Automatic Recognition of Pictured Dishes in Food-101 Project Update - 2

Abishek Kumar
Department of Computer Sciences
University of Wisconsin-Madison
akumar225@wisc.edu

1 Introduction

Classification is an important tool in today's world where big data is used to make all kinds of decisions in economics, medicine, and more. For example, detecting spam in emails can be identified as a classification problem with two classes as spam and not spam. The problem of object detection and classification is a widely researched topic in machine learning due to its broad range of applications. There is an array of different algorithms that have been developed over the years to solve this problem. Food classification in particular has many applications like helping patients track their calorie intake, auto-organizing of pictures in mobile phones, etc. But, unlike other image recognition tasks wherein we have definitive features separating each class, the problem of food classification is unique due to its high intra-class variability. Lighting, orientation, and the very realization of a recipe are some of the factors which can contribute to this variability.

In this project, we will try to understand the performance of simple classification algorithms like Linear Regression, KNN for the food classification task and compare their performance against more sophisticated algorithms like Neural Networks. We will use the Food-101 dataset for training our classifiers and evaluate their performance using holdout method, precision/recall, and ROC curve. Finally, we will propose a recommended algorithm based on our results.

2 Dataset: Food - 101

The dataset consists of 101 food categories with 101,000 images, 1000 images for each category split into 750 training images and 250 test images. Both the training and test images have been downloaded from foodspotting.com. The dataset has the top 101 food categories from the website. The test images have been manually cleaned, while the training images were left intentionally unclean and have some amount of noise. This noise comes mostly in the form of intense colors and sometimes mislabelled samples. All images have been re-scaled to have a maximum side length of 512 pixels and smaller images have been discarded.

2.1 Preprocessing

As a pre-processing step, we first separated the training and testing images into two different folders. Then, we resized each image in the dataset to the size of 128 x 128 x 3. We chose this size mainly due to memory constraints. Initially, we tried to experiment with the size of 299 x 299 x 3. But the number of pixels in this case was huge - 268,203. This meant that the model had to learn 268,203 weights for fitting the data. This led to memory issues even on a machine with 500GB of RAM. Hence, we decided to resize the images to 128 x 128 x 3 which gave a fair trade-off between image quality and feasibility of computation. We used python's cv2 library for resizing the images.

3 Experimental Setup

The experiments were done on ml.p3.16xlarge machine on Amazon AWS Sagemaker. The machine has the following configuration - 64 cores, 500GB of RAM and 250GB of hard disk space. We started with the following machines at first - 16GB RAM, 4 cores and 31GB RAM, 8 cores. But, both these machines were not able to handle the huge size of the dataset and hence we decided to go for a bigger machine. The entire implementation was done in Python and we used Python's scikit-learn library for training and evaluating the models.

4 Implementation

In the second phase of the project, we decided to implement KNN Algorithm for classification. For this, we used KNeighborsClassifier from scikit-learn library. Since the training was taking a lot of time (>17 hrs) on the entire dataset, we decided to train and test the classifier only on a subset of data. We randomly sampled 7500 images from the training data and used it for training. We ensured that the training data had samples from all classes and tested it using a set of 2500 randomly sampled test images.

The entire implementation of the project can be found in this Github Repository - Automatic-Recognition-of-Pictured-Dishes-in-Food-101.

5 Results

We trained 7 different classifiers with different settings for K values. Figure 1 shows the plot of K values vs accuracy on the test set. From the figure, we can see that the best performance is achieved with K value as 38.

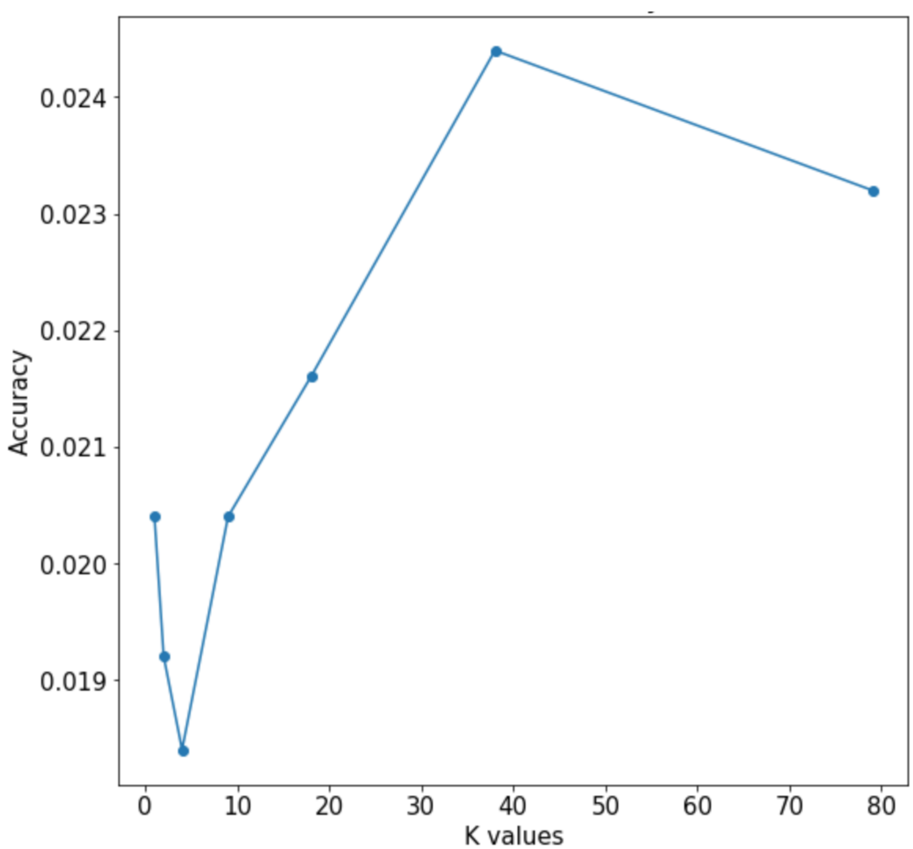

Figure 1: K-values Vs Accuracy

6 Conclusion

The accuracy of all the KNN Classifiers was very poor (<3%) as we can see from the figure. This could be due to the fact that the classifier considers each pixel as an individual feature and ignores the spatial correlation between pixels. We are also evaluating accuracy score based on the top-1 prediction. This is a very harsh metric as it expects for each sample that the correct label is predicted. Instead, we can also get the score for the class being in top-3 / top-5 predictions and evaluate accordingly.

7 Next Steps

- December 7^{th}: Evaluate the top-3 and top-5 prediction accuracy for the RidgeClassifiers and the KNN Classifiers. Develop a neural network for classification and produce results.
- December 14^{th}: Compare the results between all three algorithms and prepare final report.