Computer Vision for Food Cost Classification

Dash Bodington
Stanford University
dashb@stanford.edu

Abstract

This project aims to use various computer vision methods for the novel task of restaurant cost prediction based on food images. It uses the Yelp dataset, a dataset of 200,000 images tagged by business, which is significantly pared down to near 15,000 relevant images tagged as food, and labeled by the cost rating of the business they come from. These food images are partitioned into training and testing datasets, and the test dataset has a uniform sample distribution over the classes: cheap ($-$$ on Yelp), and expensive ($$$-$$$$ on Yelp).

Once the dataset is defined, several techniques are trained and applied to make class predictions on the test dataset. The prediction model is broken into feature extraction and classification, and significant exploration was done to come up with optimal features and classifiers for the specific task. The primary goal of this project was to achieve the highest possible classification accuracy on the reserved test dataset.

The four main feature extractors used were: simple preprocessing (crop and resize images to uniform dimensions), color histograms (a normalized distribution of color intensity at all pixels), SIFT bag of words (using normalized distributions of SURF feature common descriptors over each image), and deep neural network features (using the trained Alexnet to extract image features).

Each feature extractor was tested with each of the following classification methods: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Naive Bayes, and a custom neural network with softmax loss.

The best results were achieved from the combination of Alexnet features and color histograms, independently compressed using PCA, and classified with a shallow neural network, which acheived 70% accuracy on the test set while most other methods achieved from 50% - 78% accuracy.

1. Introduction

Since various deep learning methods have shown very impressive results in classification, detection, and localization, there has been a great deal of focus on understanding images beyond their still content, mostly in the form of generating descriptive sentences, and using semantic analysis to understand language [6, 7] to generate these descriptions. While generating descriptions is easy for a human to evaluate and judge a perceived intelligence, it is only a subset of what can be understood beyond image content through computer vision.

This project aims to tackle a subset of beyond-content image understanding through food images and restaurant ratings from the Yelp dataset. The dataset contains 200,000 images tagged with descriptors and the businesses they come from, and businesses are give a cost rating from $ to $$$$. Because of the small number of restaurants in the two most expensive classes, these cost ratings are compressed into 1-2 $ and 3-4 $, hereafter referred to as cheap or $ and expensive or $$ By filtering images by 'food' tagging, and assigning the business cost rating to the image, a dataset containing food photos and an estimated cost ranking, or fanciness ranking is created. Many computer vision approaches can be used in an attempt to understand these food images beyond their straightforward content. The simple goal of this project is to train algorithms on the training data, and maximize accuracy on the test dataset.

Of the possible approaches to this problem, this project focuses on predictive models divided into feature extractors and classifiers, and explores several options and combinations between the two model parts. Four primary feature extractors used were: simple preprocessing (crop and resize images to uniform dimensions), color histograms (a normalized distribution of color intensity at all pixels), SURF bag of words (using normalized distributions of SIFT feature common descriptors over each image), and deep neural network features (using the trained Alexnet to extract image features). Each feature extractor was tested with each of the following four classification methods: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Naive Bayes, and a custom neural network.

Very little work similar to this project has been done, but the project relies heavily on more general, existing computer vision tools and research. The remainder of this report will discuss the dataset, models, and results, and conclusions of this project in detail.

2. Implementation

2.1. Experimental Setup

The implementation of this project was completely computational, and involved dataset extraction, data preprocessing, feature extraction, and classification. All code for this project was written in python 2.7, opencv was used for most image processing [2], tensorflow [1] was used to write all neural networks, and scikit provided implementations of some other classifiers[5].

The project was run on a fast desktop computer. All neural network models were run on an Nvidia GTX 780 GPU, and the remainder of the processing was done on a 4.4GHz quad-core CPU. Even with reasonably fast hardware, feature extraction was very time consuming, so it was often done only once, and the features were cached to be used on-demand.

2.2. Dataset Extraction

Though this project uses a nicely formatted and labeled dataset, the subset required for training and testing comes from several extraction steps. The Entire Yelp dataset contains 200,000 color images from business' Yelp pages. Images can be uploaded by either businesses or customers, and are of varying quality and size. These images are tagged by Yelp's own computer vision algorithms, and can be corrected by users, as there are sometimes errors in tagging.

Initially, the image database is analyzed, and all images tagged as 'food' are extracted along with the 'id' of the business they come from. All images from the same business are grouped, and are then labeled with the consolidated cost label (mapped from Yelp's \$ - \$\$\$\$ to the binary \$ or \$\$ for this project) of the business. If a business has not been labeled with a cost rating, which happens infrequently, the corresponding images are discarded. Next the businesses ids are binned by their attached cost ratings, each bin is shuffled, and a predetermined train/test ratio (0.7 in this case) is used to assign each business id to the training set or test set. This binning by business and cost rating is used to ensure that there is a similar distribution of data in the test and training datasets, and to avoid placing images of the same item from the same business into the training and test dataset, which could be considered as mixing the two. In total, there are 11,530 images available for training, and 4,412 images available for testing, though these counts decrease depending on the desired training and testing distributions.

In the test dataset, images in each class are shuffled, and images are discarded from the class with more images until exacly 50% of the images in the test set are from each class. Enforcing this 50% distribution cuts the dataset approximately in half because of the relative rarity of expensive restaurants.

After the training and test datasets are fully defined, each image is cropped from the center to the largest square area possible, and is resized to a variable size, depending on the feature extractor used (sizes range from 64x64 to 227x227).

With this definition of the training and test datasets, there are several further processing steps which are sometimes used on the training set to improve training and performance.

- For validation or cross validation, which is only used when tuning feature or classifier parameters, or training neural networks, 20% of the training dataset is randomly designated as the validation set. All models except neural networks are trained on the whole training dataset before testing.
- Depending on the classifier and loss function being used, the training set will usually have images discarded in the same fashion as the test dataset to even the distribution of images across classes.

2.3. Feature Extraction

Several feature extractors were used in this project as inputs to the various classification systems.

2.3.1 Images as Features

In some cases, preprocessed (scaled and cropped) images were used as features themselves. This method is usually most appropriate for input into a convolutional neural network classifier, which essentially defines its own features internally, but can also be used with other classifiers with varying success. These features use images of size 128x128 or 64x64, which are vectorized unless the classifier is a convnet.

2.3.2 Color Histogram Features

As an initial step beyond images as features, it was thought that the colors in an image could give an indication about the cost of food. This feature consists of a length 30, normalized (sum 1) vector, which contains a 10-bin histogram for each color channel. These features use images of size 128x128 or 64x64 as inputs.

2.3.3 SIFT Bag of Words Features

SIFT Bag of Words features consist of frequency distributions of common feature descriptors. During training, a set of SIFT descriptors from all training images is extracted, and N (range: 20-100) 'words' are extracted with the K-means algorithm [2, 4]. Next, during training and testing, each image's feature vector is calculated as a normalized N-length histogram of the words, where each feature descriptor from the image is assigned to one word with the nearest neighbor algorithm. These features use images of size 128x128 as inputs.

2.3.4 Alexnet Features

Alexnet is a 13-layer pretrained convolutional neural network which previously achieved state of the art performance in the Imagenet large-scale image classification challenge [3]. Originally, the network's output was a 1000-class softmax layer, but because the network learns very useful features for other tasks, swapping the final layers of the network for custom-trained layers is a common practice, especially for those working without the computational power or large data volume required to train a model with similar performance from scratch. For this project, 'Alexnet Features' are considered to be the output of one of the final layers of the pretrained network.

With Alexnet, multiple feature sets were created from multiple layers of the convnet. fc8, fc7, and fc6 (the final fully-connected layers of the network). Because these features are effectively sparse (we are feeding the network a very small subset of the images it was trained to classify), and very large in size, PCA is often used to reduce the feature dimensionality before training.

This network requires image inputs of size 227x227, which are also mean-subtracted.

Using the GPU allows for a significant speedup in feature generation, over 70x faster than CPU computation on the hardware for this project, but is not enough to train a network like Alexnet from scratch with limited time and data.

2.4. Classifiers

Because of the many feature inputs used in this project, classifiers with different properties and strengths are used to increase the likelihood of good performance with each feature set. Feature vector lengths range from 30 (Color Histogram) to 12,288 (cropped and rescaled images), so models which may overfit in some cases, may perform well with fewer features.

2.4.1 K - Nearest Neighbor

The K-Nearest Neighbor classifier archives the entire training feature set, and at prediction time, calculates the euclidean distance from the input feature vector, and makes a prediction based on a majority vote from the labels of the K closest training examples (K = 20 for this project). While storage-inefficient, it is one of the simplest classifiers, and would perform well if images in the training and testing dataset were similar enough, but fails to generalize otherwise.

2.4.2 Naive Bayes

Naive Bayes classification assumes feature independence and Gaussian probability distribution of features to make a maximum likelihood estimate $\hat{y} = argmax_y P(y)\Pi P(y|x_i)$ of the class, and usually can perform well on small training datasets because it has few parameters.

2.4.3 Support Vector Machine (SVM)

The SVM is a linear classifier which defines a class-dividing hyperplane $f(x) = B_0 + B^T x$, which minimizes $\frac{1}{2}||B||^2$ subject to $y(B_0 + B^T x) \geqslant 1$ on the training set. Generally, SVMs have the advantage that they are less likely to overfit than other methods because of build-in regularization.

2.4.4 Neural Networks

While neural networks are the most general classifier used in this project, they are also the most difficult to tune, and it requires a great deal of data to train networks with many neurons. Neural networks are a layered structure of interconnected linear and nonlinear operations whose parameters can be learned with various gradient descent methods.

In this project, the classification networks presented have zero or one hidden layers, and all layers are fully connected. Each fully connected hidden layer (when present) is followed by a Rectified Linear Unit (ReLU), and the final layer is a softmax layer which takes two inputs and computes pseudo-probabilities for the input on each class according to

$$\hat{p}(x_i) = \frac{e^{x_i}}{\sum e^{x_i}}$$

.

Though neural networks have been responsible for many recent state of the art results in computer vision, they are among the most difficult models to manage on small datasets. Because of this, L2 regularization is sometimes added to the iterative minimization of the cross-entropy loss

$$Loss = -\sum p(x_i) log(q(x_i)) + \lambda R(Weights)$$

where q contains the outputs of the softmax layer, and p is the one-hot label vector for the training sample x. Other training tricks, such as dropout and projection of sparse feature vectors into lower dimensions with PCA are also attempted to increase robustness and decrease overfitting. Batch gradient descent with momentum was used to train networks for this project because it provided reliably converging results, especially when changing the class distribution (and size) of the training set.

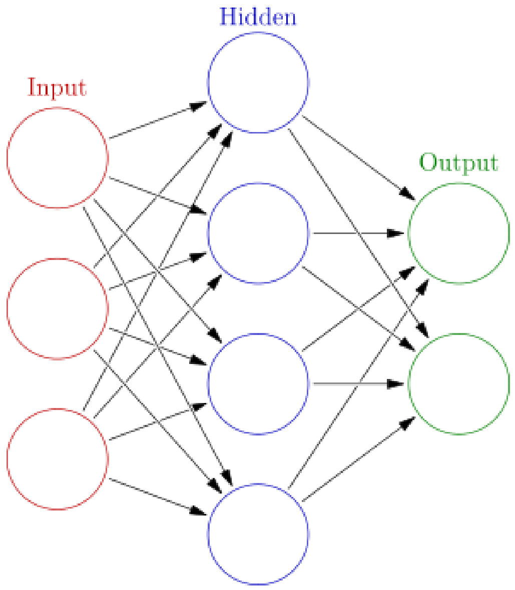

Figure 1. Example of classification neural network. A ReLU is applied to the hidden layer, and a softmax normalization is applied to the output.

3. Results

Though estimating restaurant cost-class from food images is a difficult problem, reasonably good results are tabulated and discussed below.

3.1. Accuracy

	K-NN	NB	SVM	NN
Images	0.56	0.60	0.50	
Color Histogram	0.57	0.63	0.54	0.56
SIFT + BoW	0.59	0.62	0.60	0.60
Alexnet	0.62	0.65	0.66	0.68
Multifeature	0.55	0.65	0.68	0.70

Table 1. Multifeature methods (Alexnet fc8 and color historgrams) with a neural network classifier perform the best on the test set.

3.2. Example Classifications

Though accuracy results are the goal, actual examples of results provide a more intuitive look at classification. We can see that desserts appeared frequently in the $$ class, which is not surprising, considering that they are often considered to be a luxury item, and $ -classified foods are more everyday items.

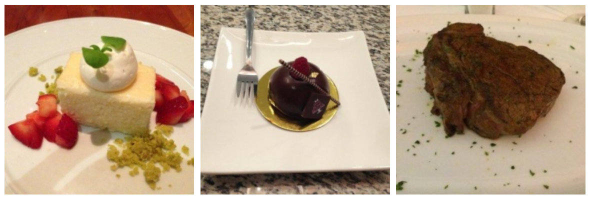

Figure 2. These three images of cheesecake, chocolate dessert, and steak, are the images with the highest estimated probability of being expensive by the neural network.

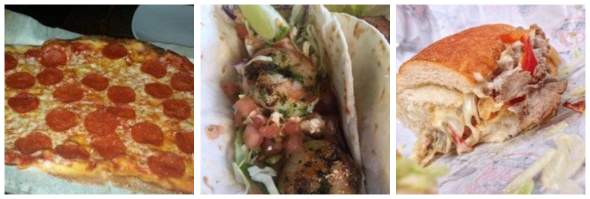

Figure 3. These three images of pizza, a taco, and a sandwich, are the images with the lowest estimated probability of being expensive by the neural network.

3.2.1 Details of Best Model

The best performing model, 'multifeature,' which achieved an accuracy of 70% on the test dataset was a multifeature model with Alexnet features, color histogram features, and a neural network classifier.

For this model, 1000-dimensional Alexnet features were extracted from the 8th fully connected layer of the network (the layer closest to the original softmax layer) and compressed to 100 dimensions and whitened with PCA. Then, whitened color histogram features were concatenated with the Alexnet features, leaving feature vectors of length 130 for the classifier.

The classifier in this model was a fully connected neural network with a hidden layer of size 50 (see Figure 1). This model was trained for 10,000 iterations of batches of 200 images, reporting validation error every 500 iterations. After full training, the model from the iteration with the highest validation accuracy was used for testing.

3.2.2 Unsuccessful Models

In addition to the previously presented models, many other less successful attempts were made to solve this task. Some simply performed poorly, were not unique, or were computationally unfeasible.

- Convolutional neural networks were implemented, but failed to train or generalize well. It is thought that with the large image size required (greater than 128x128) to resolve objects in detail in these images, the training dataset was not large enough to train the large filters also required. At small image sizes (64x63-128x128) with smaller filters, models did not generalize, likely due to the lack of recognition at such a pixelized scale.

- Other faster feature extractors were also tested for the bag of words method, such as SURF and BRIEF, but since SIFT features were the best performing, only they were presented.
- Linear Discriminant Analysis was used for classification as well, but performed similarly to the SVM, so the decision was made to focus on fewer, more unique, classifiers.
- Many attempts were made to train models on imbalanced datasets in order to have more training data, however, with the imbalanced nature of the data (with 75% of the images in the $ class), many models simply learned to predict $ consistently, since this would lead to a higher accuracy than varied predictions.
- To combat the problem above, various loss functions and update methods were tried, such as weighting each misclassification by (1 - the prior probability), and only performing updates for misclassified images. Ultimately, training on an even dataset was the most successful.

4. Conclusion

High performance in abstract computer vision tasks like this one is very difficult to achieve, which was a lesson learned during this project. Though accuracy of 70%, significantly above random (50%) was achieved, higher performance would certainly be necessary before using a prediction system like this for an unsupervised application.

4.1. Challenges

The greatest challenge in achieving good performance in this task seemed to be working with unbalanced or small datasets. Though near 15,000 food images were available, the rarity of expensive restaurants made training difficult, and discarding images to even the datasets led to a significantly smaller volume for training, though results improved.

Beyond data problems, there may also be issues with the labeling of the dataset. Yelp uses their own neural network to tag images, so some images which were tagged as food (such as salt and pepper shakers, and a hotel key), were not food. Though the frequency of this was small, it may have had an effect on performance. Additionally, the $ rating system is not perfect for this task. A Yelp $$$ restaurant in an expensive location could serve similar dishes to a Yelp $$ restaurant elsewhere, so there is certainly some blurring of the data across the class boundary, unlike most standard classification tasks.

4.2. Future Work

Moving forward, it is possible that discarding one of the middle classes $$ or $$$ from the Yelp dataset would create a set which is better divided between classes. If a dataset of photos and menu prices were available, it would be interesting to combine this cost category recognition with dish recognition to allow for prediction of actual dish prices.

70 % accuracy is satisfactory given the time and data constraints of the project, and every effort was made to raise this accuracy as high as possible. With more time and data, it would be interesting to train end-to-end convolutional neural network models, or focus more on fine-tuning state of the art models such as Alexnet or Google's Inception. As the state of the art in computer vision progresses, it is likely that tools more suited for this problem will arise, but for now, the best results of 70% have been achieved with two neural networks in combination with hand-written features.

4.3. Replicating Results

Code for this project can be found at the author's Github (github.com/DashBodington/cs231aProject), and the Yelp Dataset is available from Yelp. Code is written in Python 2.7, and requires Tensorflow, OpenCV with the feature extractors in opencv_contrib, NumPy/SciPy, and scikit-learn. The GPU implementation of Tensorflow is strongly recommended, and some CPU feature extraction methods may take several hours on the full dataset.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2

[2] G. Bradski. *Dr. Dobb's Journal of Software Tools*. 2

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 3

[4] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 2

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Ma-

chine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 2

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[7] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1