Computer Vision and Food Classification

Springboard Datascience Intensive Capstone Report

Simon Bedford

September 2016

Contents

1 Introduction

Cooking, and more broadly food in general, is a very popular content category online. Indeed, according to the FT.com [1], cooking has become a key category to focus on for driving growth in both audiences and advertising alike.

When someone is looking for a specific recipe, or even just inspiration, it is now perfectly natural for them to go online and consult one of the myriad recipe curating and aggregating websites.

From big names such as Allrecipes.com, Food.com and the Food Network, to more focused sites like Epicurious and Serious Eats, there is an increasing proliferation of online properties where consumers can find, view and share recipes and cooking ideas. Nowadays even companies like YouTube and Buzzfeed are trying to drive traffic through cooking videos and websites [1].

One only has to look at some of the numbers to see the size of the online demand for food content:

- In December 2016, Allrecipes.com drew 50M unique visitors, a 23% increase from the same month the previous year [1]
- Epicurious boasts 30,000 professionally created and tested recipes, along with 150,000 member-submitted recipes [2]
- The New York Times cooking website receives 7-8M monthly visitors, has 650,000 subscribers and stores more than 17,000 recipes online [1]

Perhaps even more impressive is the related growth in photo and video content:

- There are 168M+ posts on Instagram with the hashtag #food, and 76M+ for #foodporn [3]

- Food related videos were viewed 23bn times in 2015, a 170% increase over 2014 [1]

Furthermore, and in-line with broader social trends, a large and growing proportion of this content is produced and consumed on mobile devices. Allrecipes.com for instance states that 66% of their page views are now via mobile. [4].

Given the growing interest in food and recipes, and the increasing dominance of photo and video media, there is a clear opportunity for computer vision techniques to be applied to the world of food content.

For example, image recognition and classification techniques could be integrated into a seamless mobile experience to enable faster and more accurate recipe search and suggestions, and even enable content producers to display more relevant and targeted advertising specific to the types of food of interest to different users.

However applications are not just limited to search and advertising, but can also have health benefits. For example food classification algorithms could help people keep daily food diaries and running calorie counts, an idea that has already been explored by both Microsoft [5] and Google research teams [6].

The aim of this project is as proof-of-concept, to explore whether it is possible to create a food classification algorithm that could feasibly be implemented into a production environment, enabling automated food recognition and recipe retrieval. With this goal in mind, it is key to consider the right measures of performance:

1. Classification Accuracy
2. Resources and Cost
3. Ease of Production Deployment

Classification Accuracy

In order to measure performance, we could use a simple metric like:

$$Simple\ Accuracy = \frac{\#\,Correct\ Predictions}{All\ Predictions}$$

However this is a slightly naive measure, and instead we will use the F1-score:

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

To better understand the F1-score, consider a very simple binary classifier aiming to distinguish between Pizza and Not-pizza. There are four possible prediction outcomes which we can summarize in the Confusion Matrix below:

We can then define:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad \text{and} \quad Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

One way to think about Precision is that it measures, out of all images predicted to be Pizza, what proportion are Pizza in reality. In a sense this tells us how noisy our Pizza predictions are, or how much does the classifier confuse other types of food with Pizza.

		True Categories	
		Pizza	***Not Pizza***
Classifier Predictions	***Pizza***	True Positive (TP)	False Positive (FP)
	Not Pizza	False Negative (FN)	True Negative (TN)

Table 1: Confusion Matrix Truth Table

Similarly Recall asks, out of all true Pizza images, what proportion were correctly identified as being Pizza. This is a measure of how good the classifier is at identifying pizza images and placing them in the correct category.

The F1-score is the harmonic mean and is a measure of the balance between Precision and Recall. It is measured on a scale from 0 to 1, with 1 being the best possible score.

Resources and Cost

In terms of Resources, what we care about is how costly it is in terms of both time and money to create an accurate classifier. From a practical perspective, any company evaluating using this type of technology would need to perform some type of ROI analysis, including taking into account the costs of training an algorithm with sufficient accuracy.

During our analysis we will measure the time taken in training different algorithms, and also the cost of any external computing resources required. For the sake of this project we will ignore both man-hours and also negligible costs such as electricity etc.

Ease of Production Deployment

It is also important to consider the practicalities of implementing a successful model in a production environment in a way that would be beneficial to the customer or user experience. Some possible questions to consider are:

- Can the algorithm perform accurate classification in real time?
- Is it possible to run the model locally on a mobile device, or does it need to be run on a central server?

We will not spend a lot of time analyzing these questions in detail, however in the conclusions and recommendations we will briefly discuss the feasibility of using the best model in production.

Classification Models

Finally, with regards to models, we take three approaches, each of which will be explored in more detail in later sections:

- Traditional Machine Learning
- Deep Learning with Neural Networks
- A hybrid approach fusing Deep-Learning with traditional Machine Learning

As will be seen, deep-learning techniques enable us to achieve impressive results in a relatively short amount of time, with a good possibility of being applicable to different problems and contexts.

2 Exploratory Data Analysis

The dataset used is The Food-101 Data Set from the ETH Zurich Computer Vision Laboratory [7]. This dataset contains 101 food categories, with 1,000 pictures per category for a total of 101,000 images. Each image has a maximum side-length of 512 pixels, and the total dataset comprises approximately 5GB of data.

Early on in the project it was decided that, given the available computing resources, the original dataset was too large for practical experimentation.

In order to create a smaller dataset the existing categories were compared to the 100 most-popular recipe-related Google search terms in 2015 in the US. In total, there was an exact match with 13 classes in the dataset, and for the remainder of the project, the analysis is focused on the top 12 matching categories:

Pork Chop	Lasagna	French Toast
Guacamole	Apple Pie	Cheesecake
Hamburger	Fried Rice	Carrot Cake
Chocolate Cake	Steak	Pizza

Noisy Data

The images in the Food-101 dataset are of mixed quality. Some are very clear, well-lit and framed correctly on the food item in question. Others however are of worse quality, out of focus, poorly lit, containing other irrelevant items and, in some cases, mislabeled. Some examples of both high and low-quality pictures are shown in figure 1.

Image Sizes

Analysis of the distribution of image shapes within the dataset show that 60% of images are 512 x 512, and so the whole dataset was standardised to this shape for all classification attempts.

RGB Histograms

One of the first exploratory steps was to compare RGB histograms for the images across the different food categories. As can be seen in figure 2, there are clear differences in the distribution of pixel values for Red, Green and Blue histograms between categories.

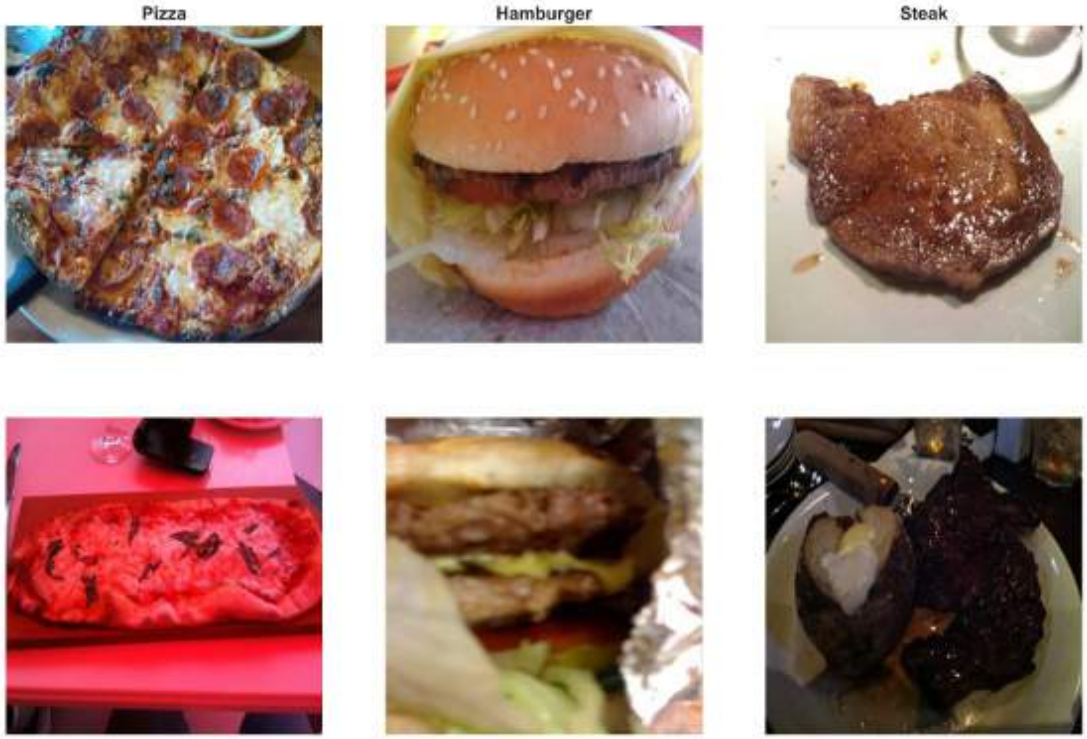
Histograms for all food categories are shown in appendix A.

Principal Component Analysis

The data was also visualized on a 2-D plot to see if any clear patterns emerged, for example images from the same class being clustered together.

For dimensionality reduction to 2-dimensions, the following approach was used:

Figure 1: Select pictures of Pizza, Hamburger and Steak categories, illustrating both high and low-quality images.



1. Initial reduction using Randomized PCA with n_components = 50
2. Further reduction to 2-D using TSNE (t-distributed Stochastic Neighbor Embedding)

Given that in total the dataset comprises 12,000 images, for practical purposes we used only the 40 nearest neighbors to the mean image for each category (480 images in total).

As can be seen from figure 3, based upon the extracted components, there are no discernible patterns or groupings in two-dimensions.

3 Traditional Machine Learning

For the purposes of this project, when we refer to Machine Learning, we mean the use of pre-selected features that are used in conjunction with computer-enabled algorithms to attempt to solve our classification problem. In particular, we will focus on using *Supervised Learning* algorithms.

One of the key points for machine learning is that we must make a conscious choice regarding the types of features we wish to use prior to training the model. In some cases the features are extracted manually or semi-manually, although in other cases we may also rely on unsupervised models to extract features.

A successful model should be able to create a general rule for correctly differentiating between food classes based upon the provided features. Thus we need to try and identify features that are

Figure 2: Histograms of RGB pixel values for mean images for the Guacamole, Chocolate Cake and Pizza categories.

similar enough for images belonging to the same category, but also different enough between classes to enable differentiation. A description of the types of features used during the project can be found in table 2.

For the very earliest attempts, some features were tested individually, but in most cases, different types of features were chained together before being used for model training.

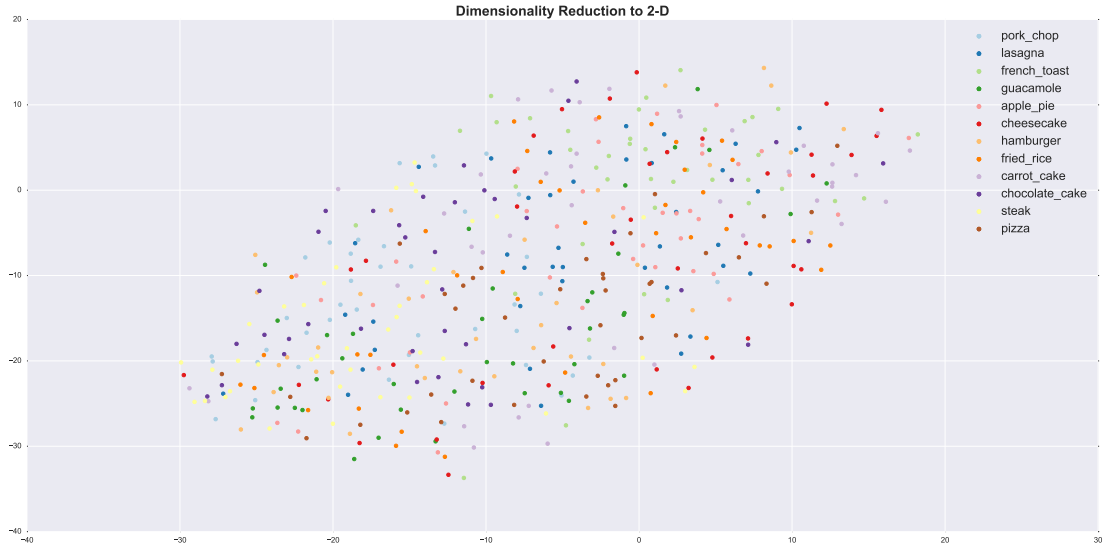
In some cases features were extracted from the image as a whole, and in other cases the images were divided into sub-sections or windows and then features extracted from each sub-section before being chained together to create one long vector of features for the whole image.

Models

As well as testing a number of combinations of different features, we also started by testing several different models on a simple set of RGB Histogram-based features, including:

- k-Nearest Neighbours
- SVM: Linear & Polynomial Kernel
- Decision Tree
- Random Forest
- ADA Boost Classifier

Figure 3: Images plotted in 2-D; First PCA was applied to reduce from 786,432 to 50 dimensions, and then TSNE further applied to reduce to 2 dimensions.



- Gaussian, Multinomial and Bernoulli Naive Bayes
- Linear & Quadratic Discriminant analysis

Based on the results of training and testing these models, see figure 4, it was decided to mainly use Random Forest classifiers, as the RF gave the best accuracy and F1 scores, and also had a very fast training and testing time.

All models, both supervised and unsupervised, were implemented using the python Scikit-learn library.

In total we tried approximately 40 traditional Machine Learning approaches using a mix of features and classifiers. The full list of results can be seen in Appendix B, and a summary of a selection of approaches is shown in table 3.

The best model was based upon the following procedure:

Feature extraction

1. Split each image into non-overlapping squares of side 32 pixels; for our 512 x 512 images this results in a grid of 256 boxes per image
2. For each box, extract the following features:
   - Average red pixel value
   - Average blue pixel value
   - Average green pixel value
   - Number of edges (using skimage canny edges algorithm)
   - Number of corners (using skimage corner-fast algorithm)

Table 2: Description of the different types of features used for machine learning attempts.

RGB Histograms	One of the simplest ways to differentiate between images is by comparing the distribution of Red, Green and Blue pixel values. Figure 2 shows that on average, some food categories have quite different distributions.
Pixel Values	In some models, individual pixel values were used as features, typically for images scaled to a smaller size.
Edges	Edge features are extracted using image processing algorithms that look for discontinuities in image brightness resulting in the appearance of boundaries or "edges" between regions. Specifically we count the number of edges within a given area. although more advanced techniques also take into account the edge orientation.
Corners	Corner features, defined as being the intersection of two edges, are also extracted algorithmically. In this case only the number of corners was considered.
Feature reduction	Sometimes it can be beneficial to use unsupervised learning algorithms to reduces the dimensionality of extracted features. Two such algorithms tested were Principal Component Analysis as well as K-means Clustering.

3. Concatenate all features together into one long feature vector of length 1,280

Classifier

The model used was a Random Forest. Parameter optimization was performed using a grid-search on the number of estimators and maximum tree depth, and then a classifier was trained using the best values. The results obtained are shown in table 4.

We see that this model is most successful at classifying the Pizza (F1 = 0.44) and Cheesecake (F1 = 0.41) classes, and performs particularly poorly on Apple Pie (0.16) and Carrot Cake (0.20).

One way to visualize the classifier performance is by looking at the confusion matrix which can show us both the categories for which the classifier performs well, as well as those classes which it finds hard to distinguish (see figure 5).

Once again it is clear that Pizza and Cheesecake are the categories with the best overall results. Furthermore, by looking at the darker shaded cells off-diagonal, we see that the model has particular difficulty distinguishing between Steak and Pork Chops, and also between the different categories of cake.

4 Deep Learning

The second approach was based upon deep learning techniques, in particular using Convolutional Neural Networks (CNNs). CNNs are currently recognised as being state-of-the-art models for image classification [8], and in fact various image classification and captioning competitions have been won consistently over the past few years using CNNs [9].

One of the biggest challenges associated with these models is that training a full CNN from scratch requires a lot of data, and takes a long time. For example, VGGNet, the winning model of ImageNet 2014, was trained on 1.3M images, for approximately 2-3 weeks [8]. Luckily all is not lost as there exist a number of techniques whereby it is possible to take a pre-trained model and adapt it to a

Figure 4: Comparison of F1 score and training time of different classifiers using RGB Histogram-based features.

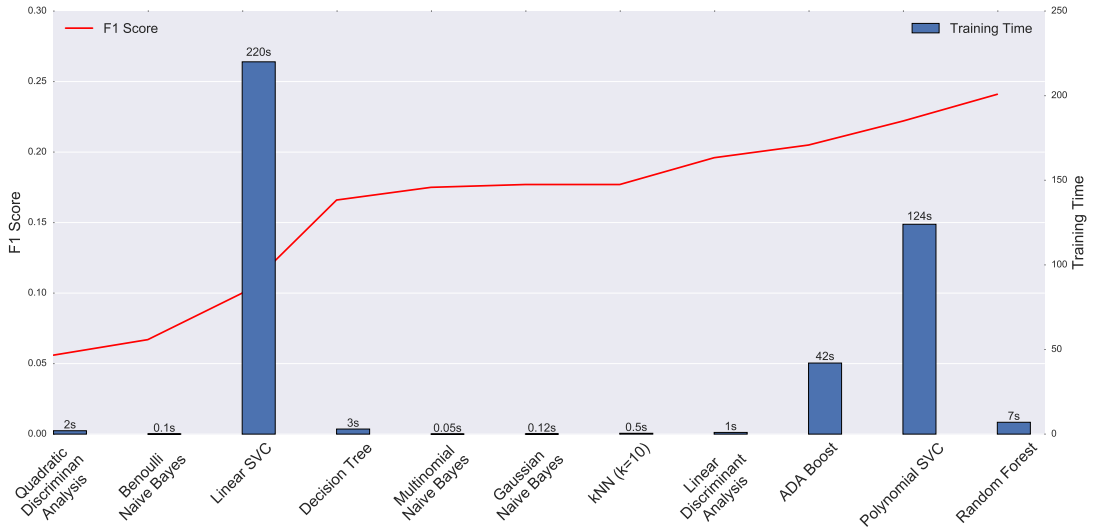
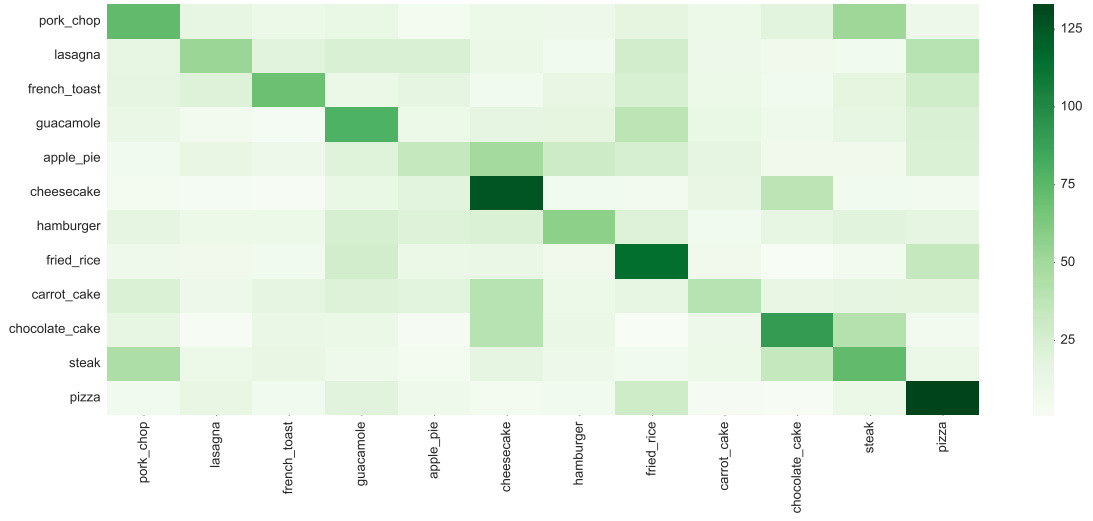


Figure 5: Confusion matrix for the best machine learning classifier. Darker shaded cells on the diagonal indicate more correct predictions for that category. Darker shaded cells off-diagonal are indicative of wrong predictions and confusion between classes.



different but related problem through a technique called Transfer Learning ([10]).

Table 3: Results of different combinations of features and supervised learning models.

Classifier	Features	F1 Score
Random Forest	100-d vector extracted using K-Means Clustering on individual pixels	0.09
Random Forest	100-d vector extracted using PCA on individual pixels	0.07
Random Forest	Images re-scaled to 32 x 32, pixels as features	0.19
Random Forest	RGB Histogram for complete image + individual pixels of 32 x 32 image	0.26
Random Forest	RGB Histogram + Edges + Coreners; zero-variance features removed; PCA to reduce to 300-d vector of features	0.27
Random Forest	Image split into 32x32 boxes; for each box extract Avg Red, Green, Blue pixel value, # Edges, # Corners	0.31
Random Forests trained on each feature type; Bayesian Net trained on probabilities from each feature classifier	Image split into 32x32 boxes; for each box extract Avg Red, Green, Blue pixel value, # Edges, # Corners	0.29
Random forest trained on segments; overall prediction based on average of segment predictions.	Image split into segments (using SLIC algorithm) and for each segment use Avg, Max, Min, Range of Color values & normed histograms for RGB	0.21

Feature Extraction

The first approach was using pre-trained CNNs for feature extraction, and then training a Linear SVC model using the obtained features.

The pre-trained models used were AlexNet [11] and VGGNet [12], and for each network, three sets of features were used based upon the outputs of Fully-Connected layers 6, 7 and 8.

The model took less than 1 hour to train, and the best result came from features extracted from Fully Connected layer 7 from VGG Net. The results obtained are shown in table 5, and we see that, in a very short space of time, we have achieved significant improvements vs. the top machine learning approach, more than doubling our overall F1 score from 0.32 to 0.69.

Fine Tuning

The second approach was to fine-tune the weights of a pre-trained network using our problem-specific dataset of food images. For this approach we worked exclusively with the AlexNet model, testing a number of strategies that combined:

- Using only the original dataset vs. using an augmented dataset
- Fine-tuning the whole network vs. fine-tuning only the fully-connected layers

Chart 6 shows the overall F1 score for each approach. We see that using augmented data gives better performance than using only the original data, and furthermore the best results come from fine-tuning the whole network vs. just the Fully-Connected layers.

The best overall classification results were obtained using the following procedure:

1. Data Pre-Processing

Table 4: Per-class results from the best machine learning classifier.

Class	Precision	Recall	F1	Support
Pork Chop	0.29	0.29	0.29	250
Lasagna	0.31	0.21	0.25	250
French Toast	0.37	0.28	0.32	250
Guacamole	0.28	0.32	0.30	250
Apple Pie	0.19	0.14	0.16	250
Cheesecake	0.35	0.50	0.41	250
Hamburger	0.30	0.23	0.26	250
Fried Rice	0.35	0.46	0.39	250
Carrot Cake	0.26	0.16	0.20	250
Chocolate Cake	0.36	0.36	0.36	250
Steak	0.26	0.29	0.28	250
Pizza	0.38	0.53	0.44	250
Overall	**0.32**	**0.31**	**0.31**	-

Table 5: Per-class results from the Linear SVC + CNN Features.

Class	Precision	Recall	F1	Support
Pork Chop	0.55	0.51	0.53	211
Lasagna	0.67	0.61	0.64	187
French Toast	0.64	0.62	0.0.63	208
Guacamole	0.90	0.89	0.90	189
Apple Pie	0.55	0.60	0.57	194
Cheesecake	0.70	0.67	0.68	206
Hamburger	0.71	0.76	0.73	200
Fried Rice	0.81	0.88	0.84	213
Carrot Cake	0.66	0.65	0.66	205
Chocolate Cake	0.70	0.76	0.73	178
Steak	0.54	0.51	0.52	211
Pizza	0.82	0.78	0.80	198
Overall	**0.68**	**0.69**	**0.69**	-

All images were first re-scaled to 256 x 256, and each class of 1,000 images was split into:

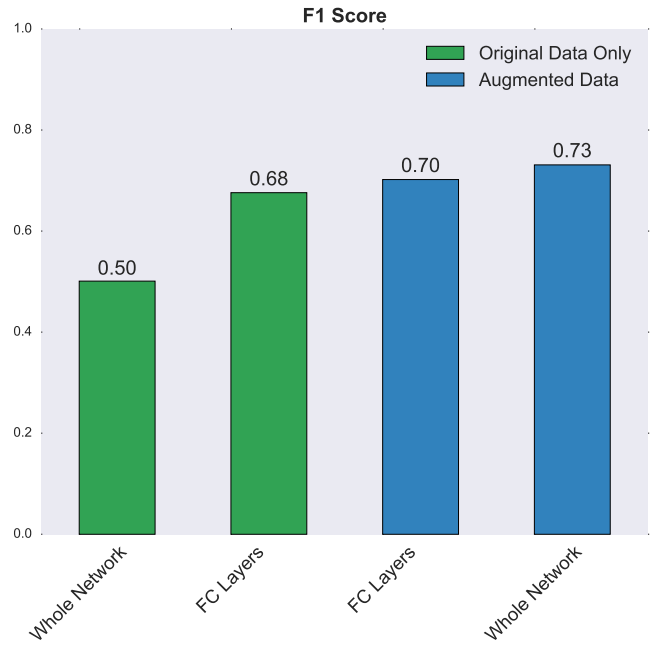
- Training - 664 images
- Validation - 136 images
- Testing - 200 images

2. Data Augmentation

Training and validation images were further augmented to generate a total of 16 images per input image:

- Original image + mirror image
- 3 Lightened Images + their mirror images
- 3 Darkened Images + their mirror images

Figure 6: F1 scores for different models created by fine-tuning AlexNet whole network vs. only FC layers, using either original data only or augmented data.



- Original image rotated by 180 degrees + mirror image

Note: The image lightening and darkening was performed using the adjust_gamma function from the skimage exposure module with fixed gamma values (Lighten = [0.45, 0.65, 0,85], Darken = [1.25, 1.50, 2.00]).

3. Training Parameters

The chosen training parameters were in general based on the default parameters provided with the pre-trained network.

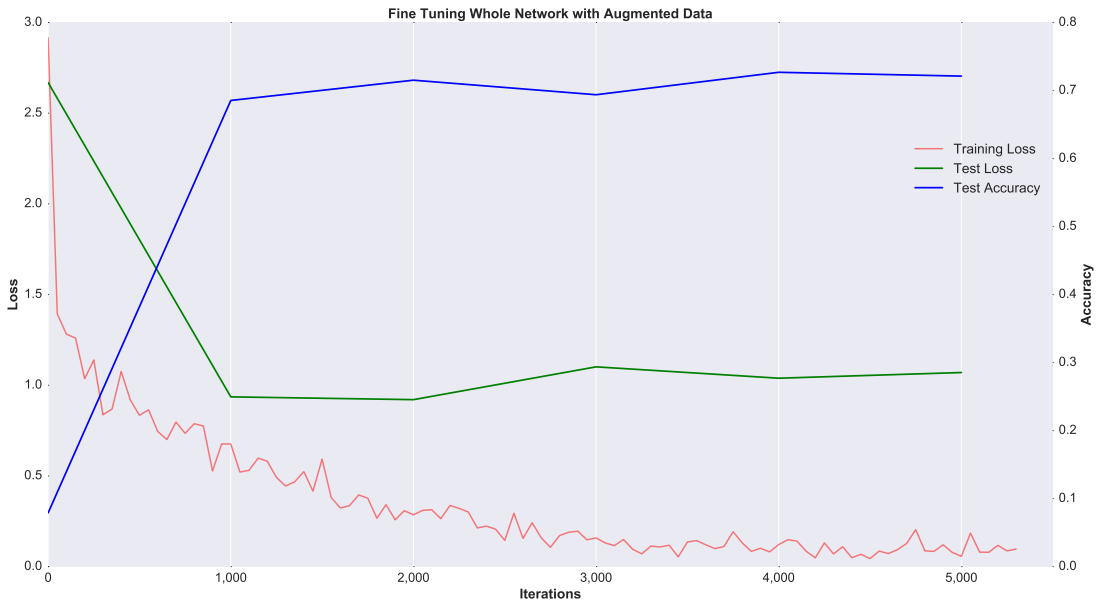
Base Learning Rate	0.001
Learning Rate Update Policy	Step, with stepsize = 3,000
Momentum	0.9
Weight Decay	0.0005
Batch Size	150
Dropout	0.5

The training time was initially set to 30,000 iterations, but training was terminated early once the network appeared to have converged (see figure 7).

At test time, the same deterministic data augmentation strategy was applied to each of the test images, and the predictions for each augmented image were averaged to give an overall prediction. This non-standard approach gave an improvement of approximately 2% on top of the results achieved via prediction on single test images. The test results are shown in table 6.

Once again we plot a heatmap of the confusion matrix for the best CNN-based model, and the

Figure 7: Training curve for fine tuning of Alexnet.



improvements in per-class results are quite clear, with the diagonal cells having far darker shading than the rest of the table.

Finally we examine the Precision-Recall curves for this classifier for each of the individual categories (figure 9). We can see that, for nearly all the categories, it would be theoretically possible to achieve near-perfect Precision or Recall through a suitable choice of threshold. However what interests us in the context of our problem is a balance between the two, with both being as close to one as possible.

From the curves it is clear that the category with the best performance is Guacamole (dark green) with a curve that gets very close to the top right-hand corner. At the same time Steak (yellow) is the category with the worst results, with the best balance between precision and recall at around 0.6.

Additional Optimization

Although we were able to make significant improvements using pre-trained CNNs, we do not believe that our results represent the best possible classifier we could train. In fact, with additional time and resources, it should be possible to continue to improve on the overall and per-class scores by:

1. Additional fine-tuning of the network hyper-parameters (e.g., dropout rate)
2. Increasing the training batch size
3. Generate additional data by expanding the augmentation techniques used
4. Fine-tuning more recent and better-performing models such as VGGNet or GoogleNet
5. Training a number of models on the data, and aggregating the predictions of each model

Through a combination of these strategies, it should be possible to achieve an F1-score of 0.80 or higher.

Table 6: Overall results from best fine tuning approach

Class	Precision	Recall	F1	Support
Pork Chop	0.70	0.54	0.61	200
Lasagna	0.73	0.76	0.74	200
French Toast	0.72	0.71	0.72	200
Guacamole	0.90	0.93	0.92	200
Apple Pie	0.61	0.74	0.67	200
Cheesecake	0.73	0.72	0.72	200
Hamburger	0.76	0.86	0.80	200
Fried Rice	0.71	0.92	0.80	200
Carrot Cake	0.74	0.74	0.74	200
Chocolate Cake	0.86	0.73	0.79	200
Steak	0.70	0.55	0.62	200
Pizza	0.89	0.82	0.86	200
Overall	**0.76**	**0.75**	**0.75**	-

Figure 8: Confusion matrix for best overall fine-tuned CNN.

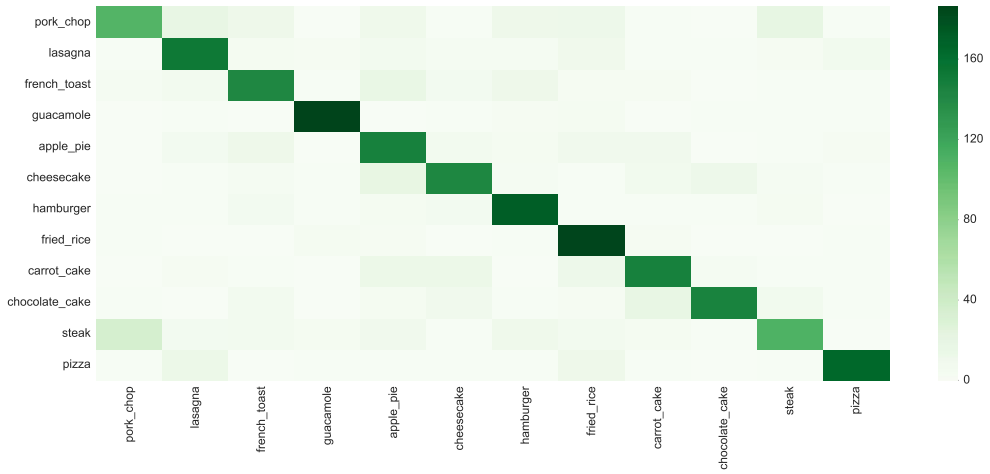
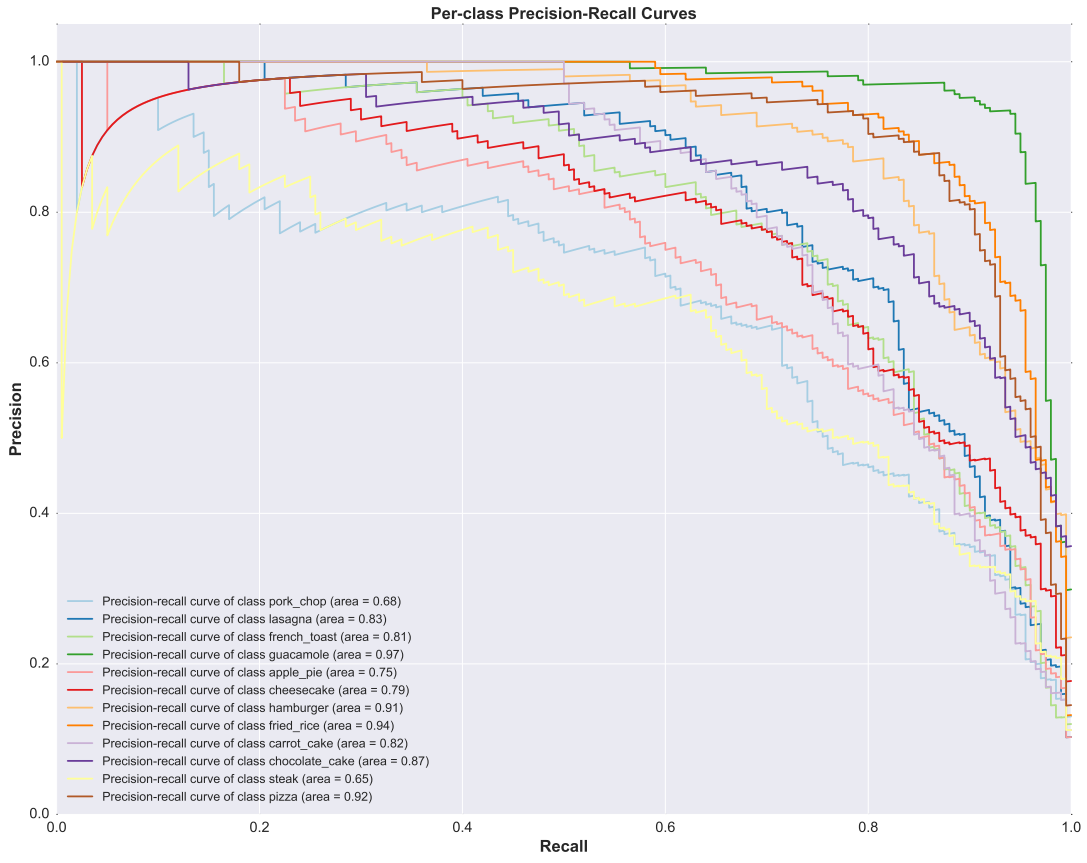


Figure 9: Precision-Recall curves for the different food classes.



5 Fail Cases

It can be illustrative to look at fail cases to better understand the weak points of our model.

In our case, even with our best model we see that Steak and Pork chop continue to present challenges:

- Steak has precision 0.70 but recall 0.55
- Similarly, Pork Chop has precision 0.70 but recall of 0.54
- Out of 200 test images:
  - 21 Steak pictures are classified as Pork Chop
  - 36 Pork Chop pictures are classified as Steak

However, when we look at example of mis-classified images, these errors become more understandable (see figure 10). Indeed, in some cases, it would be hard even for a human to correctly classify some of the images.

Figure 10: Examples of fail cases confusing Steak and Pork Chops categories.

Another area where the model still has some issues, although to a slightly lesser extent, is in distinguishing between different types of cake, for example confusing chocolate cake with cheesecake.

Should it not be possible to achieve sufficient single-class accuracy through further optimization, one approach could be to consolidate some categories into overarching super-categories, for example consolidating all types of cake together into a single Cake category, and combining Steak and Pork Chops into a Meat category. Tests using one such approach resulted in an immediate improvement for F1-score from 0.75 to 0.85.

6 Comparison of Methods

Overall, Deep Learning with Convolutional Neural Networks was clearly the better approach, resulting in far superior test outcomes. It was also in some ways, slightly surprisingly, an easier method to use, as it avoided the need to manually choose and test different combinations of features.

One drawback of CNNs is the amount of computing resources and data required to train a successful model. Some of these difficulties can be mitigated by using Transfer Learning techniques on pre-trained models, and we saw that within a short space of time we were able to significantly improve our results using these methods. However even the small amount of CNN training we carried out would have been too difficult to perform on a standard laptop, and instead we had to resort to investing in using a GPU instance from Amazon Web Services, in total spending approximately $50 USD on all of our deep learning attempts.

The benefit of traditional machine learning is that it is very easy to begin training and testing models, even using a pretty basic computer, thanks to the availability of a number of fast and user-friendly libraries (e.g., Scikit learn).

An overall summary of the two different approaches based on the experience from this project is shown in table 7.

Table 7: Summary of Machine Learning & Deep Learning approaches to the project.

	Machine Learning	**Deep Learning and CNNs**
Classification Accuracy	Relatively poor (<40%)	Excellent (75%+)
Training Speed	Mostly quite fast (<1 hr)	Anything from 1hr for simple transfer learning to 3-weeks+ for training a full network.
Testing Speed	Very Fast: 0.25 ms per image (Using MacBook Air, 2.2 GHz Intel Core i7, 8 GB 1600 MHz DDR3 RAM)	Fast: 24 ms per image (AWS g2.2xlarge GPU Instance, Intel Xeon E5-2670 CPU, 1x NVIDIA GPU 1,536 CUDA cores, 4GB Memory)
Ease of getting started	Easy: libraries mean time can be spent on feature selection rather than creating models from scratch	Hard: Requires additional tools and resources and more time spent setting-up a model
Resources Required	Low	Medium-High
Feature Selection	Greatest time investment is in selecting and testing different types of features.	Very easy; the model selects features during training.
Overall	**	*****

In conclusion, whilst it is true that Deep Learning with CNNs requires more computing resources, the truth is that it is now cheaper and easier than ever to be able to access the required processing power through services like Amazon Web Services. During this project, a large part of the incurred cloud computing cost was due to lack of experience with the tools and techniques being used. Without this learning-curve it should be possible to carry out a similar exercise in far more quickly and cheaply.

Additionally, the significant improvement in test results clearly demonstrate that future efforts would be best focused on continuing to optimize CNN-based methods.

This is not to disparage standard machine learning which continues to be a very powerful toolkit and a more than adequate solution for a number of problems. However, when it comes to computer vision, it is clear why such a strong emphasis is currently placed on Deep Learning and CNNs.

7 Conclusion and Recommendations

As a proof of concept, we believe that the results of this project demonstrate that it would be possible to implement image recognition in a recipe search and retrieval user flow.

1. We were able to achieve a good overall F1 score in a relatively short space of time once we started using CNNs
2. For some individual categories, the F1 score was even higher, close to or above 0.90
3. There still remain plenty of opportunities for further optimizing the model, and we would expect to be able to achieve an overall score of at least 0.80
4. Moreover, depending on the use case, performance could be improved further using aggregated classes
5. Prediction time of less than 1 second per image is certainly adequate, and could also likely be decreased by using a more powerful GPU
6. People are even working on implementations of CNNs that can be run in-browser or on smartphones in real-time

Our recommendations for next steps are:

- Expand the model to include all 101 food categories from the original dataset.
- Seek to increase the number of images by looking for other sources of data.
- Invest more time in optimizing the model.
- Consider a pilot based on using a smaller set of 10-15 consolidated food categories.
- Explore more advanced computer vision techniques such as object detection and semantic segmentation for identifying multiple food items within images.

References

[1] http://www.ft.com/cms/s/0/f609954c-1d46-11e6-a7bc-ee846770ec15.html

[2] http://www.epicurious.com/about/press-center

[3] http://www.business.com/social-media-marketing/food-photo-frenzy-inside-the-instagram-craze-and-travel-trend/

[4] http://press.allrecipes.com/

[5] http://research.microsoft.com/en-us/um/redmond/projects/menumatch/

[6] http://www.popsci.com/google-using-ai-count-calories-food-photos

[7] https://www.vision.ee.ethz.ch/datasets_extra/food-101/

[8] http://www.robots.ox.ac.uk/$\tilde{\text{v}}$gg/research/very_deep/

[9] Deep Residual Learning for Image Recognition, arXiv:1512.03385

[10] http://cs231n.github.io/transfer-learning/

[11] https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet

[12] Very Deep Convolutional Networks for Large-Scale Image Recognition, K. Simonyan, A. Zisserman, arXiv:1409.1556

A Appendix A: Category Histograms

Figure 11: Red Histograms for Average Images

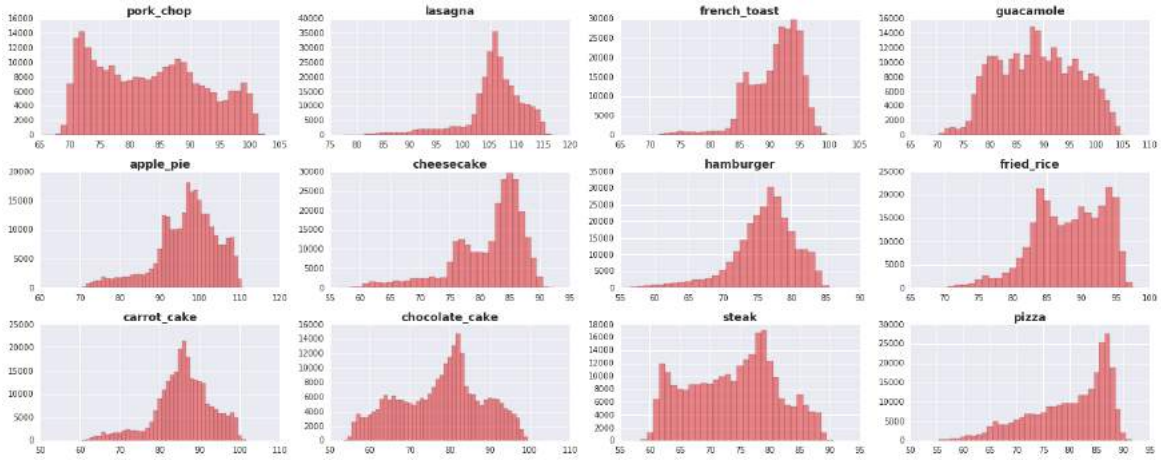
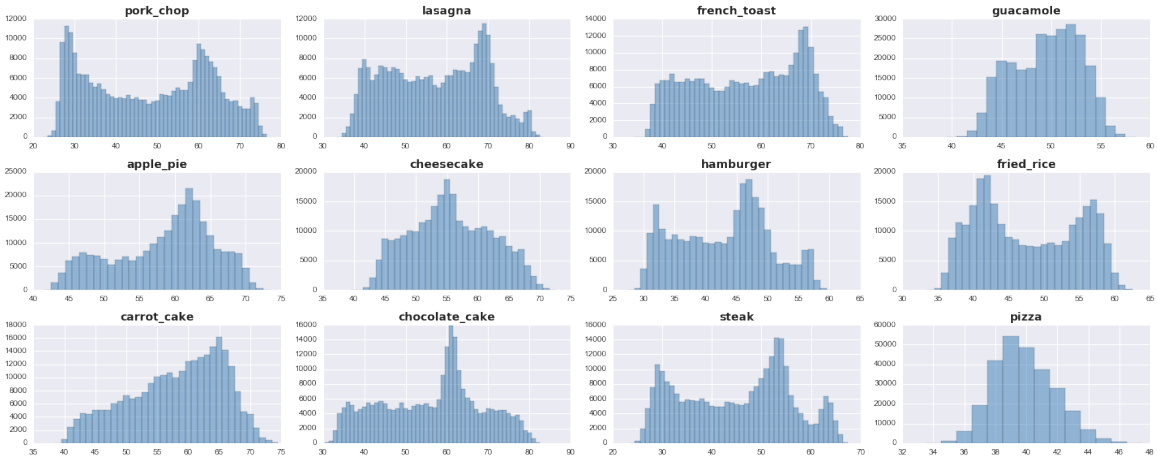


Figure 12: Green Histograms for Average Images

Figure 13: Blue Histograms for Average Images



B Appendix B: Machine Learning Models

	F1	**Features**	**Model**
1	0.04	Greyscale histogram of pixel values	Compare similarity to histogram of mean image of each class using intersection
2	0.07	RGB Histogram	Compare similarity to histogram of mean image of each class using intersection
3	0.06	RGB and Greyscale Histogram	Compare similarity to histogram of mean image of each class using intersection
4	0.16	Greyscale histogram of pixel values	kNN, k=10, Uniform voting
5	0.15	Greyscale histogram of pixel values	kNN, k=10, Weighted voting
6	0.19	RGB Histogram	kNN, k=10, Uniform voting
7	0.18	RGB Histogram	kNN, k=10, Weighted voting
8	0.18	RGB and Greyscale Histogram	kNN, k=10, Uniform voting
9	0.18	RGB and Greyscale Histogram	kNN, k=10, Weighted voting
10	0.10	Chained color histogram	Linear SVC
11	0.22	Chained color histogram	Support Vector Machine Polynomial Kernel
12	0.17	Chained color histogram	Decision Tree, max_depth=5
13	0.24	Chained color histogram	RandomForestClassifier(max_depth=12, n_estimators=40, max_features="sqrt")
14	0.21	Chained color histogram	Ada Boost Classifier
15	0.18	Chained color histogram	Gaussian NB
16	0.18	Chained color histogram	Multinomial Naive Bayes
17	0.07	Chained color histogram	Bernoulli Naive Bayes
18	0.20	Chained color histogram	Linear Discriminant Analysis
19	0.06	Chained color histogram	Quadratic Discriminant Analysis
20	0.08	k Means Clustering, k=100	kNN, k=5, Uniform voting
21	0.09	k Means Clustering, k=100	RandomForestClassifier(max_depth=5, n_estimators=15)
22	0.05	PCA, 100	kNN, k=9, Uniform voting
23	0.07	PCA, 100	RandomForestClassifier(max_depth=5, n_estimators=10)
24	0.13	32x32 images, pixels as features	KNeighborsClassifier(n_neighbors=9), uniform
25	0.19	32x32 images, pixels as features	RandomForestClassifier(n_estimators=14, max_depth=7)
26	0.17	RGB Histogram + Pixels chained together	KNeighborsClassifier(n_neighbors=9), uniform
27	0.26	RGB Histogram + Pixels chained together	RandomForestClassifier(n_estimators=70, max_depth=10)
28	0.27	RGB histogram, edge and corner features	Random Forest, 100 Estimators, max depth 8
29	0.32	32x32 boxes, avg RGB pixel values, edges, corners	Random Forest, 500 Estimators, max depth 14
30	0.23	16x16 boxes, avg RGB pixel values, edges, corners	Random Forest, 500 Estimators, max depth 10

Machine Learning Models Cont...

	F1	Features	Model
31	0.27	Features from 16x16 boxes, avg RGB pixel values, edges, corners; PCA n=30 to reduce features	Random Forest, 250 Estimators, max depth 14
32	0.13	32x32 boxes, avg RGB pixel values, edges, corners PCA n=50 to reduce features	Random Forest, 500 Estimators, max depth 14
33	0.27	32x32 boxes, avg RGB pixel values, edges, corners Sort according to features importance, top 50 of each dimension	Random Forest, 500 Estimators, max depth 14
34	0.29	32x32 boxes, avg RGB pixel values, edges, corners	Separate Random Forests for RGB Edges, Corners; Use Bayesian Net to combine output of individual classifiers and make predictions
35	0.27	32x32 boxes, avg RGB pixel values, edges, corners	Separate Random Forests for RGB Edges, Corners; Use Linear SVC to combine output of individual classifiers and make predictions
36	0.29	16x16 boxes, avg RGB pixel values, edges, corners	Separate Random Forests for RGB Edges, Corners; Use Bayesian Net to combine output of individual classifiers and make predictions
37	0.26	16x16 boxes, avg RGB pixel values, edges, corners	Separate Random Forests for RGB Edges, Corners Use Linear SVC to combine output of individual classifiers and make predictions
38	0.21	Find segments in an image and calculate RGB-based features for each segments Specific features calculated are: Max, Min, Mean and Range of RGB values Normed histograms of RGB values for predfined bins; 100 segment	Random Forest, 200 Estimators, max depth 14
39	0.28	32x32 boxes, avg RGB pixel values, edges, corners minmax scaler	Linear SVC, C=0.001