Predict attribute labels for restaurants using user-submitted photos

Vinaya Polamreddi
vinaya@stanford.edu

Shubham Gupta
shubhamg@stanford.edu

Abstract

User uploaded photos contain a great wealth of information for Yelp. Yelp can provide its users valuable information if it can tap into this resource and understand characterisitics of the restaurants they belong to. Using Convolutional Neural Networks which have been proven to have state-of-the-art performance in many computer vision tasks including single label image classification, we will analyze the images belonging to a restaurant and classify the businesses according to a predefined set of labels. We utilize transfer learning to solve the task more efficiently and explore various ways to extend on top of pretrained networks. We find that using outputs from various layers of a pretrained model and building even slightly more complex networks on top of pretrained networks results in significant improvements.

1. Introduction

Yelp has millions of photos uploaded from all around the world. These pictures can provide valuable information and insights into the restaurants they are visually describing. You might want to know if a restaurant is good for a date or has live music or serves alcohol. Yelp normally gathers these labels as users review the restaurant but this data can be incomplete and hard to gather. However, we could probably understand the characteristics of a restaurant by looking at a few pictures of it as shown in 1.

Using the pictures uploaded to Yelp to automatically assign labels to the restaurants would be very beneficial. While a lot can be done with visual data, Yelp wants to first understand some attributes of each business. In today's culture of abundance photos, there are an abundant amount of photos added to each restaurant. Yelp challenges us to take this large amount of user-submitted photos and turn into a description of the restaurant.

We propose to tackle this problem using the deep learning architecture: Convolution Neural Networks to analyze the photos belonging to a business and understand the business. CNNs have achieved state-of-art performance in many computer vision tasks such as classification [10] and detection [6]. Yelp has used CNNs previously for classifying its images in different categories [1]. This gave them promising results and is widely used now. This indicates that the user submitted photos has a lot of information and insights if properly tapped.

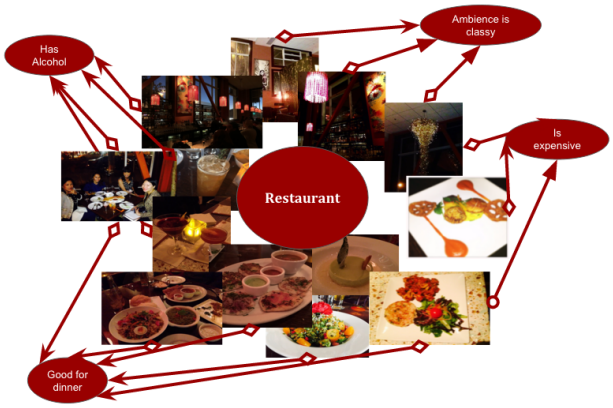

Figure 1. Images belonging to a restaurant that identify restaurant's attributes

In this problem, we are using a Kaggle dataset which consists of businesses with a set of labels and set of images. It is a multi-instance, multi-label problem where each business has many input features (images) and multiple labels. The input is a business represented by a set of images and we will use CNNs to output labels associated with the business. We explore a few neural network architectures and transfer learning in tackling this problem.

2. Related Work

Image classification has been a very well studied problem in the field of computer vision. Conventional approaches using hand-crafted features followed by classic classifiers such as random forests o SVMs have been beaten by deep learning architectures such as Convolutional Neural Networks (CNN) that automatically learn features. CNN's have achieved state of the art performance on many single label Image classification challenges such as the ILSVRC, and [10], [7] are the most cutting methods used to solve this problem. While single label image classification in which

each image is assigned a single label from a set of predefined labels has been very well studied; multi-label classification is still an open problem. Multi-label classification is a very real problem as many real-life images have more than one label. There have been many attempts at multi-label classification. In [2] and [3], the authors formulated the classification problem as a conditional modelling problem. Here, they are trying to incorporate the correlation among different labels along with the relation between images and labels. There has also been a lot of work on solving this problem using classifiers and using different image representations.

In [4], the authors presented a novel Rank minimization algorithm for solving the multi-label image classification problem. They modelled this as a problem of completing unknown label entries in a data matrix made up of training and testing features. One of the main claims is that it is robust to caveats like outliers and missing data. None of the previously described methods use deep learning which we saw beat the classic classifiers. In this [11], they explore how the ability of CNN's to do well on single label classification problem sheds on why they do poorly on multi-label classification: implicit assumption in CNN's of foreground object alignment and interaction between multiple-objects including occlusion and visibility. [11], proposed an infrastructure called Hypothesis-CNN-Pooling which uses the CNN architecture for multi-label classifying. They generate an arbitrary number of image segments called object segment hypotheses using objectiveness detection techniques. They connect each hypothesis with a shared CNN and aggregate the results from the shared CNN which is connected to each of the hypothesis to get the final multi-label classification.

In addition to this problem being multi-label, it is also multi-instance problem. In [13], they formalize multi-label, multi-instance learning as where a training example is described by many instances and multi-label 2. They continue to propose two algorithms: MIMLBOOST which uses multi-instance learning as a bridge and MIMLSVM which uses multi-label learning as a bridge. In [12], they deep learning for a lightly supervised setting where they use multiple instances in a bag that is labeled. In [9], they use a new formulation to multi-class instance learning by a fully convolutional network which accepts inputs of any size, does not need object proposal preprocessing, and offers a multi-class pixelwise loss for selecting latent instances. While the results are interesting, there are still many places for improvements before this can become close to state of the art.

Another important finding in training neural networks is the effectiveness of transfer learning [5]. On new tasks with small datasets pretrained nets consistently outperform randomly initialized nets [5], and image representations learned on large-scale datasets can efficiently be transferred to other recognition tasks with low data [8].

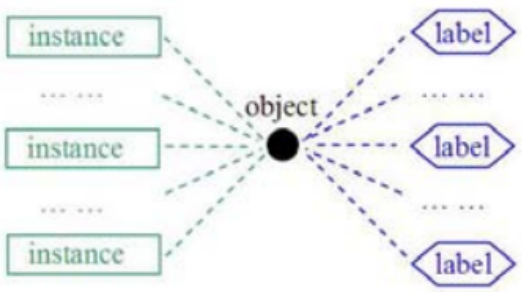

Figure 2. Multi instance, multi label classification [13]

3. Methods

3.1. Business to Image and back

In this problem, we are trying to assign multiple labels to a single business based on a set of images representing each business. In our training set, we know which labels belong to each business and which images belong to each business. The first problem we need to tackle is the relationship between the business labels and images. As explored in our related work, there are a few algorithms that explore multi-instance learning which is the relationship between businesses and images. However, these multi-instance algorithms assume that the object can be accurately described only by all the attributes in the instances; for example: a picture of a beach can be accurately described only if it has sand and water [12]. In this dataset, the instances seem to contribute individually and independently to the label of the object. As shown in 1, any single attribute for the business seems to be deducible by a single image; in other words, no attribute seems to need evidence from multiple pictures containing various scenes. While this seems like a multi-instance problem, in reality the instances independently provide attributes; therefore, instead of treating the instances in a bag, we will model this as a single instance classification problem where each image will separately be classified. We transfer all of the labels of the business in the training set to each of the images.

The second problem we face is understanding the relationship between each of the labels. Instead of two or more characteristics of an image combining to form a label, we can separately identify each of the attributes of a business. For example, in a photo of alcohol and dinner plates, the alcohol can confirm that the restaurant serves alcohol and darkly lit photos and fancy plates can confirm that the restaurant is good for dinner which seem independent of each other. Multi-label algorithms are especially useful when labeled data is sparse or in unsupervised settings [11]. In this case, where we have clearly labeled data and the la-

bels while related seem to be independently derived from an image, we believe that using state-of-the-art single label classification will perform superior to using less effective multi-label classification algorithms. We will predict each of the attributes for an image individually using single label binary classifiers and concatenate the output of each of the classifiers to get full set of labels for an image.

Once each of the images of a given business has labels predicted, for each label, we get the proportion of images for that business that were classified as having that label and take all the labels past a certain threshold to be the labels of the given business. The threshold is a hyperparameter determined using the validation set.

3.2. Image Classification

3.2.1 Transfer GoogleNet trained on ImageNet

Transfer Learning has been proven to be an effective way to tackle classsification problems where the features of one task can be generalized to another and the data set is sparse. Due to sparsity of training data and a shortage of computation resources for our problem, transfer learning and fine-tuning minimal parameters seemed the best way to approach this task. For all of our methods, we used the GoogleNet architecture with weights pretrained for the ImageNet challenge as the first building block of our networks. We used the output of 3 different layers from the GoogleNet in various ways: Inception 4a, Inception 4d and Inception 5b layers as shown in Figure 8. All these layers correspond to the dense layers before the 3 output and loss layers of the GoogleNet. We used the BVLC Caffe Deep Learning Framework from UC-Berkeley and BVLC Model Zoo as well as Apollocaffe to implement the following networks.

3.2.2 M1: Fully Connected layer on Inception 5b output

In the first model as shown in fig 4, we trained a network consisting of a single fully connected layer using the Inception 5b output as input. The fully connected layer takes an input vector of size: N * 7 * 7 * 1024 where N is the batch size, and outputs a vector of size: N x 2 x 1 x 1. We used a softmax activation and loss to train the fully connected layer. For each of the 9 attributes, we trained a separate network which performed binary classification on the image for each label.

Softmax Loss is defined as:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{1}$$

or equivalently

$$L_i = -f_{y_i} + \log\sum_j e^{f_j} \tag{2}$$

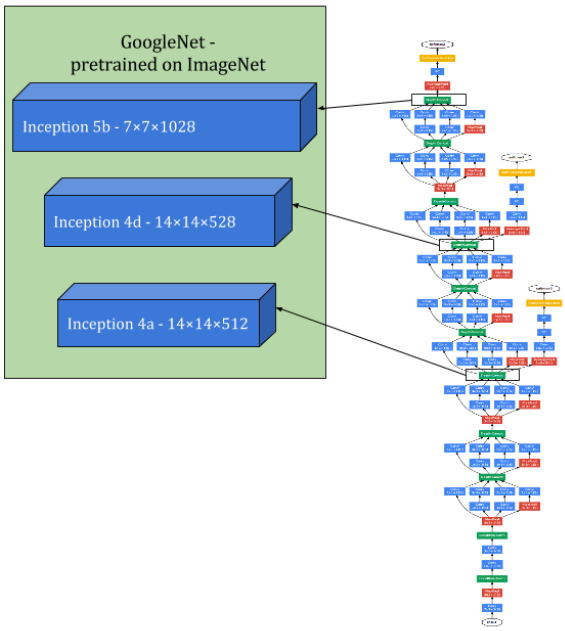

Figure 3. Google net output layers we used as input for other methods

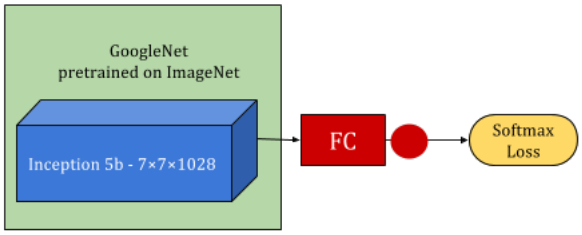

Figure 4. Training a fully connected layer on top of Inception 5b output

3.2.3 M2: Ensemble using inputs from outputs of various GoogleNet layers

In the next model, we ensemble the output of 3 nets as shown in fig 5 The three nets take in as input the output of Inception 4a, Inception 4d and Inception 5b layers respectively. They each have a single fully connected layer with a softmax activation and loss trained on the dataset for 15 iterations. The fully connected layers output a vector of size: N x 2 x 1 x 1 each of which are then ensembled by taking the majority label for each input image. We trained a separate binary classification network as described for each of the 9 attributes.

3.2.4 M3: Adding convolution layers on Inception 5b

In the next model[fig

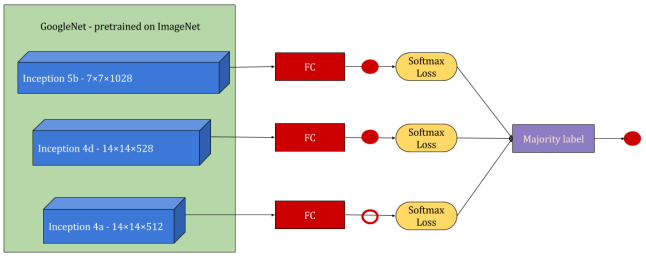

Figure 5. Ensemble on top of trained fc layers on top of each of the GoogleNet layers

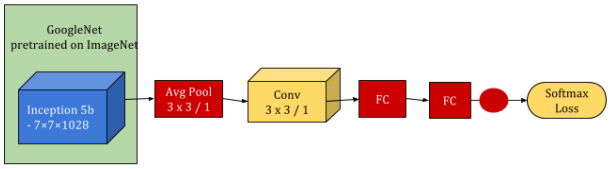

Figure 6. Training a net (pool-cnn-fc-fc) on top of Inception 5b output

3.2.5 M4: Training for all attributes and using mutliple GoogleNet layers as input

In our final model, we again use all three output layers from the GoogleNet. We add a Convolution layer on top of each of Inception 4a, Inception 4d and Inception 5b layers, Concat the outputs, and add a fully connected layer on top for each attribute with a softmax activation and loss. The convolution layers with input from Inception 4a and Inception 4d filters have 1024 filters of 2 x 2 dimension with stride 1, and the convolution layer with Inception 5b output as input had 1024 filters of 1 x 1 dimension with stride 1 for Inception 5b. Each of the layers resulted in an output of dimension: N x 1024 x 7 x 7 which are then concatenated to output vector of size: N x 3072 x 7 x 7. Each of the 9 fully connected layers take the output of the Concat layer and result in N x 2 x 1 x 1 output vector each to represent each attribute. This can be seen in figure 7. A softmax activation and loss layer sits on top of each of the fully connected layers and the 4 layer network is trained for 15 iterations. This network trains taking every attribute into account instead of a separate network for each attribute as all the other methods did.

3.2.6 Reasoning behind the network designs

We used the three different output layers of the GoogleNet to be able to use the different features captured at each level in our classifiers. Especially since the network wasn't trained entirely on this dataset, we wanted to experiment with whether higher level features exhibited earlier in the network would be better or supplemental to the finer grained

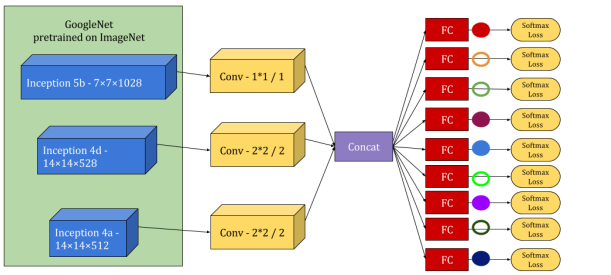

Figure 7. Training a net (pool-cnn-fc-fc) on top of Inception 5b output

features near the end of the network. By having two networks build upon just the output of the final Inception layer (Inception 5b) and two networks build upon the output of all 3 layers , we can get a sense of whether the higher level features that the network optimizes for in earlier levels help especially in these cases of transfer learning.

In addition to using the various output layers, we also have networks on top of the GoogleNet of various depths and layers to get a sense of if training just a single fully connected layer on top vs a Convolution layer with a fully connected layer makes a significant difference.

The final network trains not based on each attribute but based on the loss of all the fully connected layers combined. While, this is still single label classification as the earlier 3 networks, it considers all of the labels in training at a time rather than one at a time which would help us compare training separate binary classifiers vs having multiple fully connected layers each outputing a binary vector.

4. Dataset and Features

We are using the dataset provided by Yelp in one of the kaggle challenge [`https://www.kaggle.com/c/yelp-restaurant-photo-classification/data`]. The training set contains around 200K images corresponding to a set of 2000 businesses. Figure 8 shows the distribution of photos across all the business. It can be seen that the photos are very unevenly distributed across all the businesses. It also provides a set of attributes/labels that belong to each of the businesses. But, there is no direct mapping provided from photos to their corresponding labels. There are a total of 9 different attributes in this problem:

0: good for lunch
1: good for dinner
2: takes reservations
3: outdoor seating
4: restaurant is expensive
5: has alcohol

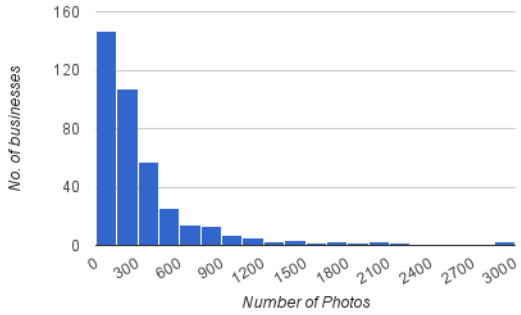

Figure 8. Distribution of photos across businesses

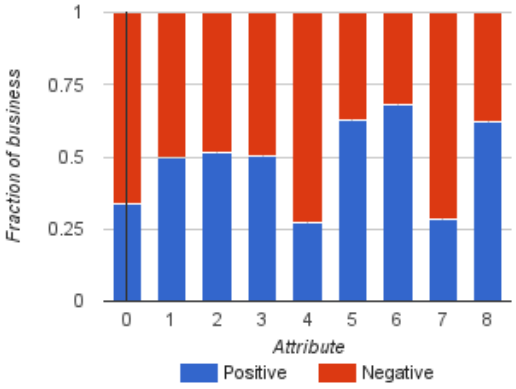

Figure 9. Distribution of Attributes across Businesses

6: has table service
7: ambience is classy
8: good for kids

These labels are annotated by the Yelp community. As you can see from figure 9 that labels are not very uniformly distributed across all the businesses which would make training a bit harder. In the training set, photos have dimension either of 350×500 or 500×350. We preprocessed the images by resizing them to a uniform size of 224×224.

To better deal with the unequal distribution of the photos and to alleviate computational resource constraints, we subsampled the data to include only 50 images from each business. Any business with less than 50 images was not used. This resulted in 1171 businesses with a combined 58500 images. We used 80-20 split for training and validation. With 934 businesses (46850 images) as the training set, and 234 businesses (11700 images) as the validation set.

For the purpose of our algorithms, we would treat the matrix representation of each of the images as an input to the pretrained Googlenet [10]. A forward pass over a part of this pretrained model, then gives us a set of features which we use to train our models. This is described previously in the Methods section.

5. Experiment and Results

5.1. Evaluation Metrics

We will be considering 2 metrics for measuring the performance of our method.

1. **Accuracy**: This is the fraction of data for which the algorithm correctly predicted if it can be tagged with the attribute or not. For photos, we would be analyzing the accuracy separately for each of the attributes over both training and validation set. For business we would be considering the mean accuracy, i.e., the fraction of labels for which we correctly predicted the outcome.

2. **F1-Score** also known as example-based F-measure. This scores measures accuracy using the statistics precision and recall. Precision is the ratio of true positives to all predicted positives. Recall is the ratio of true positives to all actual positives. Lets say the true positive is denoted by 'tp', false positive as 'fp', false negative as 'fn', precision as 'p' and recall as 'r'. The F1 score is given by:

$$F1 = \frac{2pr}{p+r} \text{ where } p = \frac{tp}{tp+fp}, r = \frac{tp}{tp+fn}$$

This metric gives equal weightage to both precision and recall and will try to maximize both precision and recall simultaneously. This would favor a moderately good performance on both over extremely good performance on one and poor performance on the other. Again for photos, we would evaluating this metric separately for each of the attributes to understand the performance of our classifiers and find insights about individual attributes. For businesses we would be calculating a mean F1-score as suggested on the Kaggle website. Suppose we have the following ground truth and predicted labels for a set of 4 businesses:
y_true = [[1, 2], [3, 4, 5], [6], [7]]
y_predicted = [[1, 2, 3, 9], [3, 4], [6, 12], [1]]
We calculate the the F1 score for each business. The "true positives" are defined as the common labels in ground truth and predicted label set for a given business. False positives are predicted items that aren't real, and false negatives are real items that aren't predicted. Then we average the F1-score over all businesses. For the above example, we will get a mean F1-score of 0.53

5.2. Experiments

For each of the methods we used a mini-batch gradient descent approach using momentum update to learn the network parameters. We also clipped the gradients to make sure that loss doesnt go out of bounds. We used a batch size of 50. From various papers and discussion in lectures, we have observed that batch sizes of 32 or 64 are the most common choice. We used 50 instead of 64 or 32 as this was giving us a trade-off between the memory usage and run time of the update algorithm. For method 1, we started with a low learning rate of 0.05 but it was not sufficient to make the loss converge to 0 even for a small dataset of 10 images. Then, we increased the learning rate of 0.2 and added a momentum of 0.8 which made convergence much faster. The results showed that this method is not performing well so we didn't perform any further experiments on this.

For Method 3 and Method 4 we started with the learning rate from method 1 and varied it ranging from 0.1 to 0.8. We also varied the momentum from 0.7 to 0.9. We observed that extreme hyperparameters lead to a drop in the performance, though it was not very significant. In the next section we would be presenting only the best result corresponding to a learning rate of 0.3 and momentum of 0.8. We also varied the value at which we are clipping the gradients and found good results for 0.25.

For method 2, we used the learning rates from previous methods and didn't do any further experimentation due to significant amount of training time for just 1 set of hyperparameters.

All the experiments were trained on GeForce GTX 680 GPUs. Methods 1, 3 and 4 each took around 8-10 hours to run once on a training set of 47k images. Method 2 involved 9 different CNNs took and hence took upward of 20 hours for 1 complete run of 30 iterations over the ehole dataset.

5.3. Results

5.3.1 Image classification

Let us first examine the classification accuracy of our images.

Figures 10 and 11 shows the performance of different methods on the validation set described in Section 3 for each of the 9 attributes. Upon analyzing the accuracies and F1 scores based on attributes, we can see that some attributes are easier to classify than others. Attributes 5 and 6 have high accuracy and F1 scores followed by 1, 2, 8 which are has alcohol, has table service, good for dinner, takes reservations and good for kids respectively, while attributes 0,4 and 7 had high accuracies and low F1 scores which are attributes: good for lunch and restaurant is expensive respectively. Upon further investigation of why the F1 score is low, we saw that the recall was low for these attributes which suggest that our methods are predicting much

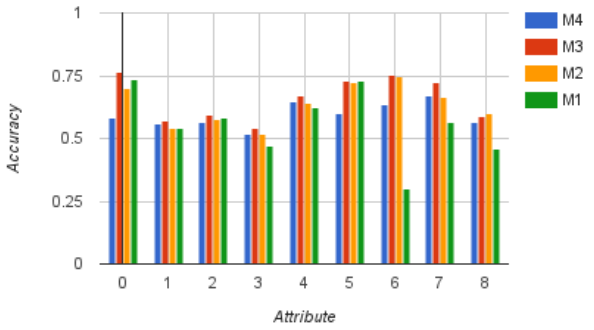

Figure 10. Validation Accuracy per attribute over images

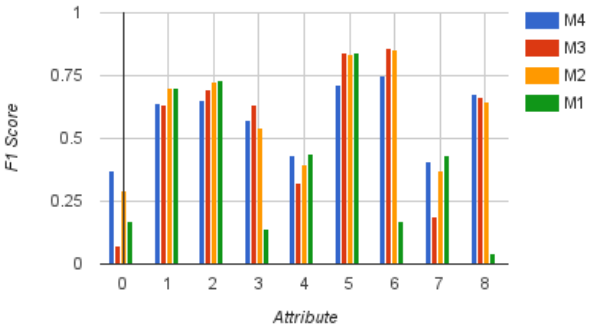

Figure 11. Validation F1 score per attribute over images

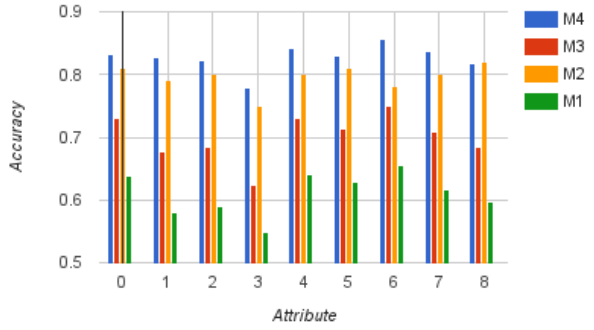

Figure 12. Training Accuracy per attribute over images

less than what it should be predicting. This can be due to relatively lower fraction of positive examples for these attributes in the dataset as shown in figure 9.

While most of the methods did similar to other methods in each attribute, Method 1 was significantly lower than the rest of the methods for several of the attributes which seems to indicate that a single fully connected layer using only the end output isn't sufficient to capture as many features of the

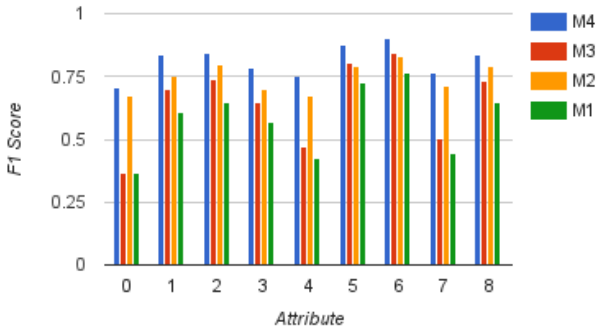

Figure 13. Training F1 score per attribute over images

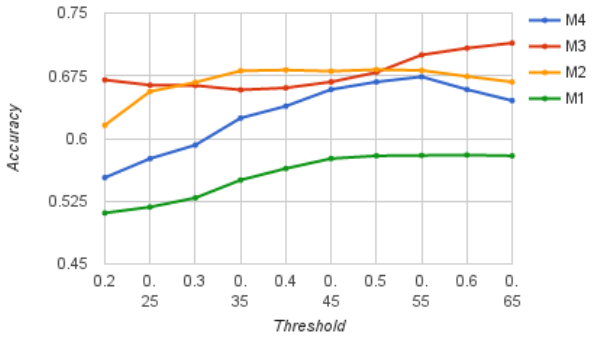

Figure 14. Overall training accuracy for business label classification over various thresholds

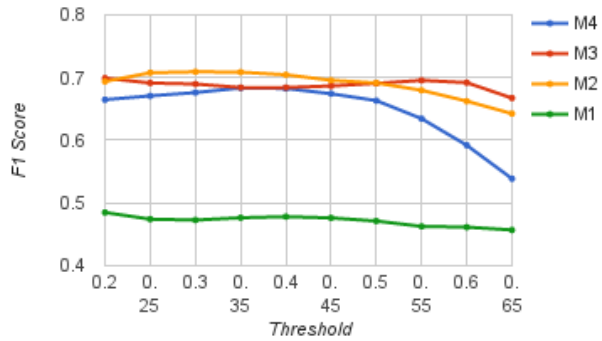

Figure 15. Overall Validation F1 Score for business label classification over various thresholds

image as the other methods.

5.3.2 Business classification

Figures 14 and 15 shows the overall performance of all the 4 methods for the validation set. From these graphs, we can

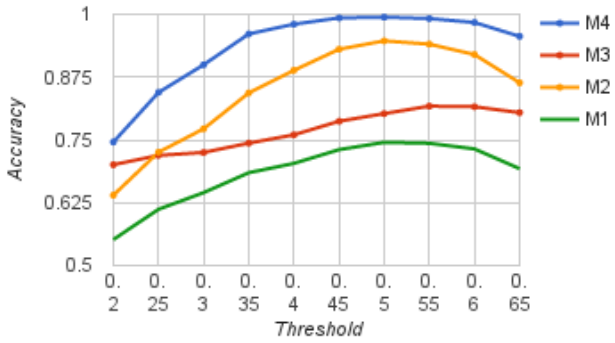

Figure 16. Overall Validation accuracy for business label classification over various thresholds

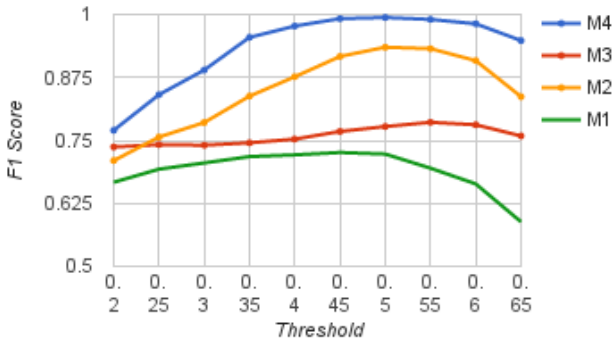

Figure 17. Overall Training accuracy for business label classification over various thresholds

Table 1. Performance of Different Methods over Validation Set

Method	Validation		Training	
	Max F1 Score	Max Accuracy	Max F1 Score	Max Accuracy
M1	0.48	0.58	0.73	0.75
M2	**0.71**	0.68	0.93	0.95
M3	**0.7**	**0.7**	0.79	0.81
M4	0.68	0.67	0.99	0.99

compare how the methods do overall in classifying businesses. Method 1 does significantly worse than the others at every threshold in both accuracy and F1 score which is in accordance to our previous analysis and it shows that we need better and bit more complex networks to correctly model this classification problem. Another interesting observation is that even the method 4 which has F1 scores in a similar range to the other two methods for image label prediction, is consistently performing worse than Method 2 and Method 3 in both accuracy and F1 score. However, Method 4 has the highest training accuracy and mean F1 score (approx 1) which is also much more as compared to its validation accuracy. This shows that Method 4 is overfitting on the training set and indicates that using a better regularization such as a dropout layer would have helped in

enhancing its performance on the validation set.

The validation of both Method 2 and 4 that use features from the 3 different layers in the pretrained net is much lower than the training which implies that pulling features from multiple layers is causing overfitting which intuitively makes sense. If we compare the training accuracies of these two methods against a similar network using only the final layer features, it seems pulling features from multiple layers of the net instead of just the final layer significantly improves training performance based on evidence Method 4 having around 20%+ lead over Method 3 and Method 2 having 20% lead over Method 1. This calls for better regularization methods while using a more complex network/higher-dimensional feature space.

6. Conclusion and Future Work

We took a complex multi-instance, multi-label classification problem and simplified it with reasoning through examining the dataset into a single label, single instance classification. Then to save computation resources and reduce the bias among businesses, we subsampled the dataset by choosing 50 random images per business and used only around a quarter of the original training set. We then used transfer learning to use the learnings from a network previously proven to work very well on a different dataset [10]. With these simplifications and methods, we achieved around 73% validation accuracy and a F1 score of 0.71.

We saw from our results that adding a little complexity even as simple as training a convolutional layer and a fully connected layer instead of just a fully connected layer on top of the pretrained network significantly improves performance. Additionally, pulling features from multiple layers through a pretrained network also seems to help significantly, although we need to employ proper regularization techniques to avoid overfitting.

We used many simplifying techniques to solve a complex problem and performed with decent results. If we want to achieve much better accuracy, there are many easy avenues of improvement. Using more data to train, training for more iterations are very simple ways of improving. More importantly, tackling the problem without some of the simplifications might improve the overall accuracy. Comparing how the same network architectures do with multi-label classification instead of several binary classifiers would be interesting. Additionally, training the entire GoogleNet or at least a larger chunk of the network instead of freezing it with weights trained for another dataset would be a definite next step.

For this image classification problem, there are many ways to use CNNs to solve the problem some of which include complex networks to try to improve a particular aspect of the problem such as multiple instance or multi-label but even starting with an established pretrained network, using transfer learning and training small networks on top are great ways to make headway at the problem and provide a good foundation to add more specific techniques on top of to incrementally increase accuracy hereafter.

References

[1] How we use deep learning to classify business photos.

[2] K. Balasubramanian and G. Lebanon. The landmark selection method for multiple output prediction. *arXiv preprint arXiv:1206.6479*, 2012.

[3] W. Bi and J. Kwok. Efficient multi-label classification with many labels. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 405–413, 2013.

[4] R. S. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino. Matrix completion for multi-label image classification. In *NIPS*, volume 201, page 2, 2011.

[5] D. C. Cireşan, U. Meier, and J. Schmidhuber. Transfer learning for latin and chinese characters with deep neural networks. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6. IEEE, 2012.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[8] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724, 2014.

[9] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. *arXiv preprint arXiv:1412.7144*, 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. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[11] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. Cnn: Single-label to multi-label. *arXiv preprint arXiv:1406.5726*, 2014.

[12] J. Wu, Y. Yinan, C. Huang, and Y. Kai. Deep multiple instance learning for image classification and auto-annotation. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3460–3469. IEEE, 2015.

[13] Z.-H. Z. M.-L. Zhang. Multi-instance multi-label learning with application to scene classification.