Helping Yelp: Restaurant Photo Classification

Kelvin Ritland – 34725119
kelrit@gmail.com

Gisli Thor Thordarson – 73996150
gisttor@gmail.com

Ronald Monillas – 42469114
r.monillas@alumni.ubc.ca

Abstract

We present methods to predict attribute labels for Yelp restaurants using user-submitted photos provided by Yelp in a 2016 Kaggle competition while avoiding the usual large hardware and time requirements of machine learning on images. After classifying images with a pre-trained GoogLeNet, we explored various multi-label classification algorithms: neural networks, random forests, SVMs, and logistic regression. The different prediction techniques were then compared and analyzed. We found that these models performed very similarly on the generated features and greatly outperformed a basic convolutional neural net trained with similar hardware.

1 Introduction

A Harvard Business School study published in 2011 found that each "star" in a Yelp rating affected the business owner's sales by 5-9% [7]. Because revenue comes from business advertising in online marketing companies like Yelp, crowd-sourced reviews from the user community are particularly vital. In addition, Yelp's users upload an enormous amount of photos everyday alongside their written reviews, especially in this present age of food selfies and photocentric social storytelling. To tackle their ambitious goal of holistically understanding pictures, Yelp challenged aspiring data scientists in a Kaggle competition in 2016 to build a model that correctly performs multi-label image classification on Yelp data.

For this competition, participants are tasked with creating a model that predicts binary business attribute labels for restaurants based on the restaurant's photos [12]. The provided dataset consists of user-submitted photos for a subset of restaurants on Yelp, where each restaurant has a variable number of photos. Nine user-reported business attributes are also attached to each restaurant and take on "yes" or "no" values. These attributes are as follows:

0. good_for_lunch
1. good_for_dinner
2. takes_reservations
3. outdoor_seating
4. restaurant_is_expensive
5. has_alcohol
6. has_table_service
7. ambiance_is_classy
8. good_for_kids

As an image classification problem, there are inherent computational challenges. Many current classifiers (primarily convolutional neural networks) for images take days or even weeks to train on clusters of GPUs [9, 10], making iteration time and hardware cost very high. For an average researcher or Kaggle participant, such requirements could be unrealistic.

In this paper, we address the problem of automatically tagging restaurants with multiple labels on commodity hardware. We investigate various multi-label classification algorithms in the context of computer vision and show that some methods work better than others for this application.

2 Related work

Various attempts in large-scale multi-label image classification have been explored in the past. Prominent work in this domain includes ImageNet classification with deep neural networks [6], the rise of multi-label LDA [11], and the introduction of randomized clustering forests [8]. Krizhevsky et al. [6] achieved the winning test error rate in the ImageNet Large-Scale Visual Recognition Challenge in 2012 by training a large, deep convolutional neural network to classify millions of images. Considered a benchmark in image classification performance, this work highlighted the use of CNNs as an accurate image classifier and significantly contributed to our own approach.

However, the nature of the ImageNet challenge differs from that of the Yelp challenge. ImageNet posed a single-output multi-class problem, where the classes were mutually exclusive and each image belonged to only one of many classes. Meanwhile, Yelp's challenge involves multiple labels with binary values, where the classes are not mutually exclusive and each image can be linked to any number of classes. Wang et al. [11] proposed to tackle the task of multi-label image annotation using a modified version of linear discriminant analysis. While this novel method demonstrated promising results on five different datasets, multi-label LDA requires label correlations to work. Given limited time constraints, we were not able to extract this information, leaving us with similar linear-boundary-based techniques like logistic regression and support vector machines to try on the data.

In addition, previous efforts related to the use of random forests in content-based image recognition, like that of Moosmann et al. [8], influenced our decision to experiment with a random forest as an image classifier. Not only is the 'extremely randomized clustering forest' (their newly devised variant of random forests which combines clustering techniques with decision trees) fast, accurate, and scalable, but also it can be run on unlabeled data. However, literature related to this seemingly ideal algorithm is very minimal and implementing it would be difficult and time-consuming.

3 Data

As per other Kaggle competitions, the data provided by Yelp was split into a training set, where the business attributes were supplied, and a test set, where the business attributes were withheld. The training set consisted of 2,000 restaurants, each with a variable number of photos, with 234,842 photos in total. The test set consisted of 237,152 photos of 10,000 restaurants, some of which were fake for the purpose of deterring competition participants from hand labeling. Some sample images are shown in Figure 1. A CSV file described the mapping from restaurants to photos for the training and test sets. The business attributes supplied to the training set were collected by Yelp through photo captions, photo attributes, and crowd-sourcing [2], and were assumed to be their true labels. The Yelp challenge makes a good machine learning task because of the vast amounts of data that could be used to solve the classification problem.

Because restaurant labels are manually selected by Yelp users when they submit a review, there are several potential issues. Some restaurants are only partially categorized while others are not categorized at all. This was mainly because selecting the labels is optional. Furthermore, there might have been cases where a user submitted the same photo twice or a business chain submitted the same photo to its branches. As a result, some images were duplicated. We did not consider these factors when constructing our models but they could be subjects for further research.

Due to the unavailability of the test set output and competition limitations (e.g. the competition required that submissions come from single-person teams only), assessing model performance on the provided test set proved to be inconvenient. As an alternative, we tried to emulate the conditions of the competition by randomly splitting the provided training set (whose output was known) into two parts; 10% of the provided training set became the validation set and the remaining 90% became the 'new' training set. Consequently, the performance of the models was evaluated on the validation set and the provided test set (whose output was not given) was discarded.

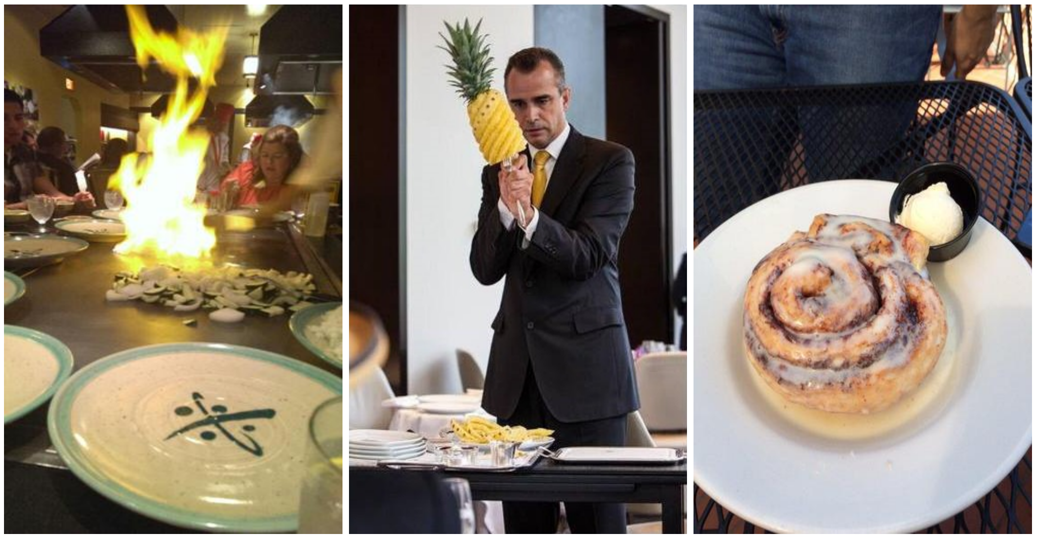

Figure 1: Sample images from the Yelp dataset. (1) Does the restaurant have a classy ambience? (2) Is this restaurant good for children? (3) Is this restaurant expensive?

4 Methods

In this section, we discuss the techniques that we investigated for multi-label image classification.

4.1 Evaluation

In order to compare the results of the different models, we used the mean F1-score on our validation set (10% of the given businesses). The F1-score is the metric used in the Kaggle competition and is computed as follows:

$$F = 2 \cdot \frac{pr}{p+r} \tag{1}$$

where, if tp, fp, fn are the respective true positives, false positives and false negatives of the predictions:

$$p = \frac{tp}{tp+fp} \qquad \text{and} \qquad r = \frac{tp}{tp+fn} \tag{2}$$

The relevant statistics are precision p, the ratio of true positives to all predicted positives, and recall r, the ratio of true positives to all actual positives. The F1-score, which ranges from 0 (worst) to 1 (best), weights recall and precision equally. Good performance relies on maximizing both p and r.

4.2 Direct convolutional neural nets

We first tried using the current state-of-the-art technique for image recognition, convolutional neural nets (CNNs). These are neural networks that have ‘convolutional’ layers: layers that use a set of 2D convolutions to produce the inputs to the next layer. These convolutions take into account how images tend to have spatially localized features, and are learned during training time.

A challenge with directly applying a CNN to the dataset is that each business in the training set corresponds to a variable number of images. Therefore, directly training a CNN on the data would require the network to be able to take a variable number of inputs, such as a recurrent neural network. However, as an initial attempt, we simply tagged the images with the labels of their corresponding business, and trained a CNN on this data.

Using the machine learning framework Keras [1], we tried a variety of architectures and parameters for our CNN, such the one in figure 2. We used ReLU rectifiers as activation layers. We down-sampled the images to sizes that range from 90×90 to 256×256, applied 50-80% dropout, and

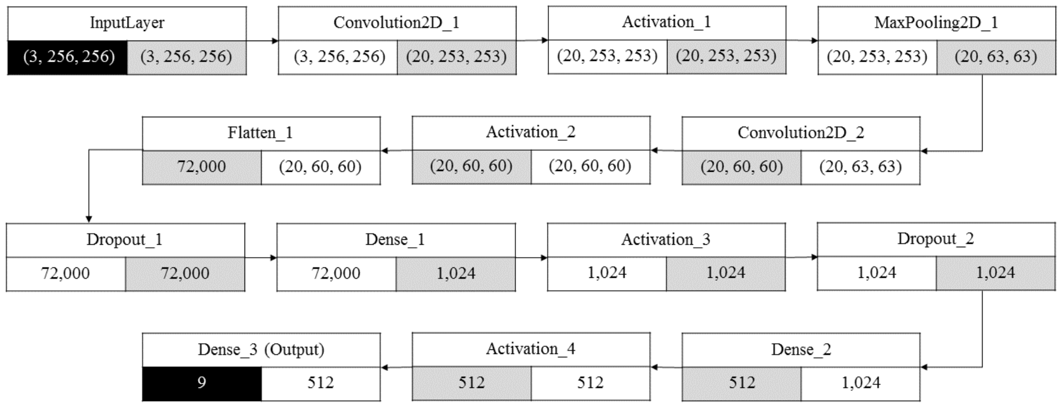

Figure 2: An illustration of an architecture for our CNN. Each box represents a layer of the network, where each layer consists of an input and an output. Sections shaded in gray describe the dimensions of that layer's output; sections shaded in white describe the dimensions of that layer's input; and sections shaded in black describe the dimensions of the entire neural net's input and output.

employed various optimizers (Adagrad [3], Adam [5], Adadelta [13]). Training was done on a GTX 770 GPU and training time took about 20 minutes to 4 hours per CNN.

Interestingly, the results were uniformly inadequate. For instance, some classes ended up being assigning the same every picture, regardless of its content. The best mean F1-score was 0.6660.

4.3 GoogLeNet labeling

We conjectured that training adequate models directly on the training set would take very long periods of time and require more hardware than what was accessible. To sidestep such requirements, we used a pre-trained CNN for the ImageNet Large-Scale Visual Recognition Challenge, GoogLeNet [10]. GoogLeNet takes images of size 224×224 as input and uses three internal classifiers to generate three vectors of length 1000 as output for each image. Each entry of a vector represents the probability that the image belongs to one of 1000 predetermined classes.

We downloaded a pre-trained model of GoogLeNet, classified all the training images with the model, and used them as features for the subsequent models. We produced features for each business by taking the average prediction across all of the images of that particular business. Concretely, if image j has the class vector c_j (the concatenation of the three outputs of GoogLeNet) and B_i is the set of images for business i, then the overall feature vector for the business is

$$x_i = \frac{1}{|B_i|} \sum_{j \in B_i} c_j \tag{3}$$

Producing the labels from the images using the pre-trained net was done in Keras [1] on a GTX 770 GPU and took 4 hours to complete.

What follows is an exploration of different supervised machine learning algorithms on this data. For each of the following methods, the F1-score for each class can be seen in Figure 4. A comparison of the various methods based on validation set accuracy and mean F1-scores can be found in Figure 5.

4.4 Neural networks

Neural networks have provided the best results for image classification so far, so we first decided to try one on the generated features. We trained a simple neural network with two hidden layers with the Keras framework 3. The input layer was a vector of length 3000 (the averaged GoogLeNet prediction) and the output was a set of 9 nodes, one for each class. The classifier was trained using minimum hinge loss, 50% dropout, ReLU activation functions, and the Adam optimizer [5].

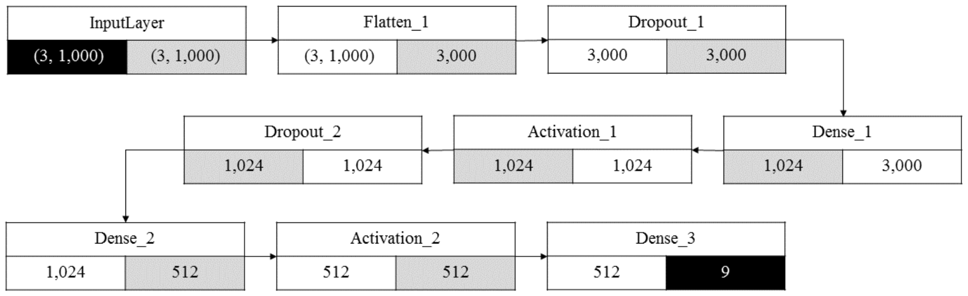

Figure 3: An illustration of the architecture of our neural network classifier. The color scheme is the same as in Figure 2.

4.5 Support vector machines

Support vector machines or SVMs are some of the most widely used learners in classification problems. Its kernel-based framework is powerful and flexible and a global optimum is guaranteed [4]. We implemented SVMs using `LinearSVC` which is found in the Python package `scikit-learn`. We used a support vector classification model with a linear kernel for speed and optimization purposes and tuned the regularization parameter by running a 3-fold cross-validation procedure. To handle the problem of predicting properties of a single object that are not mutually exclusive, we employed multi-label classification using `OneVsRestClassifier` from `scikit-learn`.

4.6 Random forests

The random forest is a popular ensemble learning method known for being applicable to both classification and regression tasks and for performing fairly well on any given dataset. Its strength lies in that it trains weakly correlated tree-based learners on bootstrapped samples of the data and outputs the majority result (for classification), yielding reasonably accurate predictions [4]. We used the Python implementation of a random forest classifier, `RandomForestClassifier`, from `scikit-learn`. A 3-fold cross-validation procedure was run to choose the forest size that minimizes prediction error. Like the implementation for SVMs, `OneVsRestClassifier` was used to solve the multi-label classification issue.

4.7 Logistic regression

Although the underlying model highly assumes a linear decision boundary which rarely exists in practical applications, logistic regression is simple, fast, and easy to implement and interpret [4]. Logistic regression is implemented in Python using `LogisticRegressionCV` from `scikit-learn`. The regularization parameter for the L2 penalty is selected by a built-in cross-validation procedure, and multi-label classification is supported by a one-vs-the-rest scheme performed in parallel across all folds and labels.

5 Results

The results of our experiments are summarized in Figures 4 and 5. We see that the methods that used the GoogLeNet class labels performed similarly for every class and on average. In fact, their performance is close to that of the top rankings in the Kaggle competition. On the other hand, the CNN lagged far behind the others in most of the classes. Particularly, the CNN achieved an F1-score of 0 in classes 0, 3, 4 and 7; an F1-score of 0 occurs when the same label is predicted for every example. The mean F1-score obtained by the CNN is, on average, about 16% less than that obtained by the other methods.

In addition, putting aside the four hours that it took for GoogLeNet to classify the images (which needs to be done only once), the running time for training each of the models that used the GoogLeNet features (neural network, random forest, SVM, logistic regression) was less than 10 minutes. This is a massive improvement over the running time for training the CNN given the same hardware, which could be anywhere from 20 minutes to 4 hours. As a result, more techniques and iterations could be performed with the GoogLeNet-generated features than what could be performed with the raw images alone.

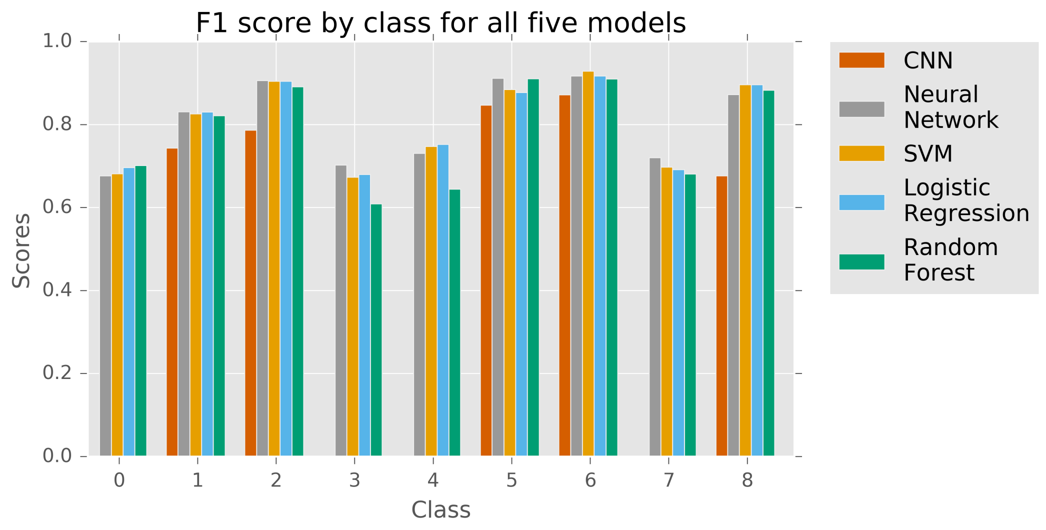

Figure 4: F1-score for each class for each of our models.

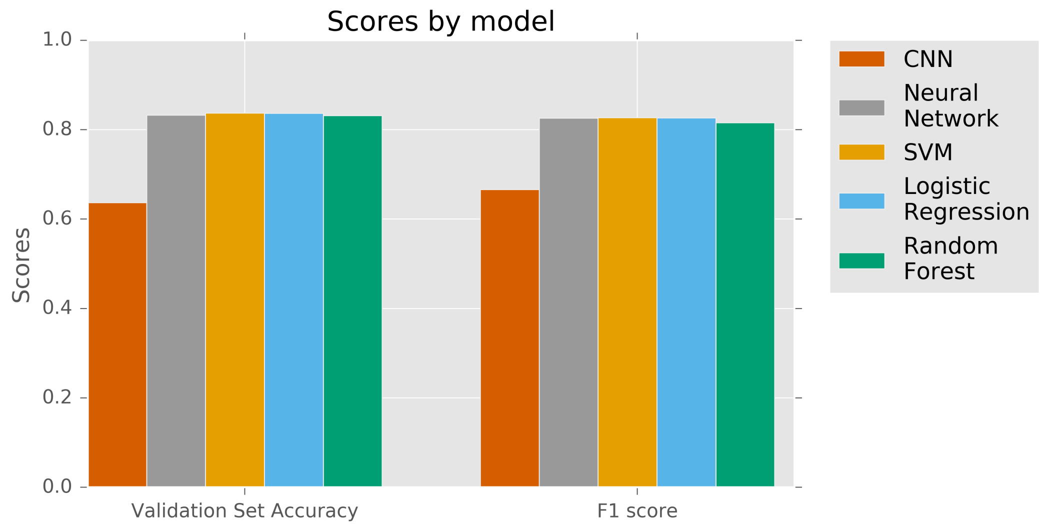

Figure 5: Validation set accuracy and mean F1-score for each of our models.

6 Discussion and Conclusions

Overall, we discovered that many standard supervised machine learning techniques worked similarly well on the data after applying the GoogLeNet class labels. We also found that despite using commodity hardware, this method of first classifying with GoogLeNet and then learning with standard models yields more accurate predictions and costs less time and memory than running a CNN. The widespread availability of these pre-trained models allows less well-equipped machine learning companies or individuals to achieve state-of-the-art results without needing access to an extensive memory or computing capacity (e.g. a cluster of GPUs).

While using the pre-trained network gave immediately better results, it may not always give the optimal features. For example, some classes, like *ambiance_is_classy* (class 7), always did not perform well. This suggests that this class is not well-represented by the image features.

Future work could address improving predictions for these classes. One direction could be replacing the 1000-node input layer of GoogLeNet (or some other pre-trained neural network) with a 9-node layer (one node per label) and training the network directly with the business labels. While going this direction would be much more intensive than training a simple classifier such as an SVM, we think that doing so would still be less computationally intensive than starting from an uninitialized deep CNN. To address the issue of each business corresponding to a variable number of images, we could concatenate the images into a long fixed-length vector of n images and put zeros where there are not enough images to fill the vector. We could then enforce a requirement that the network use the same weights for each image in the vector, essentially cloning the network n times. Afterwards, we could merge the layers together to form the neural network and obtain the predictions.

References

[1] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[2] Wei-Hong Chuang. How We Use Deep Learning to Classify Business Photos at Yelp, 2015.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[4] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[6] Alan Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, abs/1409.1556, 2012.

[7] Michael Luca. Reviews, Reputation, and Revenue: The Case of Yelp.com. Harvard Business School Working Papers 12-016, Harvard Business School, September 2011.

[8] Frank Moosmann, Bill Triggs, and Frederic Jurie. Fast discriminative visual codebooks using randomized clustering forests. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 985–992. MIT Press, Cambridge, MA, 2006.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *ArXiv e-prints*, September 2014.

[11] Hua Wang, Chris Ding, and Heng Huang. *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI*, chapter Multi-label Linear Discriminant Analysis, pages 126–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[12] Daniel Yao. Introducing the Yelp Restaurant Photo Classification Challenge, 2015.

[13] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.