Food Recognition Using Convolution Neural Network

Subhojit Roy Bardhan(193050002) Debtanu Pal(193050043),
Aakash Banerjee(193050034) Samrat Dutta(193050026)

Indian Institute Of Technology, Bombay

Abstract.

This report was written as a part of CS-725(Foundations of Machine Learning) course Project. The Project deals with a simple problem of Recognizing Food Images using a Convolutional Neural Network Architecture. The base dataset was a custom dataset taken from [1] as well as we created a small custom dataset from [2] and trained our model on a combination of both. We explored the fact the Data augmentation can significantly improve the performance of a model on a dataset as it provides extra data for training by some operation on original images. Finally, we play with the Network Architecture a bit and try a different type of learning rate with the given model and dataset and compare the relative performances of these.

1 Introduction

As we are moving forward in time more and more low cost imaging devices is available in the form of smartphone cameras, normal cameras etc. With the advent of Artificial Intelligence everywhere, computer vision community is constantly working on imaging devices to incorporate automatic object detection with high accuracy. CamFind, Google Lens are some apps being used worldwide for image recognition.

Among various objects being detected food has been of particular interest in recent times. With more and more people becoming health conscious it will of great help if type of food can be easily detected and even better if calorie count,fat content etc can also be shown. People can use their cameras to keep track of their food habits.Also someone travelling to a foreign land may be interested in finding out what a particular food may be. Someone suffering from a particular disease may not be allowed to have some particular food, or someone may be allergic to some food. Food recognition may be used here.

Our project focuses on Food Image Recognition using Convolutional Neural Network. Convolutional neural network is the new meta for object detection/classification and has almost replaced Support Vector Machine(SVM) everywhere. CNN uses the same weight vector across various parts of the image by using the properties of translational invariance and locality. This reduces the storage requirements for the parameters and makes CNN feasible.

2 Main Goals

Our main goal is to train a deep Convolutional Neural Network to Recognize images of food using a Dataset that consists of Single Food Images [1].

We primarily intend to only train a model to Recognize Images of Food as accurately as possible from our newly created Dataset in this Project.

We also planned to modify/tweak the dataset a bit to include the calorie or price in the dataset and also predict the calorie/price using the improved model, but unfortunately we were unable to complete that within our stipulated time.

3 Related Literature

As computation power has evolved with time, application of neural networks and its derivative has become widespread. Y. Lu [1] has applied a Convolution Neural Network(CNN) with 3 convolution layers with kernels of size (7x7), (5x5), (3x3) with 32 features in the first, 64 features in the second, and 128 features in the 3rd layer. Following this a hidden layer with 128 neuron units were used and the output layer consisted of 10 unit. On all layers other than the output layers a 'Relu' activation was used, whereas a 'Softmax' activation was used in the output layer. The learning rate was annealed proportional to the magnitude of loss at the current epoch.

In their paper E. Aguilar *et al.* [3], proposed a food detection architecture. Given an input image, it classified it into two class, food or non-food items. It used the GoogleLeNet model to first extract features from the input images, followed by which Principle Component Analysis(PCA) was performed to reduce the dimensionality of the features extracted from the last step. It ensures that the greatest variance lies in the first axis so as to capture the major features but at the same time reduce the dimensions required. From the features extracted at the last stage, it is fed to a Support Vector Machine(SVM), which is trained by GridSearchCV strategy on 3-folds, which is finally used to classify the input images into two classes from the features extracted using PCA.

A. Myers *et al.* [4], applied food detection techniques to predict nutritional values. Initially using offline images, a multi-label classifier was trained. During execution, image is collected from a smartphone camera, to which the trained model is applied to detect the foods present. Both size, and labels of the objects are predicted based on a CNN classifier. From this data, the corresponding nutritional facts are fetched from a lookup store, and is displayed to the used.

In this paper [5] S. Christodoulidis used a system for the recognition of already segmented food is proposed using deep CNN approach. For the classification of a food item,a set of non overlapping patches is extracted and each of those patches is classified using a CNN and the class with the majority of the votes wins.The dataset size is first increased using 16 different transformations.The CNN network has 4 convolutional layers and the activation function used is ReLU. Each convolutional layer is followed by 3*3 pooling regions with stride 2.The last two layers are fully connected with 128 and 7 units.Softmax function is used at the last layer for normalization. For the recognition a voting scheme is used with the most classified class is assigned the food item.

In this paper [6] Kagaya first builds a dataset of ordinary images by scanning through two months of recorded data from Food-Log(FL), a publicly available app. After that the images are scaled to 80*80 and then randomly crops it down to 64*64 using cuda-convnets python module. The dataset is divided into 6 parts: 4 for training, 1 for Validation and 1 for Testing.The results were compared using SVM-based recognition. The CNN improved accuracy by over 10%. Here various hyperparameters are used to compare results and 5*5 kernel performed the best while hyperparameters showed varying results. In the second part food detection is done which classifies inputs as food or non-food. A new dataset was used here wahich was collected from social media. The input images were scaled down to 80*80 and cropped to 64*64 pixels randomly by the cuda-convnet python module.The dataset was divided into 10 groups: 8 was used for training, one for validation and one for testing. There were two hidden layers with 32 nodes and size of kernel used was 9*9, 7*7. The experiments showed an accuracy of 93.8% which is a significant improvement over the baseline method.

4 Approaches

We have tried several approaches towards solving this simple Problem of Food Image Recognition starting from Exploring Available Datasets, Finding new Datasets, Creating a Newer Dataset by taking elements from both, to updating the architecture and changing the hyperparameters of the Model to improve/fail to improve the accuracy of the Model.

4.1 Dataset Creation

An extensive image dataset is critical for our food recognition system because it helps in the learning of more general features and therefore prevent overfitting. Due to such requirement, food database with public access were always our first choice for training the model.

We have gathered a visual dataset with nearly 7700 food images, organized into 16 classes. Majority of the images have been taken from the dataset used in [1]. The images were of uniform dimension 128 * 128, with 3 channels(RGB). Also, we have used some classes from the dataset "UEC FOOD 256" [2]. Created in 2014, it contains 256-kind food photos, summing up to 31,397 instances. Most of the food categories in this dataset are popular foods in Japan. This dataset was built to implement a practical food recognition system intended to be used in Japan. Since this dataset contained images of varying dimensions, we scaled all of them to standard 128 * 128 used by our model.

4.2 Splitting the Training and Test Datasets

We have used 80% of the Dataset as the **Training Dataset** and 20% as the **Test Dataset** for the Food Image Recognition Task.

4.3 Dataset Augmentation

We used a Keras Library Function called ImageDataGenerator which can randomly change the orientation of the image by **Translation**, **Rotation**, **Sheer**, **Height Shift**, **Width Shift**, **Change Brightness** . More References can be found on [7].

This step was important because the Dataset we created by Merging certain Items from two different Datasets, did not contain enough images for good training. Each time before starting a new iteration over the Dataset, we randomly used some of these operations on the images so as to increase the number of Images that are seen by the Network.

4.4 Experimentation

We Tried to keep things simple in the Project. We modelled a network which does single label classification of Food Images using a sequence of Convolution and Max Pooling Layers and finally two fully connected Vanilla Neural Networks.

Our Experimentation included changing the below mentioned things:

➔ Increasing/Decreasing the number of Convolution Layers.

➔ Increasing/Decreasing the number of Pooling Layers.

➔ Changing the function for the learning rate α.

➔ Varying the batch size across different runs of the code.

➔ Varying the number of Layers in the Vanilla Neural Network.

➔ We Also tried some basic Image Processing Tasks on our Datasets, like Compression, Varying the number of Channels(Color - Black and White), Varying the resolution.

4.5 Variations of the Model Approaches

There are 3 variations of the dataset, which are Old, New and Merged-Old-New. We used each of these datasets on the below given Models and Evaluated the performance of each of these Models on these Datasets.

– **MODEL-1**

This Model consists 9x9, 7x7, 5x5, 3x3 Convolutional Layers in sequence with a 2x2 Max Pooling Layers in between each Convolutional Layer.
The Output from the above Network is then passed through a 2 Layer Fully connected Vanilla Neural Network. The First layer having 128 neurons and the 2nd Layer having 64 Neurons. Finally a softmax has been applied to classify items in the listed categories.(16 or 9) The Loss Function used was **Categorical Cross Entropy**
The Batch size tried was 64, number of epochs were 100 and the Dropping Factor(Arbitrarily dropping neural units to prevent overfitting) was 0.30. The learning rate was kept as a function of training loss such that the learning rate was exp(training loss)

– **MODEL-2**

This Model consists 7x7, 5x5, 3x3 and another 3x3 Convolutional Layers in sequence again with a 2x2 Max pooling Layers in between each Convolutional Layer.
The Output from the above Network is then passed through a 2 Layer Fully connected Vanilla Neural Network. The First layer having 128 neurons and the 2nd Layer having 64 Neurons. Finally a softmax has been applied to classify items in the listed categories.(16 or 9) The Loss Function used was **Categorical Cross Entropy**
The Batch size tried was 32, number of epochs were 100 and the Dropout rate was 0.5. The learning rate was kept as a function of training loss such that the learning rate was exp(training loss)

– **MODEL-3**

This Model consists 7x7, 5x5, 3x3 Convolutional Layers in sequence again with a 2x2 Max pooling Layers in between each Convolutional Layer.
The Output from the above Network is then passed through a 2 Layer Fully connected Vanilla Neural Network. The First layer having 128 neurons and the 2nd Layer having 64 Neurons. Finally a softmax has been applied to classify items in the listed categories.(16 or 9) The Loss Function used was **Categorical Cross Entropy**
The Batch size tried was 32, number of epochs were 100 and the Dropout rate was 0.1. We decided to play around a bit with the learning rate here. We assigned the learning rate as $exp(loss) + 0.5 * (1 + log(accuracy))$. We thought this might lead to something good because we are making the learning rate simaeltaneously on both loss and the accuracy so that the learning rate does not become too high or too low.

4.6 Final Model

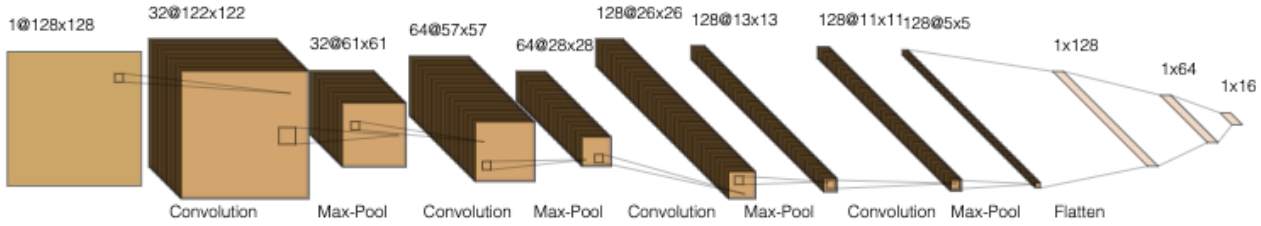

Fig. 1. The Architecture Used

After testing out various models as described above, MODEL-2 seemed to perform better than the other models. The input shape to the model is (batch_size, 128, 128). There are 4 convolution layers,

the first uses a kernel of size (7x7), the next three layers uses a kernel of size (5x5), (3x3), followed by a pooling layer of (2x2). Number of features used at each layers are 32, 64, 128, 128. This is followed by two hidden layers of 128 and 64 units. At each layer 'Relu' activation is used. The output layer consists of 16 output unit activated using 'Softmax' function. Stochastic gradient descent has been used to train the model. The learning rate is updated proportionally to the loss at the current instant [1].

$$learning\_rate = \eta_0 * e^{loss}$$

5 Experimental Results

5.1 Code Description

Description Initially the dataset was segregated using folders on the basis of their classes. The first section of the code deals with dividing the dataset into Training and Testing sets. Following this, the code to define the model has been written. A function which is dynamically called using callbacks to update the parameters is written. ImageDataGenerator is used to apply basic transformations to the dataset each time it is fetched. After each 25 epochs the model weights are saved. The keras fit_generator function is used to train the model which fetches data in dynamically to keep the RAM free. Finally the model is trained for 100 epochs. Next using matplotlib the graphs are generated. At the end predict_generator() is used to generate the confusion matrix(Manually generated for each class).

Other Details We have implemented the project using Python3. We have used openCV library for simple image processing operations, the main model has been built using Keras using Tensorflow as backend. Various plots have been generated using Matplotlib. Also for other fast matrix operations we have used Numpy library. Approximately 150 lines of codes were written. We used the dynamic learning rate implementation, proposed in [1]. The code along with the DataSet can be found here: `https://github.com/gamebird96/FoodRecognition` $_{G}ithubLink_{C}lickHere$

5.2 Experimental Platform

We have used initially Jupyter Notebook on our local machine, which was taking exceptionally long to train. For the final experimentation we have used Google Colab Environment to train and test our model. On local machine approximately it took 11hrs to complete 100 epochs, whereas in Google Colab using GPU runtime it took about 2.5 hrs to train.

5.3 Results

Model-1 Results Number of Epochs - 100
Ran the Dataset of 16 categories compiled from [1] [2] with the network Architecture mentioned in Section **4.5**.
Training Accuracy = 78.47%
Test Accuracy = 75.47%

Model-2 Results Number of Epochs - 100 Ran the Dataset of 16 categories compiled from [1] [2] with the network Architecture mentioned in Section **4.5**.
Training Accuracy = 81.41%
Test Accuracy = 78.30%

Model-3 Results Number of Epochs - 100 Ran the Dataset of 16 categories compiled from [1] [2] with the network Architecture mentioned in Section **4.5**.
Training Accuracy = 77.14%
Test Accuracy = 74.08%

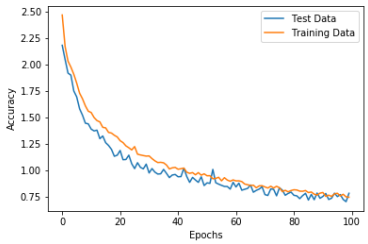

Fig. 2. Model-1 Loss

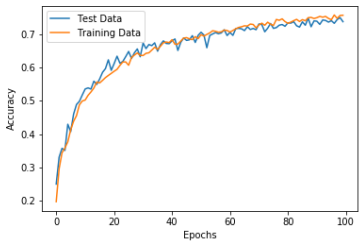

Fig. 3. Model-1 Accuracy

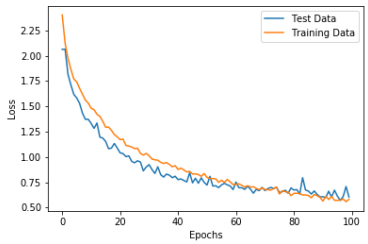

Fig. 4. Model-2 Loss

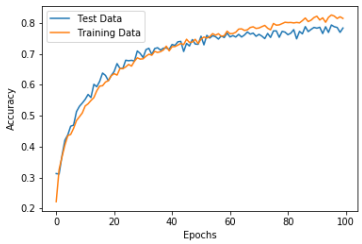

Fig. 5. Model-2 Accuracy

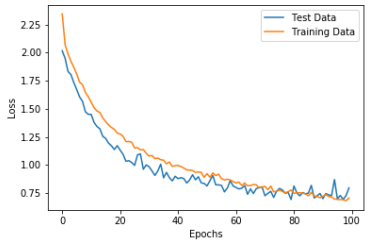

Fig. 6. Model-3 Loss

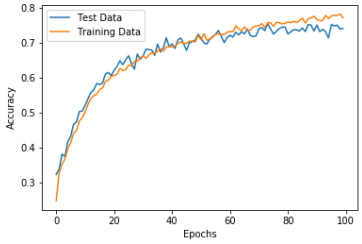

Fig. 7. Model-3 Accuracy

5.4 Evaluation of Model-2 Through 3 Datasets

Dataset-1 [1] Number of Epochs - 100
Training Accuracy = 87.28%
Test Accuracy = 89.60%
Ran the Dataset compiled from [2] with 9 Image classes with our Model.

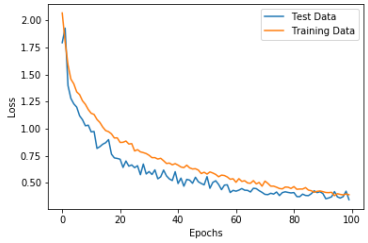

Fig. 8. Dataset-1 Loss

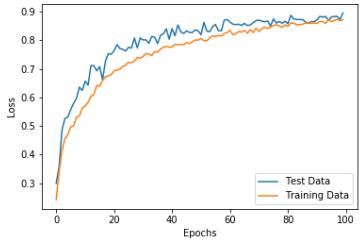

Fig. 9. Dataset-1 Accuracy

Dataset-2 [2] Number of Epochs - 200
Training Accuracy = 94.15%
Test Accuracy = 90.04%
Ran the Dataset compiled from [2] with 9 Image classes with our Model.

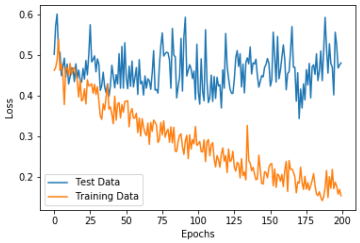

Fig. 10. Dataset-2 Loss

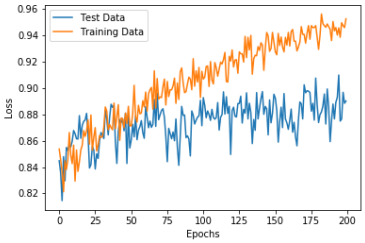

Fig. 11. Dataset-2 Accuracy

Dataset-1 + Dataset-2 Number of epochs - 200
Training Accuracy = 88.46%
Test Accuracy = 80.91% Ran the Dataset formed by combining *Dataset-1* and *Dataset-2*. It has 16 classes, since some classes were coinciding.

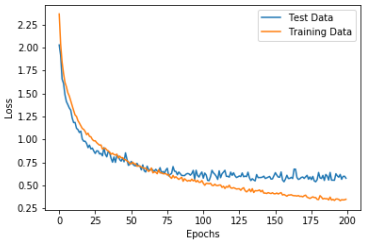

Fig. 12. Dataset-1 + Dataset-2 Loss

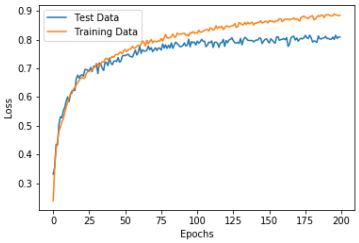

Fig. 13. Dataset-1 + Dataset-2 Accuracy

5.5 Predictions of the Model on Our Test Dataset

The Following Confusion Matrix gives a glipmse of the Prediction of our Model on The Test Dataset of about 1300 images.

Table 1. Confusion matrix for test images

	Predicted Category															
Actual Category	Apple	Banana	Broccoli	Burger	Croissant	Egg	Frenchfry	FriedChicken	FriedNoodle	Hamburger	Hotdog	Muffin	Pizza	Rice	RoastedChicken	Strawberry
Apple	201	2	0	0	1	3	0	0	0	0	0	0	0	0	0	3
Banana	2	46	0	0	0	8	3	0	0	0	1	0	2	0	0	0
Broccoli	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0	0
Burger	0	0	1	76	2	1	0	2	3	6	7	0	6	0	0	0
Croissant	0	0	0	0	15	1	3	3	0	0	0	2	0	0	0	0
Egg	1	0	1	1	1	113	0	0	0	1	1	1	1	5	0	0
Frenchfry	0	0	0	0	1	0	52	0	2	0	1	0	4	0	0	0
Fried Chicken	0	0	0	0	4	6	2	27	6	0	2	6	5	0	2	0
Fried Noodles	0	0	0	0	0	0	0	6	49	0	0	0	2	3	0	0
Hamburger	0	0	0	20	0	1	0	0	2	32	4	0	0	0	0	1
Hotdog	0	0	1	5	3	9	3	2	1	0	98	0	8	122	0	0
Muffin	1	0	0	0	1	1	2	8	2	1	1	4	2	0	0	0
Pizza	0	0	0	1	0	2	1	0	5	0	5	0	251	2	0	0
Rice	1	0	1	0	0	2	0	0	2	0	1	0	4	184	0	0
Roasted Chicken	0	0	0	0	4	0	0	7	0	0	2	1	6	0	2	0
Strawberry	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87

6 Efforts

About 20% of the total time was dedicated towards literature survey where we have gone through various papers and projects related to similar topics as mentioned under *Related Literature*. About 15% of the time was devoted to arranging and processing the dataset. About 50% of time was devoted to experimenting with the model and changing the hyper-parameter so as to obtain the best possible results. The remaining time was allocated to fine tuning, data gathering and report writing.

It was quite challenging while tuning the hyper-parameters of the models as it was taking a large amount of time to modify and test the changed model every time.

The tasks were equally distributed amongst the team members. Model-1 was developed by 193050043, Model 2 was implemented by 193050026, whereas dataset processing and Model 3 designing part was done by 193050002. The results and data were gathered and organised and by 193050034. The report and the corresponding presentation was contributed by every members of the team equally.

7 Conclusion

The Project turned out to be quite simple, yet informative due to the few experiments we carried out.
It turns out that our Model could recognize some Image classes extremely well such as *Apple*, *Banana*, *Egg*, *Strawberry*, *Pizza*, *FrenchFry* and *Rice*. We attribute this Excellent performance to one of the two factors i.e Large number of Training Examples e.g *Apples* and *Pizza* or having a special kind of shape or pattern(A Speciality) e.g *Banana* and *Strawberry*. So, Our Training Model could do well in case a reasonable high number of training examples are there or the Food Image has some sort of special shape.
Also, we intentionally included similar looking Food Items e.g *Burger Hamburger* and *hotdog* to find out whether the model is able to segregate well between these two Image classes. It Turned out that Burger was quite well recognized by the model but almost 40% of the hamburger images were classified as Burgers. Similarly, the model seemed to have confused between Fried Chicken and Roasted Chicken. Classifying almost 50% of the Roasted Chicken images as Fried Chicken.
Most of the Images of Rice were white in color, we attribute the success of the model to the significant white color of rice.
So, in conclusion the model described in the Results perform well if number of training Examples are reasonable in number(Obvious), but it also turns out to perform well when the Food Image has a special kind of appearance(such as Most of the IMage is white in case of rice, The special curved shape of a banana etc.).It was also observed that Fruits were better classified than Food items of other genre, so in future it can be extended for automated pricing and calorie prediction for Fruits in supermarket(Maybe).
We had mentioned in Section **5.3** that we use a special kind of function to update the learning rate in case of Model-3, it turned out that it did'nt really outperform an existing learning rate function.

References

1. Y. Lu, "Food image recognition by using convolutional neural networks (cnns)," 2016.
2. "Uec256." `http://foodcam.mobi/dataset256.html`.
3. E. Aguilar, M. Bolaños, and P. Radeva, "Exploring food detection using cnns," in *Computer Aided Systems Theory – EUROCAST 2017* (R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, eds.), (Cham), pp. 339–347, Springer International Publishing, 2018.
4. A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy, "Im2calories: Towards an automated mobile vision food diary," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
5. S. Christodoulidis, M. Anthimopoulos, and S. Mougiakakou, "Food recognition for dietary assessment using deep convolutional neural networks," in *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops* (V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, eds.), (Cham), pp. 458–465, Springer International Publishing, 2015.
6. H. Kagaya, K. Aizawa, and M. Ogawa, "Food detection and recognition using convolutional neural network," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, (New York, NY, USA), pp. 1085–1088, ACM, 2014.
7. F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.