Ambience is classy, or good for kids? Yelp Restaurant Photo Classification

Jee Ian Tam
Stanford University
Stanford, CA 94305
jeetam@stanford.edu

Abstract

We apply a convolutional neural network to solve the multi-instance problem of assigning labels to a business given one or more images of the business. We pose the problem as a multi-label classification problem by assigning labels to training images from their corresponding businesses. The evaluation metric used is the F1 score. We fine-tune up to 80% the VGG-16 network, and evaluate the binary relevance, label powerset and BP-MLL transformations used to deal with the multi-label classification problem. We find that the BP-MLL algorithm transformation gives the highest F1 score of 0.75 over the test set. We also find that the network is able to distinguish between the two main label clusters (dinner and lunch) of the training and test sets.

1. Introduction

Yelp is company that publishes crowd-sourced reviews about local businesses, especially restaurants. A key part of the Yelp platform is the ability of users to upload photos of a restaurant's interior and food as part of their dining experience. Furthermore, restaurants can also provide information about themselves via labels such as ”good for lunch”, ”outdoor seating” and ”restaurant is classy”. The photos that are uploaded by users provide rich business information about these categories. Given images of a business, Yelp would like to be able to predict labels of that business. Yelp has thus released a dataset towards that purpose on Kaggle, a platform where people compete in machine learning challenges. I participated in a multi-label classification challenge hosted by Yelp in this context, where the challenge is to predict business labels solely from the photos that have been uploaded by users.

The input to our system are images of shape 224x224x3, and the output is a set of business labels. The training data set consists of a list of businesses, their labels, and a corresponding set of images for each business. The test set consists of a list of business with a corresponding set of images for each business. Our goal is to predict the labels of the businesses in the test set. We approached the problem as one of classifying the labels of images, and then aggregated label predictions across constitutent images to predict the labels of the corresponding business. We fine-tuned a pre-trained convolutional network to predict the labels of images, and explored different kinds of transformation methods used to deal with the multi-label classification problem.

2. Related Work

The problem is a multi-instance learning (MIL) problem, where each labeled business is associated with multiple unlabeled instances (images), and we are trying to predict the business labels from those unlabeled images. Keeler et. al [1], Dietterich [2] and Maron, Lozana-Perez [3] first explored methods for dealing with MIL problems. They elaborate on the standard MI assumption, which takes each instance (images in our case) to have an associated label which is hidden. For the training step in our application, we assume that the set of labels for images is equal to the set of labels for businesses, propagate the labels of each business to its constituent images, and focus on predicting image labels from the images themselves. For the testing step of our application, after computing label predictions we use a simple metadata-based algorithm to infer the business labels as follows : We use a SimpleMI algorithm [4], where the metadata of the business is the mean statistic over the image labels. That is, we compute the mean label for a business by taking the mean of labels of its constituent images, and compute a business label from the mean label by thresholding over a pre-determined value.

The multi-instance learning approach detailed above transforms the problem into one of multi-label classification [6], where we aim to predict one or more labels for each instance (image). There have been multiple methods explored in the space of multi-label classification [7]. The method of binary relevance [5] trains one independent binary classifier per label. This method is straightforward and easy to implement, the disadvantage of it is that it does not consider correlations between labels. Another transformation is the label powerset transformation [8], where we create a binary classifier for each possible label combination. While this method is able to capture correlations between labels, the complexity of the transformation is 2^N where N is the number of labels. Thus, a relatively large number of binary classifiers have to be trained, meaning that this method is high-dimensional and requires more data than other methods. Furthermore, in our case there are only a few meaningful label clusters, and so this method is additionally wasteful. Another powerful method for dealing with multi-label classification problems is BP-MLL [9] [10], which is an adaptation of backpropagation to multi-label learning. Considering that we will be using convolutional networks as classifiers, this is a method that is suited for our application. We will elaborate more on these methods in section 4.

We use a pre-trained VGG-16 [11] convolutional network model as a base model and fine-tune it to our dataset. The architecture of convolutional networks was taken from Krizhevsky[12], and the main idea of fine-tuning pre-trained networks on new datasets can be found in [13]. Previous work of applying computer vision methods for multi-label classification can be found in Boutell, Lou and Brown [14], where they used various image features as input to multiple binary classifier SVMs for prediction of labels. Nam [15] investigated the use of neural networks with BP-MLL in the context of multi-label text classification, where they found that methods such as dropout and ReLu units were effective when used with BP-MLL. Other methods for multi-label learning in the context of computer vision can be found in work by Dimou and Tsoumakas [16], where methods such as BP-MLL, MLkNN and RAkEL were explored.

3. Dataset and Features

The data set is provided by Yelp via a Kaggle competition [17], and it consists of a map of business ids to labels (only for the training dataset), a map of business ids to photos, and the photos themselves. The photos vary in resolution, the median resolution is 375 x 500. They consist of pictures taken at various business establishments. Yelp notes that there are duplicate photos in the training data set due to users uploading the same photo more than once, we choose to not remove these duplicates as they do not impact the training process, and the duplicates consist of a very small fraction ($\sim 3\%$) of the total dataset. The training dataset consists of roughly 2,000 businesses and 230,000 photos, whereas the test dataset contains roughly 10,000 businesses and roughly 240,000 photos.

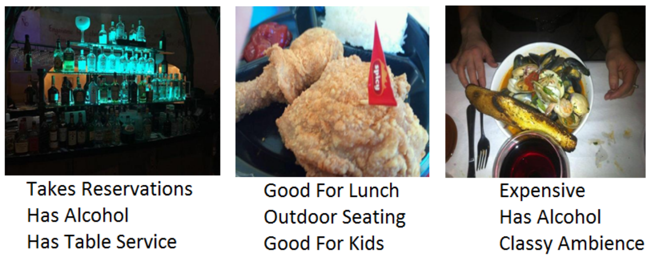


Figure 1: Sample pictures and labels from training data set

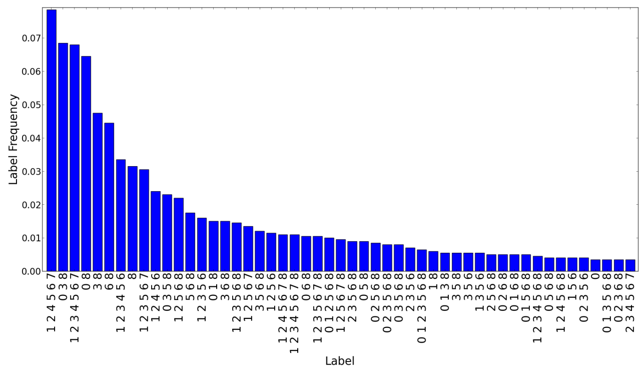


Figure 2: Top 50 most frequent label sets in training dataset

Due to computational constraints we only train on a subset (100,000 images) of the total training set, but we predict labels for all images in the test set. A validation set of 1,000 images is used at every epoch to compute validation loss. The validation set changes across training batches, as the validation set is taken from a single-fold 10% of the current training batch. Sample images can be found in figure 1

There are 9 possible labels for each business, and exploration of the labels in the dataset shows that the labels are not evenly distributed. The frequency and name of each individual label in the training dataset is presented in table 1:

ID	Label Name	Relative Frequency
0	Good for lunch	0.336
1	Good for dinner	0.497
2	Takes reservations	0.513
3	Outdoor seating	0.502
4	Restaurant is expensive	0.274
5	Has alcohol	0.625
6	Has table service	0.680
7	Ambience is classy	0.286
8	Good for kids	0.619

Table 1: Label Names and Frequencies

Furthermore, a bar graph showing the top 50 most frequent business label sets in the training dataset is shown in figure 2

We see that the top 2 most frequent label sets are [Good for dinner, Takes reservations, Expensive, Alcohol, Table Service, Classy] and [Good for lunch, Outdoor Seating, Good for kids]. Furthermore, we note that the most frequent label combinations after those are generally combinations of those two main label clusters. Thus, we observe that there are two main label clusters in the data set, the former corresponding to restaurants associated with dinner, and the latter generally corresponding to lunch or kid-friendly restaurants. We denote the former as the DINNER label set, and the latter as LUNCH label set.

As we use a pre-trained VGG-16 network, we subtract from each image the mean pixel BGR values [11] of [103.939, 116.779, 123.68] used to train the original network. We downsample the input image dimensions to 224 x 224 via bilinear interpolation (each image is maintained as a color image, so each image input has size 224x224x3). We perform real-time data augmentation [18] before passing it into the network. The augmentations are random operations as presented below. Each operation has a 50% chance of being applied to the input image.

- Value (brightness) change of +/- 5%
- Horizontal shift of 5 pixels to the left or right
- Rotation of +/- 5 degrees
- Horizontal Flip
- Crop of 90% of image

We do not extract features from the images to feed into the network, but instead feed in raw image pixels into the network (after performing the preprocesing steps noted above). This is because we aim to progressively fine-tune the VGG-16 network, and the original network took in raw pixel values as well. Thus, in order to make sure that the activations do not deviate too much when given the same image when we fine-tune the network, we have to feed in raw input pixels as well.

4. Methods

4.1. Multi-Label Classification Methods

In this section we elaborate on the BP-MLL algorithm, which we used to achieve the best model performance compared to binary relevance and label powerset.

Recall that binary relevance trains a binary classifier for each label. In the context of fine-tuning VGG-16, this is equivalent to replacing the last layer with N output neurons (N being the number of labels), then placing a sigmoid layer followed by a binary cross-entropy loss across each neuron. This method is simple, but ignores label correlations.

Label powerset, on the other hand, trains a binary classifier for each possible label combination. In our context, this is equivalent to replacing the last layer with 2^N neurons, then placing a softmax layer follows by a categorical cross-entropy loss across all neurons, similar to what would be done for a multi-class classification problem, as done in [12]. This method can capture label correlations, but is exponential in the number of labels and thus can be harder to train.

BP-MLL is an algorithm that was originally explored by Zhang[9] with applications to functional genomics and text categorization. It introduces a loss that takes into account label correlations, and is an adaptation of backpropagation for multi-label classification. Backpropagation[19][20] is the process of propagating gradients from the loss back towards the input layer of a neural network in order to compute gradients for every weight in the network. Let Y represent the set of all possible labels. We replace the last layer of VGG-16 with N output neurons, and the BP-MLL loss for a batch is given as :

$$E = \sum_{i=1}^{m} E_i = \sum_{i=1}^{m} \frac{1}{|Y_i||\overline{Y}_i|} \sum_{(k,l) \in Y_i \times \overline{Y}_i} exp(-(c_k^i - c_l^i)) \tag{1}$$

where m is the number of batch training examples, E_i is the loss for the i-th training example, $Y_i \subseteq Y$ is the set of labels of the i-th training example (in-set labels), $\overline{Y}_i$ is the complement of Y_i ($\overline{Y}_i = Y \backslash Y_i$, out-of-set labels), and c_p^q is the activation of the p-th neuron in the final layer for the q-th training example.

From equation 1, we see that for a particular training example i, minimizing E_i corresponds to maximizing the difference between all activations corresponding to in-set labels and all activations corresponding to out-of-set labels, as BP-MLL enumerates over the cross-product pairs of in-set and out-of-set labels. Thus, all activations corresponding to in-set labels will be simultaneously optimized to be higher, while activations for out-of-set labels will be simultaneously optimized to be lower. Label correlations (both in-set and out-of-set) are thus able to be captured by minimizing the BP-MLL loss as presented above. The gradients corresponding to BP-MLL loss can be found in [9], we write our implementation in Theano[21] which calculates gradients symbolically.

In order to predict binary labels for an input image from the network output activations after optimizing the network on BP-MLL loss, a threshold has to be either specified or learnt. We use a variant of BP-MLL as presented by Grodzicki[10] that allows for simultaneous threshold learning while training the convolutional network. In our context, this method involves replacing the last layer of VGG-16 with $2N$ output neurons, where for $0 \leq p < N$, the 2p-th output neuron corresponds to the activation for the p-th label, and the (2p+1)-th output neuron corresponds to the threshold for the p-th label. We ”squash” the output neuron activations using a tanh activation for each neuron. Thus, the final form of loss for a single training example is given by :

$$E_i = K\Big[\sum_{(k,l)\in Y_i \times \overline{Y}_i} e^{-(c^i_{2k} - c^i_{2l})} + e^{-(c^i_{2l+1} - c^i_{2k+1})} + \sum_{r\in Y_i}\sum_{t\in Y_i} e^{-(c^i_{2r} - c^i_{2t+1})} + \sum_{s\in \overline{Y}_i}\sum_{t\in \overline{Y}_i} e^{-(c^i_{2t+1} - c^i_{2s})}\Big] \quad (2)$$

where $K = \dfrac{1}{2|Y_i||\overline{Y}_i| + |Y_i|^2 + |\overline{Y}_i|^2}$ is a normalizing factor. Minimizing the above loss not only minimizes the differences between in-set label activations and out-of-set label activations as per the original BP-MLL loss, it also optimizes the thresholds such that the thresholds are as low as possible from the in-set label activations (2nd summation term), and as high as possible from the out-of-set label activations (3rd summation term). Furthermore, minimizing the loss also results in lower threshold values corresponding to labels in Y_i compared to those of labels in $\overline{Y}_i$ (2nd exponential term in 1st summation term). Thus, by minimizing the above loss, individual thresholds for all labels are able to be learnt autonomously during optimization of the network.

4.2. Convolutional Network Architecture

We use a pre-trained VGG-16 network as a starting network, and replace the final layer of the network with a number of neurons that depends on which kind of loss that we are testing, as detailed in section 4.1. The network architecture is shown in figure 3 :

4.3. Multi-Instance Learning

In order to predict the label of a business given multiple images of it, we compute label predictions for each of its images, then calculate a mean label vector for the business from its constituent images. We then compute the business label via binarizing the mean label vector by thresholding at 0.5. We experimented with different threshold schemes such as using the prior label frequencies presented in table 2, but we find that a simple threshold at 0.5 gave the best results.

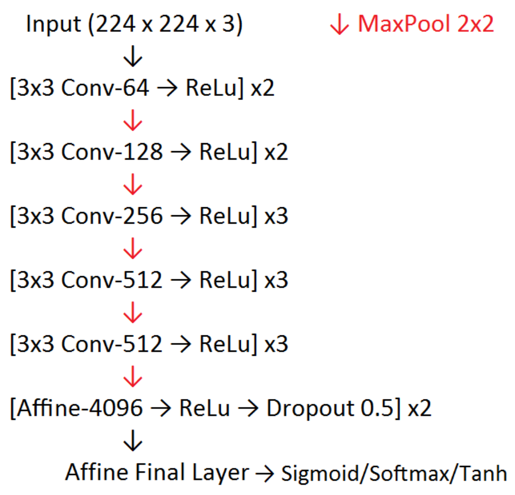


Figure 3: Network architecture used (VGG-16)

5. Experiments / Results / Discussion

5.1. Training Details

The network was implemented in Keras [22], with the pre-trained VGG-16 network weights obtained by converting the Caffe model [23]. The network was trained on an NVIDIA GRID K520 GPU.

We experiment with the binary relevance, label powerset and BP-MLL methods. We train on a set of 100,000 training samples. Due to memory constraints, we train on batch sizes of 10,000. We use a mini-batch size of 20, and validate on a single fold of 1,000 validation samples, taken from 10% of the training batch. The mini-batch size was chosen based on the amount of GPU memory available.

For each method, we fine-tune the original VGG-16 network by progressively training layers starting from the final layer up towards the input layer. The fine-tuning details are shown below.

Fine-Tuned Layers	**Epochs Trained**	**Initial Learning rate**
Last Layer	2-3	8.7e-4
Top 16 ($\sim$ 40% of net)	4-5	1.1e-4
Top 32 ($\sim$ 80% of net)	5-6	1.5e-5

Table 2: Fine-Tuning details

The number of training epochs was chosen based on time constraints, and we aim to spend more training epochs on stages where we fine-tune more layers of the network. We use mini-batch Gradient Descent with momentum and learning rate decay, along with L2 weight regularization to optimize the convolutional networks. For each fine-tuning stage, the learning rate was chosen after a hyperparameter search of 10 samples over a range of 1e2 using a small validation set of 1,000 samples (single-fold). For the final layer, the learning rate hyperparameter search range was based around $\frac{1}{10}$ of the original VGG-16 initial learning rate (0.01) and for the layers in the middle of the network, the search range was based around $\frac{1}{100}$ of the original initial learning rate. For the L2 regularization parameter, we use the same value used for training the original VGG-16 network (5e-4). We use a learning rate decay of 0.98 and momentum of 0.5. The learning rate is divided by 10 when the validation loss plateaus.

We also used with RMSProp[24] with similar learning rates for initial training of the final layer of the network.

5.2. Evaluation Metrics

The metric that the Kaggle submissions are judged on relative to the test set is the F1 score. The F1 score is defined as

$$F1 = \frac{2pr}{p + r} \tag{3}$$

where the precision p is given as $p = \dfrac{tp}{tp + fp}$ and the recall r is given as $r = \dfrac{tp}{tp + fn}$, where tp is the number of true positives, fp is the number of false positives, and fn is the number of false negatives across the test set. The F1 metric rewards moderately good performance on both precision and recall over extremely good performance on one at the expense of the other.

5.3. Results / Discussion of different models

We show the test-set overall F1 score for the various multi-label classification methods described in section 4 . Note that we do not have access to the true labels of the test set due to the Kaggle competition - We can only upload predictions on the test set and receive a computation of the F1 score from the competition server:

Method	F1 Score
Binary Relevance	0.67
Label Powerset	0.62
BP-MLL	0.75
Yelp Benchmark	0.64
Current leaderboard top score	0.83

Table 3: Test set F1 scores for various MLL methods

We see that BP-MLL gives the best performance across the test set, followed by binary relevance and lastly label-powerset. BP-MLL gives the best performance because it is able to account for label correlations with complexity linear in the number of output neurons. We see the binary relevance is able to meet the benchmark, whilst label powerset performed the worst. We hypothesize that this is due to the larger number of output neurons needed for label powerset, making the method more prone to overfitting the training dataset compared to the other two techniques.

5.4. Performance of model using BP-MLL loss

We now focus on the performance of the best model that uses BP-MLL loss. Precision, recall and F1-scores for each label on a validation set of 10,00 samples are shown in table 4 :

Label ID	Precision	Recall	F1
0	0.65	0.27	0.38
1	0.79	0.83	0.81
2	0.82	0.88	0.85
3	0.54	0.58	0.56
4	0.73	0.66	0.69
5	0.84	0.92	0.88
6	0.86	0.96	0.90
7	0.71	0.54	0.61
8	0.78	0.75	0.76
Average	0.77	0.77	0.76

Table 4: Precision, Recall and F1 for BP-MLL model

We plot the distribution of label predictions over the test set in figure 4. The frequencies of predicted test labels are also reported in Table 5.

ID	Label Name	Relative Frequency
0	Good for lunch	0.0113
1	Good for dinner	0.7357
2	Takes reservations	0.8516
3	Outdoor seating	0.2830
4	Restaurant is expensive	0.0572
5	Has alcohol	0.9330
6	Has table service	0.9653
7	Ambience is classy	0.0260
8	Good for kids	0.7962

Table 5: Test Label Frequencies

We see that label ID 0 ("Good for lunch") performs the worst in terms of Recall and F1 Score. Thus, the multi-label classifier gives a particularly large number of false negatives for label 0. This can also be seen from the test label frequencies, where label 0 has a particularly low prediction frequency. It isn't clear why the performance and frequency of predictions for label 0 are particularly low. Furthermore, we also see that the range of possible labels predicted for the test set is not as diverse as is present in the training set , as seen by comparing figure 4 and

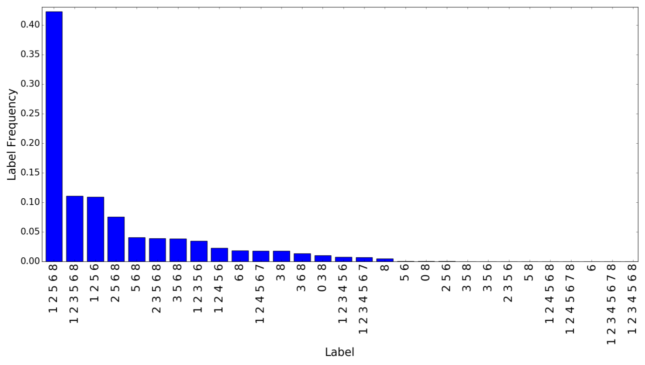


Figure 4: Most frequent label sets (predicted) in test dataset

LUNCH cluster : Good for lunch & Outdoor Seating& Good for kids

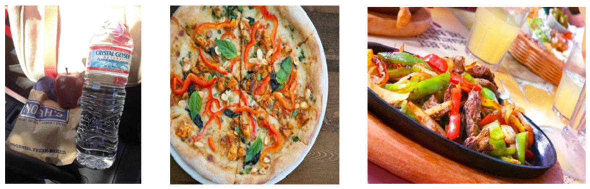


Figure 5: Predicted test set images in LUNCH cluster

DINNER cluster : Is expensive & Classy ambience & Has alcohol

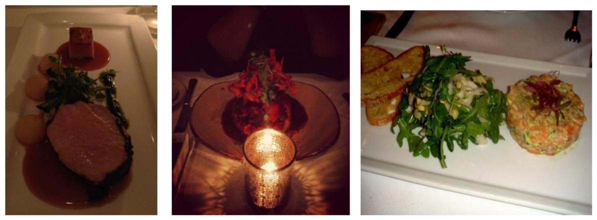
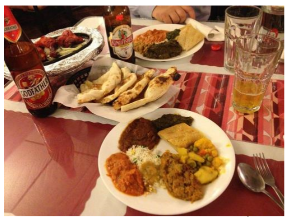
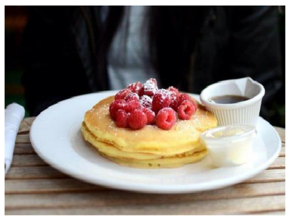


Figure 6: Predicted test set pictures in DINNER cluster



True Labels	Predicted Labels
Good for lunch	Outdoor seating
Good for dinner	Takes reservations
Has table service	Has alcohol
Good for kids	



True Labels	Predicted Labels
Good for dinner	Good for lunch
Takes reservations	Outdoor seating
Restaurant is expensive	Good for kids
Has alcohol	
Has table service	
Ambience is classy	

Figure 7: Examples of incorrectly-predicted labels

figure 2. The label set [1,2,5,6,8] is predicted roughly 40% of the time, which explains the high frequency of predictions in some labels and low frequency of predictions in other labels as seen in table 5. It could be that the network has learnt to predict a label set that is some combination of the DINNER and LUNCH clusters most of time.

Earlier in section 3 we identified two main label clusters corresponding to DINNER and LUNCH. We present a few samples of test set images identified by the model belonging to those label clusters in figures 5 and 6. We see that the network is able to label images that generally seem to belong to labels in the LUNCH cluster or the DINNER cluster. More results can be seen in figure 10

There are some cases where the model predicts entirely incorrect labels. A few cases from the training set are shown in figure 7. We see that the model makes reasonable errors. Taking the example of the image on the right of figure 7, where what might be construed as ”good for lunch” due to the presence of pancakes in the image might actually be attributed to a more fancy restaurant that is actually in the DINNER cluster. Similarly, looking at the image on the left of figure 7, the network predicts the label ”has alcohol” possibly based on the presence of beer bottles in the image, even though the establishment does not label itself as such, possibly because the beer bottles were not purchased at the restaurant. Looking at the failure cases of the network, we hypothesize that the network assigns labels by focusing on objects / textures that are detected locally, and does not take into account the other image attributes that might give an indication of the business labels such as picture quality, cutlery/silverware arrangement and general ambience cues of the restaurant.

In order to better visualize the performance of the classifier in being able to assign and distinguish between the two main label clusters, we run the t-SNE[25] algorithm using sklearn[26] on the 4096-dimensional features extracted from the second-to-last-layer of the modified VGG-16 network. We extract features for roughly 15,000 training samples. The t-SNE plots for the DINNER label cluster of [1,2,4,5,6,7] and the LUNCH label cluster of [0,3,8] is shown in figure 8.

We see that network has learned an embedding on the data manifold that is able to distinguish a subset of the DINNER label cluster from data samples in the LUNCH cluster, as can be seen in the lower-left, upper-right and edges of the plot in figure 8. However, there are also other training cases on the data manifold for which the network was not able to achieve a clear separation between datapoints in the two clusters, as can be seen in the middle portion of the same figure. We also present t-SNE plots for all of the individual labels in figure 9

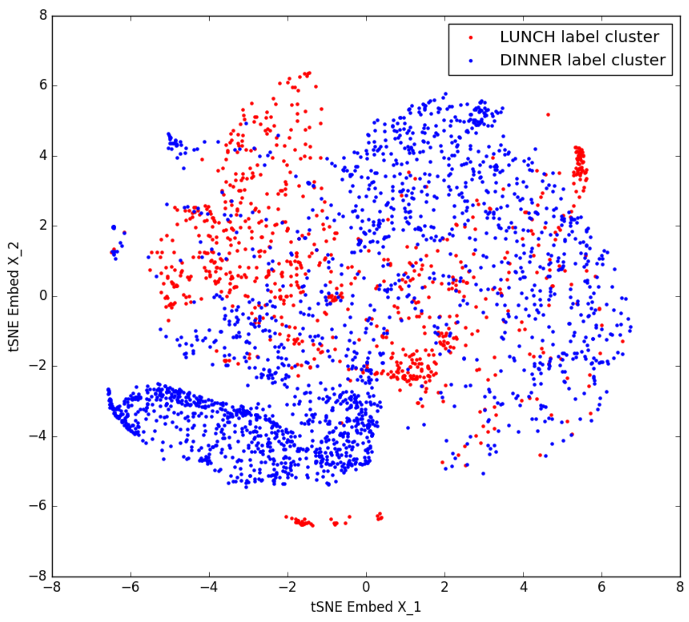


Figure 8: t-SNE embedding of training samples for LUNCH and DINNER label cluster

It is unlikely that the model trained using BP-MLL loss overfit the training data set as the training process was closely monitored such that the training loss and the validation loss across all batches did not diverge too much. The final training loss for the model was 0.65, whereas the validation loss was 0.68, a difference of 5%. Thus, the validation loss being close to the training loss indicates that the model has not overfit the training data set. We also see this in the similarity of the F1 scores on the test set and the validation set in Tables 3 and 4.

6. Conclusion / Future Work

In conclusion, we find that BP-MLL is able to provide a boost in network performance (in terms of F1 score) over binary relevance and label powerset methods. We obtain an F1 score of 0.75 on the competition test set, and we find that the trained network with BP-MLL loss is able to discern high-level clusters such as label clusters corresponding to dinner and lunch.

Future work includes exploring using BP-MLL loss with fine-tuning of a more recent network architectures such as ResNet[28] or GoogleNet[29]. There was some difficulty in getting the weights from GoogleNet and ResNet to be imported into Keras, and as such future explorations would probably be easier to do in the Caffe[23] framework. In addition to that, other methods for multi-label learning such as binary relevance classifier chains [27] would also be of interest.

In section 5.4 we hypothesized that the network assigns labels by focusing more on locally-detected objects and textures rather than global image features. Given more time, we could use the model to produce saliency maps to check if the model is actually attending to such local objects and textures, as was done in [31]. It would be interesting to train from scratch a classifier that takes in features regarding picture quality[30] and combine with the fine-tuned model that we have developed here into an ensemble to see how much improvement can be obtained.

Lastly, it would be also be interesting to explore methods to generate images based on the network that we have trained on the Yelp dataset - For example, we could try to deep-dream[32] images based on the later layers of the network after fine-tuning those layers to find out if the network actually ”dreams” of food-related objects.

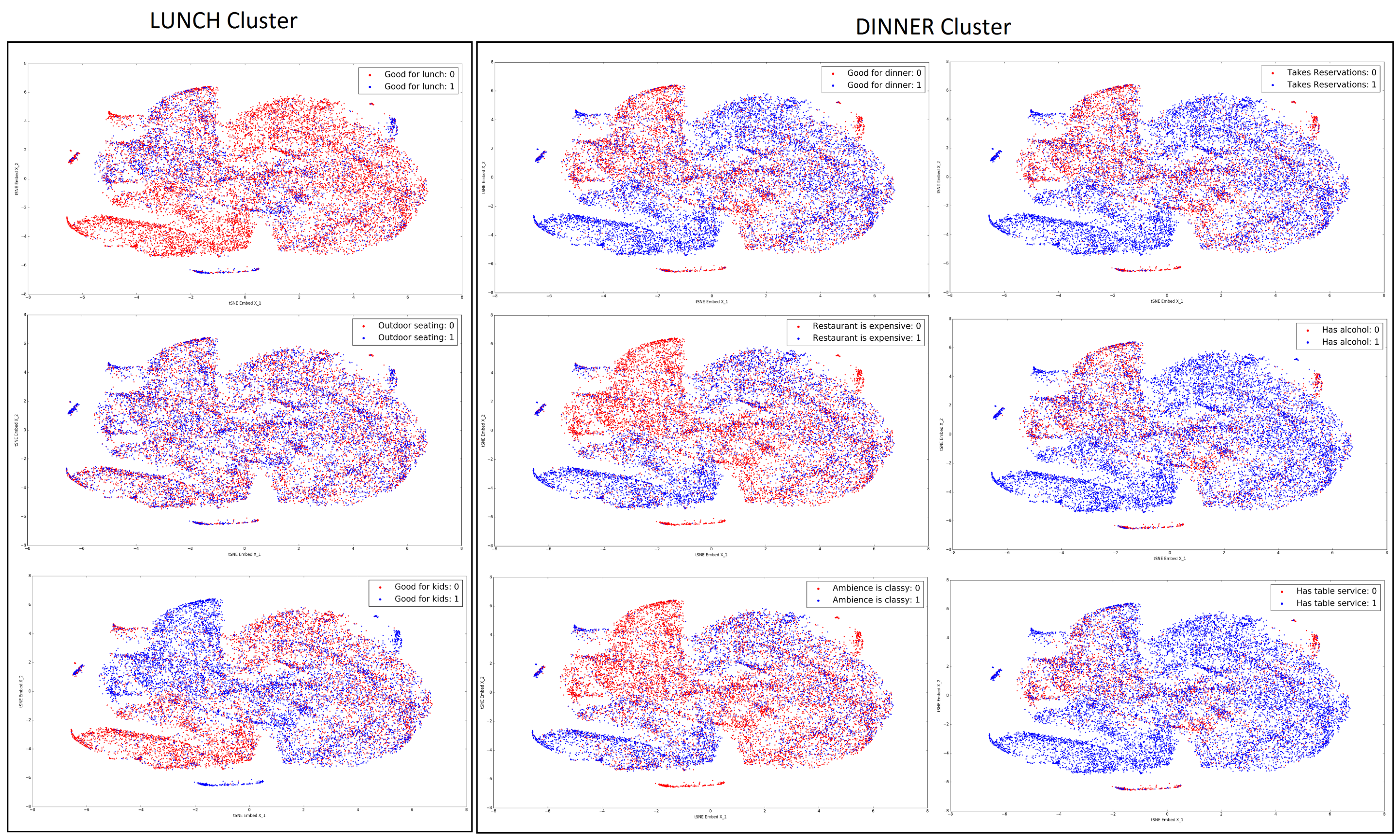


Figure 9: t-SNE embedding of training samples for individual labels

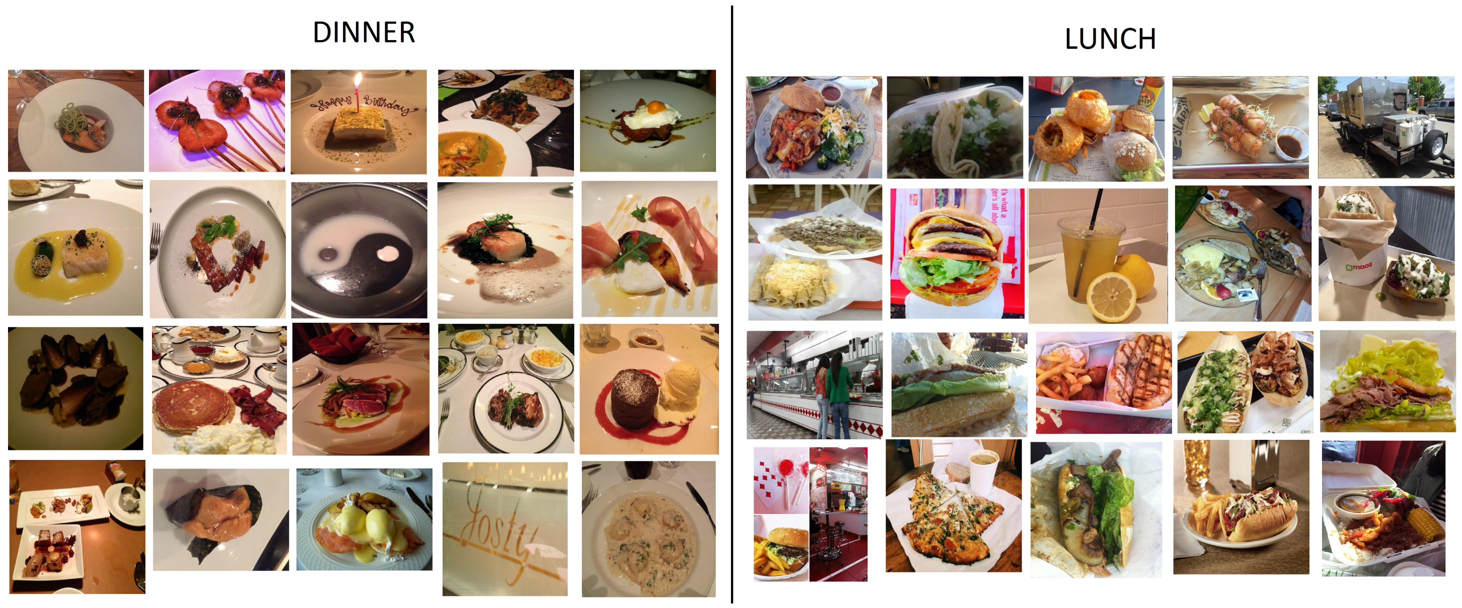


Figure 10: Random samples of test set images predicted to be in DINNER label set [1,2,4,5,6,7] or in the LUNCH set [0,3,8]

References

[1] Keeler, James D., David E. Rumelhart, and Wee-Kheng Leow. Integrated Segmentation and Recognition of Hand-Printed Numerals. Microelectronics and Computer Technology Corporation, 1991.

[2] Dietterich, Thomas G., Richard H. Lathrop, and Toms Lozano-Prez. ”Solving the multiple instance problem with axis-parallel rectangles.” Artificial intelligence 89.1 (1997): 31-71.

[3] Maron, Oded, and Toms Lozano-Prez. ”A framework for multiple-instance learning.” Advances in neural information processing systems (1998): 570-576.

[4] Dong, Lin. A comparison of multi-instance learning algorithms. Diss. The University of Waikato, 2006.

[5] Tsoumakas, Grigorios, Ioannis Katakis, and Ioannis Vlahavas. ”Mining multi-label data.” Data mining and knowledge discovery handbook. Springer US, 2009. 667-685.

[6] Tsoumakas, Grigorios; Katakis, Ioannis (2007). ”Multi-label classification: an overview” (PDF). International Journal of Data Warehousing & Mining 3 (3): 113. doi:10.4018/jdwm.2007070101

[7] Zhang, Min-Ling, and Zhi-Hua Zhou. ”A review on multi-label learning algorithms.” Knowledge and Data Engineering, IEEE Transactions on 26.8 (2014): 1819-1837.

[8] Tsoumakas, Grigorios, and Ioannis Vlahavas. ”Random k-labelsets: An ensemble method for multilabel classification.” Machine learning: ECML 2007. Springer Berlin Heidelberg, 2007. 406-417.

[9] Zhang, Min-Ling, and Zhi-Hua Zhou. ”Multilabel neural networks with applications to functional genomics and text categorization.” Knowledge and Data Engineering, IEEE Transactions on 18.10 (2006): 1338-1351.

[10] Grodzicki, Rafa, Jacek Madziuk, and Lipo Wang. ”Improved multilabel classification with neural networks.” Parallel Problem Solving from NaturePPSN X. Springer Berlin Heidelberg, 2008. 409-416.

[11] Simonyan, Karen, and Andrew Zisserman. ”Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).

[12] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. ”Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems. 2012.

[13] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. ”Reducing the dimensionality of data with neural networks.” Science 313.5786 (2006): 504-507.

[14] Boutell, Matthew R., et al. ”Learning multi-label scene classification.” Pattern recognition 37.9 (2004): 1757-1771.

[15] Nam, Jinseok, et al. ”Large-scale Multi-label Text ClassificationRevisiting Neural Networks.” Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2014. 437-452.

[16] Dimou, Anastasios, et al. ”An empirical study of multi-label learning methods for video annotation.” Content-Based Multimedia Indexing, 2009. CBMI’09. Seventh International Workshop on. IEEE, 2009.

[17] ”Yelp Restaurant Photo Classification.” Data -. Web. 13 Mar. 2016. ¡https://www.kaggle.com/c/yelp-restaurant-photo-classification/data¿.

[18] Tanner, Martin A., and Wing Hung Wong. ”The calculation of posterior distributions by data augmentation.” Journal of the American statistical Association 82.398 (1987): 528-540.

[19] LeCun, Yann, et al. ”Backpropagation applied to handwritten zip code recognition.” Neural computation 1.4 (1989): 541-551.

[20] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. ”Learning representations by back-propagating errors.” Cognitive modeling 5.3 (1988): 1.

[21] Bastien, Frdric, et al. ”Theano: new features and speed improvements.” arXiv preprint arXiv:1211.5590 (2012).

[22] Francois Chollet, Keras, (2015), GitHub repository, https://github.com/fchollet/keras

[23] Jia, Yangqing, et al. ”Caffe: Convolutional architecture for fast feature embedding.” Proceedings of the ACM International Conference on Multimedia. ACM, 2014.

[24] Tieleman, Tijmen, and Geoffrey Hinton. ”Lecture 6.5-rmsprop.” COURSERA: Neural networks for machine learning (2012).

[25] Van der Maaten, Laurens, and Geoffrey Hinton. ”Visualizing data using t-SNE.” Journal of Machine Learning Research 9.2579-2605 (2008): 85.

[26] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[27] Read, Jesse, et al. ”Classifier chains for multi-label classification.” Machine learning 85.3 (2011): 333-359.

[28] He, Kaiming, et al. ”Deep Residual Learning for Image Recognition.” arXiv preprint arXiv:1512.03385 (2015).

[29] Szegedy, Christian, et al. ”Going deeper with convolutions.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[30] Kang, Le, et al. ”Convolutional neural networks for no-reference image quality assessment.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.

[31] Xu, Kelvin, et al. ”Show, attend and tell: Neural image caption generation with visual attention.” arXiv preprint arXiv:1502.03044 (2015).

[32] Mahendran, Aravindh, and Andrea Vedaldi. ”Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images.” arXiv preprint arXiv:1512.02017 (2015).