[[1]](#footnote-0)

**Food Dishes Prediction**

Laxman Kumar, Rima Shah

***Abstract*— Nowadays people are getting conscious about healthy living, nutritious food and diet. That is why food dish prediction is getting popular all over the world to monitor dietary data of people.** **In this paper, we have executed an approach to recognize a food dish from its image taken using transfer learning and Convolutional Neural Networks (CNN). Using the Food 101 image dataset to build the food recognition system, we performed transfer learning on a pre-trained Densenet121 model and then implemented the CNN model using Keras. The architecture of our model was designed in such a way to shelter the Food 101 dataset which is of 5GB. The model was built using various CNN layers, max pooling layers and drop out layers to obtain the best results. The technique of max pooling was used to essentially extract the main image features for training our model. This model worked well on randomly selecting 3, 5 and 7 classes out of the total 101 classes, giving an average test accuracy of 95.3, 79% and 75% respectively. From the results of our experiment, we are able to conclude that our system can predict the food dishes accurately with the current amount of data. In future we aim to run this model for almost 32 classes of data to obtain a more accurate result.**

# INTRODUCTION

People today are leading lazy lifestyles, unhealthy diets and low exercise. This is making them obese and unproductive. The obesity rate of the world is increasing by a great rate. Looking at this awareness is being raised among people to improve their lifestyle and diet. It is important to eat well to avoid health issues. Since awareness is being spread about healthy diet, people are becoming conscious about eating healthy.

The most convenient way to keep track of your daily diet is using smartphones. Using diet diaries or making notes is time consuming and takes effort. Instead we can use our smartphones to log the same data about food by taking images of food. There are many applications in which the user has to enter the food name after taking a picture of it. We attempt to solve this issue by detecting the food dish from the image.

Great efforts are being made to food detection and recognition, although an accurate solution to this problem has not been found for food classification or detection. This is because there is a wide variety of food present in every dish and it is challenging to correctly identify each food item dish since many of them look similar in terms of color, texture, size, etc. Many research papers have described this task as impractical to resolve. Due to this it is important to accurately identify the food to understand its dietary value which can then be put to good utilization for human benefit. This will also provide us with an insight to understand the food we eat, give us the additional information about our daily food consumption, the total calorie intake and other details.

In this paper, we have attempted to create a new structure of food recognition systems using Convolutional Neural Networks (CNN) and deep learning models. The image taken will then be processed by our model to first check if this image contains food or no. After this the image is passed through our dataset to check which food category it belongs to by performing image recognition of the food dish passed to our model.

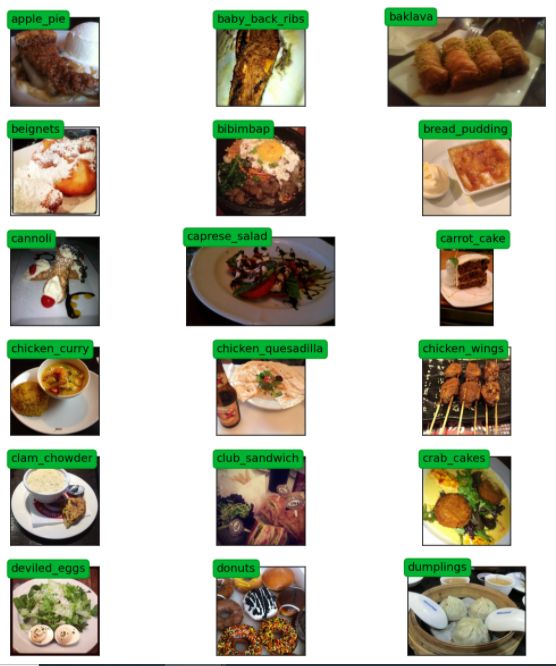
We have used the Food 101 image dataset for training our CNN model.

# PROBLEM STATEMENT & DATA

The goal of this project is to build a system that accepts an image of a food dish as input and can generate the possible name of the dish and if possible, its ingredients.

As the problem statement states, the goal is to predict the name of the dish based on the image. Currently, we are only focusing on 101 different food classes due to computation limitations.

We are using the Food-101 dataset for our food recognition system. As the name suggests it comprises 101 categories of food with a total of 101,000 images and around 5.4GB data. The original Food 101 dataset is very large and requires great computing power for performing any process in it as it constitutes images. That is why we are working on a sample dataset of 5.4GB which is identical to that of the large Food 101 dataset. These categories are namely apple pie, Baby back ribs, bread pudding, Caesar salad, carrot cake, crab cakes, hummus, pad Thai, etc. contain 1000 images each. Every class has 750 train images and 250 test images. The data files were initially all in text files. The next step was to prepare the test and train data for which we created two new folders with train images and test images. Our code reads the details of train images from the train text file and copies it into the train folder. This process is repeated for test data also. We now have two separate folders inside which there is a folder for each class of food for train and test images respectively. After the creation of these folders we then move forward with performing various data visualisation processes on it. To get an idea of what the images in our dataset look like and how its labelling is done we plot a graph of a few images selected randomly from each class and display them. Fig.1. displays random images selected from each class.

Fig. 1 Overview of Images

These images impurities like noise, distortion, intense color and some are also given wrong labels. Further, we have done rescaling to the images to have a size of 224x224 pixels. The image resizing is done so that all images have the same size for further processing. For efficient training purpose data augmentation has been performed on the images by implementing resizing, adding hue and saturation, flipping the image horizontally and rotating the images. Fig. 2. Displays a clip of the images after performing data augmentation on it. To obtain better results our model is trained with all types of noisy images so that it can detect the food image accurately.



Fig. 2. Images after performing data augmentation

# APPROACH

As we move forward to the algorithmic approach which we had taken to solve this complex problem. We will discuss the steps taken to prepare the data for modeling. All the data was separated into a train and test folder using python. We have used transfer learning and Convolutional Neural Network as the neural network.

We first started with transfer learning using pre trained Densenet121. We used the transfer learning approach at the beginning since building a model which can work for 101 classes of image classification is beyond the scope of the project. Not only building a model from scratch which can serve 101 classes with good accuracy requires complex models but also heavy computation resources. Also training a model with 101 classes required a huge dataset and 5GB dataset will not be sufficient. Since our dataset is a sample taken from Recipe1m+ dataset which is around 95GB, using transfer learning was the only step we could think of. We even tried with Alexnet and Resnet but none was able to beat the result of Densenet. We first ran all three models with 20 epoch and then compared the result obtained. After observing the results, we decided to run the Densenet121 for 100 epoch.

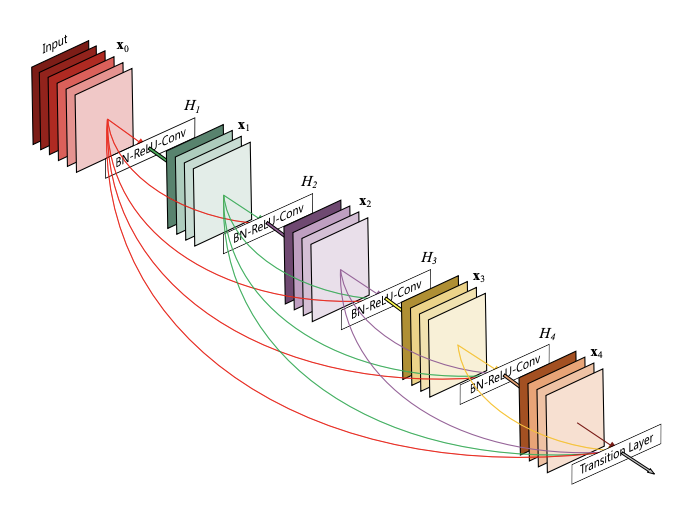


Fig 3 - Densenet Architecture

After transfer learning, we moved forward to CNN. Initially we tried various combinations using Keras and PyTorch both. Later for simplicity we used Keras for further modeling. We built an architecture with 9 CNN layers, 3 MaxPooling layers, 4 dropout layers and a dense layer. We were able to build this architecture after several trials. This architecture works well for 3 classes, 5 classes and 7 classes. Detailed approach to how we trained our model is explained in the next paragraph.

We first approached with a neural net for 3-classes out of 101 classes. We built a function which randomly selects 3 classes from the 101 classes and uses that class for modeling. We ran the experiment several times for 50 epoch to check the accuracy for random clases. We then changed the architecture of the model and repeated the same experiment but were not able to improve the average accuracy. After 3 classes, we repeated the same experiment for 5 classes and 7 classes also and got good results. For 5 class neural networks, we ran the same model which we have used for 3 classes and also designed some new model but the first model performed better than any other new model created and therefore we kept the same model for 5 classes and for 7 classes.

We tried for more than 7 classes but were not successful and hence we stopped our experiment at 7 classes and will try to make a model which can work well with more than 7 classes in the future.

1. RESULT

Transfer learning works well in case of image processing and our task includes processing the food images to detect the food name. We tried different transfer learning models (ResNet, DenseNet and AlexNet) but DenseNet121 works better than the other two in our case with train accuracy 68.12 and validation accuracy 64.10. We tried transfer learning since we have 101 classes and getting good accuracy with a model which is built from scratch with the computation resources which we currently have is extremely difficult.

After transfer learning, we designed our CNN model with 9 CNN layers and a few MaxPooling and Dropout layers. We first just made a try run by running this model on all the classes but we already knew it wont work well and we got train accuracy of 36% which is good if we compare the model complexity and the number of classes we have but not that good in terms of final semester project. Later we made a function which can randomly select n number of classes and run the model on them. We tried running it in 3 classes first. We ran 3 class experiments 4 times to check the accuracy on randomly selected classes and we were able to achieve the highest train accuracy of 93.18% and highest test accuracy of 95.3%. We then repeated the same experiment for 5 classes and was able to achieve the highest train accuracy of 83.16 and highest test accuracy of 79.%. Finally, we ran the same experiment on 7 classes and were able to achieve the best train accuracy of 76.30% and best test accuracy of 75%. All the results are mentioned in table 1. We then tried running the same experiment for a higher number of classes but didn’t work well.

Each experiment took more than 15 hours to run on the computation resources which we currently possess and trying out different combinations of models was again a difficult part. We tried designing a complex model for a higher number of classes but testing out and tuning the model did not work well on our system and therefore we ended our experiments on the current state.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Train\_Acc** | **Val\_Acc** | **Test\_Acc** |
| DenseNet121 | 68.12 | 64.10 | - |
| 3Class – v1 | 86.27 | 81.07 | 75 |
| 3Class – v2 | 84.76 | 78.40 | 85.9 |
| 3Class – v3 | 82.18 | 74.53 | 76.6 |
| 3Class – v4 | 93.87 | 88.80 | 95.3 |
| 5Class – v1 | 82.12 | 74.74 | 79.7 |
| 5Class – v2 | 82.90 | 78.26 | 67.2 |
| 5Class – v3 | 83.16 | 74.61 | 71.9 |
| 7Class – v1 | 59.68 | 48.44 | 42.2 |
| 7Class – v2 | 76.30 | 73.96 | 75.0 |
| 7Class – v3 | 70.05 | 58.20 | 65.6 |

Table 1: Model Result

# DISCUSSION

We built a model which worked well for any 7 randomly chosen classes but number of classes greater than 7, the model didn’t work well. The reason we stopped the experiment at 7 classes is due to computation limitation. Tuning the model with 100,000 images on a personal computer was very challenging. Each dry run was taking more than 10 hours to run on 50 epoch and therefore running all the models fully was not easy. Currently we have a model with 9 CNN and few other layers. A model with this complexity was taking hours to run on 50 epoch. We built a model with 15 CNN layers for a number of classes greater than 7 but the model was taking days to finish on 11 classes and therefore we stopped the experiment. Also, running the transfer learning took days to finish. Therefore, all the limitations of this project will mostly be covered in the future.

In future, we would like to work on building a complex model which can give good accuracy till 32 randomly chosen classes and also will run the model for at least 200 epoch. Running for 200 epochs will better fit the model and fine tune all the weights of the network and at the same time will increase accuracy. Also, we will be playing with some more transfer learning applications and will run the model for 500 epoch on our dataset to compare the accuracy.

1. REFERENCES

1. . [↑](#footnote-ref-0)