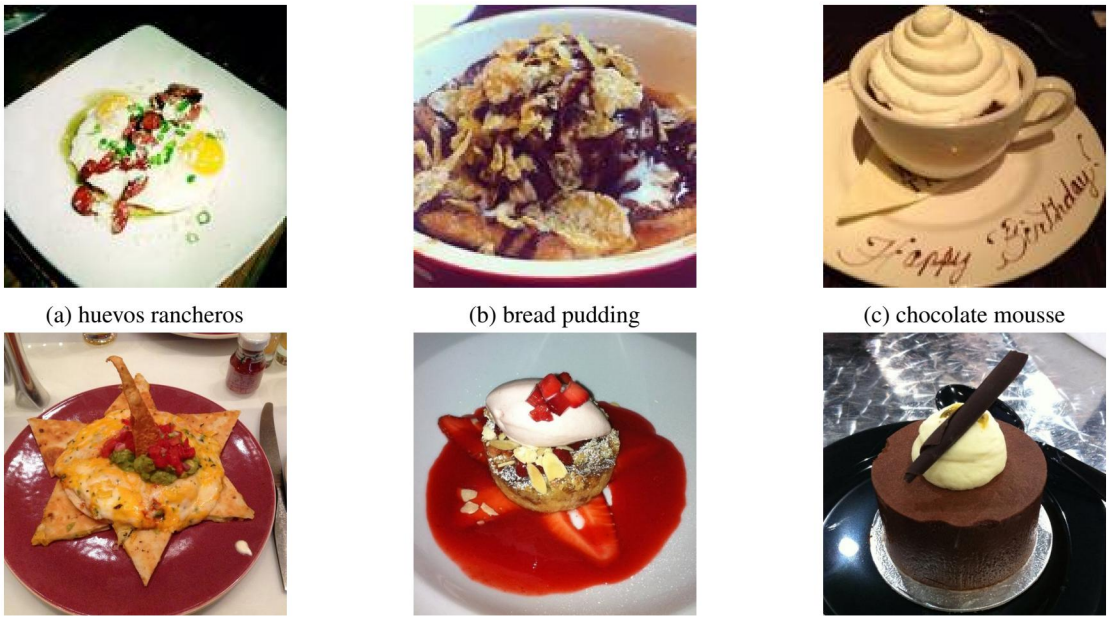
In Figure 3 it is possible to see some high level image features being shown in the last convolutional layer. However, none of the features look like features resembling food. Indeed, when trying to view the visualizations of output classes of the network, many of the outputs do not result in meaningful images (Figure 4) and some do not even converge (see Output class 36 and 37). This may demonstrate our network's lack of ability to learn the complex features of food images.

5.4 Error Analysis

Table 4: Top 8 and bottom 8 performing food classes.

Food	Acc.	Food	Acc.	Food	Acc.	Food	Acc.
Edamame	0.98	Miso Soup	0.92	Pho	0.90	Lobster Bisque	0.90
Creme Brulee	0.88	Spaghetti Carbonara	0.86	Seaweed Salad	0.86	Oysters	0.86
Steak	0.40	Omelette	0.40	Foie Gras	0.40	Bread Pudding	0.38
Apple Pie	0.38	Tuna Tartare	0.36	Huevos Rancheros	0.34	Chocolate Mousse	0.30

In Table 4 we show the top 8 and worst 8 classes based on accuracy. Based on the results, the model performs really well on simple looking food classes. For example, classes like pho, miso soup and edamame have images with consistent, well defined shapes and boundaries and relatively consistent presentation of the dishes. On the other end, classes like huevos rancheros and bread pudding contain many different presentations of the dish with drastically different ingredients, shapes and colors (Figure 5). From this analysis, we could see that our network was not able to learn all of the features of these classes to classify them well.



(a) huevos rancheros (b) bread pudding (c) chocolate mousse

(d) huevos rancheros (e) bread pudding (f) chocolate mousse

Figure 5: Complex images with huge possible variations.

6 Conclusion/Future Work

Throughout this project, we explored various CNN architectures and performed hyperparameter search to help find the ideal configuration to help us achieve a high accuracy model. InceptionV3 was the best model to use for us to progress on due to its accuracy and training time relative to deeper CNNs. However, throughout the search, we consistently had a variance problem. Through visualization of the learned filters, we saw that the network is not really learning meaningful features of the dataset. During error analysis, we see that the network does well on image classes with high consistency and simplicity and does poorly on image classes with lots of variance. This could be a factor of the dataset itself, but we believe it can be addressed by incremental training using additional data, as the network is able to overfit the training set yet the weights did not generalize well to the validation set.

To improve our results, we would first enhance our dataset dramatically. To obtain high accuracy, we would try running for more than 30 epochs. Given the success of dedicated residual networks in previous work, we could also try using state of the art residual networks, instead of the Inception-Resnet model we tried. Additionally, we could attempt to augment the data more aggressively to further force the network to use a diverse set of features in its classification. Also worthy of further investigation is to utilize different optimizers when performing incremental training.

7 Contributions

Bryce created the initial dataset loading functions and the first draft of the baseline model, as well as performing architecture search in the literature and on training. He also generated the neural network layer visualizations using keras-vis. I-Sheng worked on the hyperparameter search, and did most of the model training. He also created the web crawler to download additional images from Google into our dataset (as well as manually cleaned them), and implemented restoring of weights from a previous run. Priyal implemented the on-the-fly data generator and augmentation using Keras methods. He also implemented the auto-saving of the highest performing model per epoch, as well as tools to analyze the test set predictions including graphing outputs for each class. All members contributed equally to the final report and the poster.

8 Code

Our baseline and final models are available at: https://github.com/molohov/cs230-project

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[2] Alexander Iyabor. Deepmeal: Food recognition and localization with yolo. 2018.

[3] Patrick Rodriguez. Deep learning, applied. project #1, January 2017. blog.stratospark.com [Online; posted 22-January-2017].

[4] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. Food image recognition using very deep convolutional networks. In *Proceedings of the 2Nd International Workshop on Multimedia Assisted Dietary Management*, MADiMa '16, pages 41–49, New York, NY, USA, 2016. ACM.

[5] Keun dong Lee, DaUn Jeong, Seungjae Lee, and Hyung Kwan Son. 딥러닝을이용한음식영상분류. GPU Technology Conference Korea, 2016.

[6] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Wide-slice residual networks for food recognition. *CoRR*, abs/1612.06543, 2016.

[7] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.