# MIML Learning with CNNs: Yelp Restaurant Photo Classification

Pulkit Agrawal
Stanford University
Stanford, CA
pulkital@stanford.edu

Raghav Gupta
Stanford University
Stanford, CA
rgupta93@stanford.edu

## Abstract

*We present the conditions of a data science challenge from Kaggle, which can be viewed as a multi-instance multi-label learning problem in the image domain, and describe the official training dataset provided. We discuss our technical approach, and address the challenges in using transfer learning and with finetuning, trying out different strategies to tackle the multi-instance aspect of the problem. Lastly, we present our results, and analyze in depth the quantitative and qualitative performance of our models.*

## 1. Introduction

In this work, we tackle the problem of automatically tagging restaurants with multiple labels corresponding to restaurant attributes using unlabeled photos mapped to restaurants. Specifically, we work with the dataset provided by Yelp through a competition hosted on Kaggle.com [1]. Automatic attribute prediction from images can serve as a valuable source of business information for Yelp, helping them categorise businesses correctly and make better recommendations to users. Even in general, solutions to this problem open up further innovations in image-based categorization of entities, wherever automatic tagging would be valued and image labels aren't available for all the images.

About the setup - we have multiple unlabeled images for each restaurant, and each restaurant can in turn have some (or none or all) of 9 binary attributes. We model this problem as a Multiple Instance Multiple Label (MIML) classification problem, upon which we elaborate more in Section 2.

Broadly speaking, our models use feedforward neural networks and/or SVMs, which are trained using transfer learning, on top of a finetuned pretrained convolutional neural network for an image classification task. The input to our model is a set of images corresponding to a single restaurant, and our prediction is a subset of the set of 9 attributes, which describe the underlying entity (restaurant).

As mentioned, we explore transfer learning and finetuning techniques to train our classification models. For transfer learning, we build on a pre-trained VGG16 net [2] initially as a fixed feature extractor by removing the fully connected layers and using the hidden layer activations as features to train an MIML classifier. Depending on the results thus obtained, we further finetune the higher layers of the pretrained network to learn activation functions specific to our problem, and report results for each experiment.

The rest of the report is organized as follows - we present our formulation of the underlying classification problem - multi instance multi label (MIML) classification. We then describe related work on multi instance as well as multi label approaches to classification problems, in the image domain as well as other domains. After that, we describe the dataset in detail, including examples. Then, we write about our methods and algorithms, providing a brief theoretical background to these. We then describe our experiments, followed by a quantitative and qualitative discussion of our results. We end with conclusions and ideas for future work.

## 2. Problem Description: MIML Classification

The multi instance multi label (MIML) classification paradigm seeks to assign a variable-sized bag of features representation one or more labels from a given set of ground labels. Like a regular classification problem, there are two phases - training and testing.

During the training phase, our input is a set of labelled data points $\{X_i, Y_i\}_{i=1}^{N}$, where each $X_i = \{x_{i1}, x_{i2}, \ldots, x_{ik_i}\}$ is a set of $k_i$ instances ($k_i$ could be different for each of the $N$ training examples) corresponding to an entity and each label $Y_i \subseteq \mathcal{Y}$ is a set of attributes, which is a subset of the ground label set i.e. $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$, which consists of $n$ attributes. How $Y_i$ attaches to each of the $x_{ik_i}$s is a choice of method.

During the testing phase, we are provided a bag of features $X^* = \{x_1, x_2, \ldots, x_k\}$ supposed to correspond to a single entity, and our output must be a set of attributes $Y^* \subseteq \mathcal{Y}$ for this entity. The goal is to learn a classifier $f : X \rightarrow 2^{\mathcal{Y}}$ which, given a bag of instance features, predicts the set of attributes corresponding the underlying entity.

For our problem, a restaurant corresponds to an entity, say $X = \{x_1, x_2, \ldots, x_k\}$, and each $x_i$ in $X$ corresponds to an image of the restaurant. For our problem, we deal with 9 attributes (more in Section 4), thus $\mathcal{Y} = \{y_1, y_2, \ldots, y_9\}$.

## 3. Related Work

There has been a lot of work on MIML classification problems in general and their application to image classification tasks in particular. Typically, MIML classifiers are used for classifying images containing multiple objects. The image is divided into several segments, where each segment serves as an instance, and is associated with a a semantic label corresponding to the object it contains. The image usually gets labels of all the objects present in it.

[3] discusses two different MIML approaches for scene classification from images. Their first approach, called MIMLBOOST transfers all the labels of the entity (scene) to the instances (image segments) and learns instance level classifiers for each instance which are combined using an AdaBoost like algorithm. In their second approach, called MIMLSVM, they perform a $k-$mediod clustering to transform the multiple instances of an entity into a numerical vector which encodes structural information about the entity data, particularly what fraction of the data represents each of the several sub-concepts corresponding to the $k$ mediods (each representing one of the labels to be learnt). They then train a multi-label SVM in this transformed feature space. Both of these approaches are iterative in nature and not easily extensible to a neural network scenario.

[4] extends logistic loss function for a single instance single label classification problem to an MIML setting by training a separate classifier for each label and assuming that a label is not assigned to an entity only when all its instances suggest that it should not be assigned. They also introduce a Trace Norm regularization term in the loss which tends to learn a shared sparse weight matrix for the different label level classifiers, thus learning correlated multi-label classifiers instead of pure binary classifiers for each label. They apply their methods to an object recognition task where multiple objects may be present in a single image. They focus on linear models trainable by SGD and do not extend their results to a neural network setting. One of our MIML strategies (Weak-Or, see 5.4) is inspired from this work leaving out the regularization aspect.

Multi-instance learning has been applied to image classification tasks in CNN settings as well. In [5], the authors develop a sliding window detector which efficiently searches for an object over an image across multiple positions, scales and aspect ratios. Each window serves as an instance and the window which responds maximally to the classifier is used to compute the loss which is then used for training the deep CNN. We adapt this approach and it's slight variations in some of our MIML models (Mean, Max, Weighted Mean, see 5.4).

## 4. Dataset Description and Preprocessing

We use the dataset provided by Yelp through the Yelp Restaurant Photo Classification competition hosted on Kaggle.com at [1]. The full dataset is comprised of approximately $234,000$ untagged RGB images corresponding to $2,000$ restaurants. The number of images corresponding to each restaurant ranges from $1$ to $2,974$ across restaurants, with about $117$ images per restaurant on average. Each business can have nine self-explanatory attributes, namely

- 0: good_for_lunch
- 1: good_for_dinner
- 2: takes_reservations
- 3: outdoor_seating
- 4: is_expensive
- 5: has_alcohol
- 6: has_table_service
- 7: ambience_is_classy
- 8: good_for_kids

The images themselves are of variable size, ranging from icon-size to $500 \times 500$. However, almost all of them are larger than the required input size ($224 \times 224$) to our pretrained VGG16 net (to be elaborated later). For preprocessing, we resize all images to $224 \times 224$ and perform per-channel mean image subtraction for our experiments.

Figures 1, 2 and 3 contain examples of restaurant pictures from the dataset. While these pictures do not reflect this, almost all pictures are of food items, which can be good for labels 0, 1 and potentially 4, but not really for the others. Outdoor seating may be reflected by chairs and the exterior co-occurring in the picture, which is generally represented in the businesses having that property. Some other labels (5 and 6) may also gain from the fact that we use a pretrained CNN trained on the ImageNet classification task, since alcohol and tables correspond to certain ImageNet classification categories.

We found the full dataset too large to work with and, opting to try out a larger variety of models instead, we chose to work with only 1000 businesses, and randomly selected 16 images for each business as our data. Of these 1000 selected businesses, we chose $80 - 20$ train-val split i.e. 800 businesses ($800 \times 16$ images) for training and 200 businesses for validation.

## 4.1. Evaluation Metrics

In a multi-label setting, traditional evaluation metrics like classification accuracy do not immediately make sense because even a subset of correctly identified attributes is better than all incorrectly predicted attributes. Hence, based on past work [6], we report the following evaluation metrics to test the performance of our models: *hamming loss* and *mean F1 score*. Both these metrics are defined below for a classifier $f$ over a test set $T = \{X_i, Y_i\}_{i=1}^{m}$.

- *Hamming Loss*: The hamming loss over the set $T$ is given by $\text{hloss}(f) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{Y}|}|f(X_i)\Delta Y_i|$, where $\Delta$ is the symmetric difference between the sets $f(X_i)$ and $Y_i$. For each item-label pair, it measures the number of incorrect predictions and missed attributes. Smaller hamming loss indicates better prediction.

- *Mean F1 Score*: The mean F1 score simply computes the arithmetic mean of the F1 score obtained for each of the 9 labels individually i.e. from the precision and recall values calculated by considering the correctness of each label individually. This is the metric used to rate the submissions to the Kaggle competition.

One thing to note is that both these metrics attach an equal importance to classification performance on all 9 of the attributes, even when some labels can be much harder to classify than the others.

## 5. Methods

In this section, we first describe the basic building blocks of our models - convolutional neural networks (CNNs) and transfer learning. After that, we describe our models, with motivation and figures.

## 5.1. Preliminaries

**Neural Networks-** A neural network (NN) is a biologically inspired model of computation comprising of nodes (storing the initial/intermediate inputs, called activations) which are interconnected (each connection with it's own weight). NN computation is commonly represented in layered fashion, with one layer serving as an input to the next layer. The most common layer is the fully connected (FC) layer, which performs the computation $f(X) = WX + b$ ($W$ and $b$ - layer's parameters). ReLU layer, a nonlinear activation function $f(X) = \max(0, X)$ - typically follows each FC layer. Non-linearity ensures only a few input nodes are 'activated'. Convolutional layers, an integral part of CNNs, perform a forwarding function similar to FC, but relying on spatial locality of images. Details are omitted here.

**Transfer learning-** CNNs can get huge and slow to train, especially on regular hardware. Thus, it is common to pretrain a CNN on a very large dataset and then either use the CNN as a fixed feature extractor, or finetune parts of the CNN for the task at hand. This technique is called transfer learning.

## 5.2. Pretrained model

Relying on transfer learning, we use the pre-trained 16-layer VGG network, trained on the ImageNet, from the Caffe Model Zoo, to extract basic image features (CNN codes), and replace layers towards the top of the network. The ImageNet contains 1.2 million images with 1000 categories, and thus CNNs trained on the ImageNet are common for transfer learning. Instead of trying out various pretrained nets, we chose to perform more MIML experiments instead.

An overview of the architecture of VGG16 is shown in Figure 4. It consists of 5 [CONV-CONV-ReLU] blocks separated by $2 \times 2$ max pooling layers, followed by 3 fully connected layers before classification, with dropout layers in between the FC layers.

## 5.3. Baseline - SVM on CNN codes

As a quick baseline, and to verify the results presented in [7], we trained for each label one-versus-rest SVM classifiers on CNN codes i.e. activations of the input images when forward passed through the VGG16 net, using scikit-learn [8]. To train the SVMs, we perform 4 different experiments, using CNN codes from different layers -

- CONV4 - Output of fourth [CONV-CONV-ReLU] block after max pool
- CONV5 - Output of fifth and final [CONV-CONV-ReLU] block after max pool
- FC1 - Output of first FC layer
- FC2 - Output of second and penultimate FC layer

The locations of these layers within the VGG16 net are also shown in Figure 4. Our motive behind this experiment is to verify activations at which stage are better for off-the-shelf usage with simple classifiers, earlier or later.

## 5.4. MIML Classifiers

We now talk about the loss functions we use, adapting from single-instance single-label classification to the MIML setting.

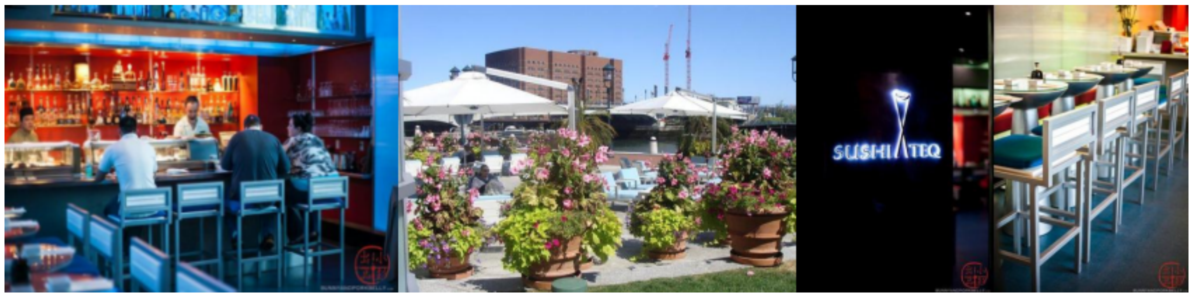


Figure 1: Photos of a sushi-tequila restaurant, with images indicative of alcohol, outdoor seating and expensiveness

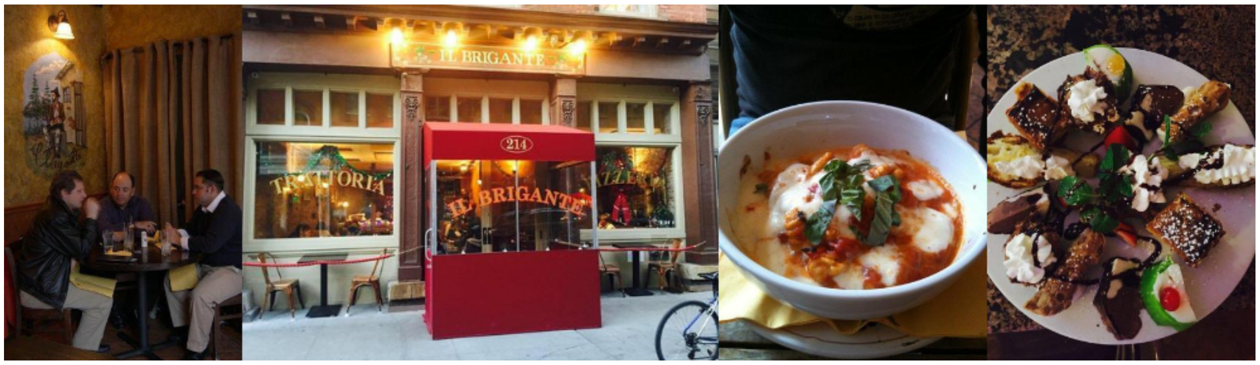


Figure 2: Photos of a classy family restaurant that has table service and takes reservations

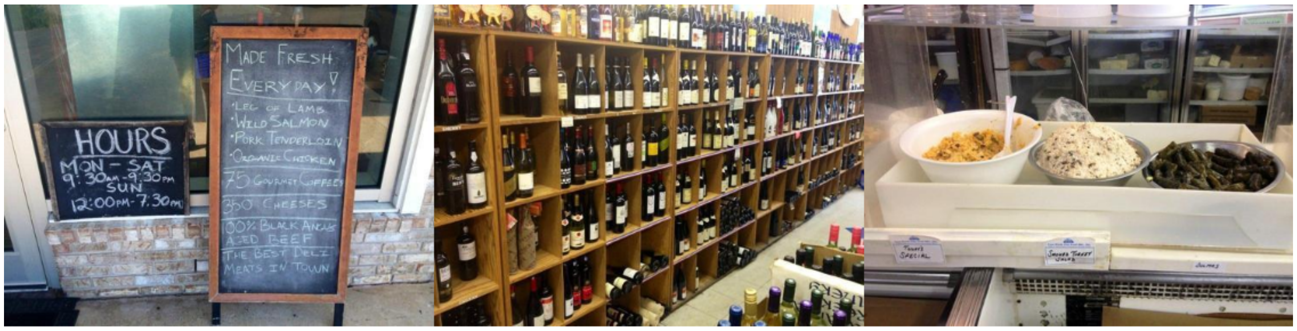


Figure 3: Photos of a restaurant with outdoor seating that is good for lunch and serves alcohol

Many of the MIML learning algorithms proposed in literature like MIMLBOOST and MIMLSVM [3] are iterative algorithms which cannot be easily integrated with CNN architectures. Therefore, we use differentiable loss functions for MIML classifiers which can be easily integrated with a traditional CNN architecture at the top layer and learnt using backpropogation.

For learning multiple attributes for each inastance, we train a binary classifier (using SVM/Softmax/Logisitc loss) for each attribute independently (9 binary classifiers for our task). We define the label vector $Y^i = [Y^i_1, \dots, Y^i_{|\mathcal{Y}|}]$ for each instance $X_i$, where $Y^i_j = 1$ if the $j^{th}$ attribute lies in the set $Y^i$ and 0 otherwise. The loss for $X_i$ is given by, $L_i = \sum_{j=1}^{|\mathcal{Y}|} L_{ij}$, where $L_{ij}$ is the loss of the $j^{th}$ classifier on the instance $X_i$. Different images of a restaurant are indicative of the presence or absence of different attributes. As a result, it becomes crucial how the inputs to each attribute's classifier are selected or aggregated and how $L_{ij}$ is computed. Below, we describe the different strategies (referred to as *example selection strategies*) that we experimented with for selection/aggregation of the input presented to each classifier and the overall loss function for each scenario.

- **Mean-** At the classification layer, say our activations are of size $N \times |\mathcal{Y}|$ where $N$ is the number of images for the restaurant and $|\mathcal{Y}|$ is the number of attributes. Let's call this activation $X^i$ for the $i^{th}$ instance. An average selection basically takes the mean

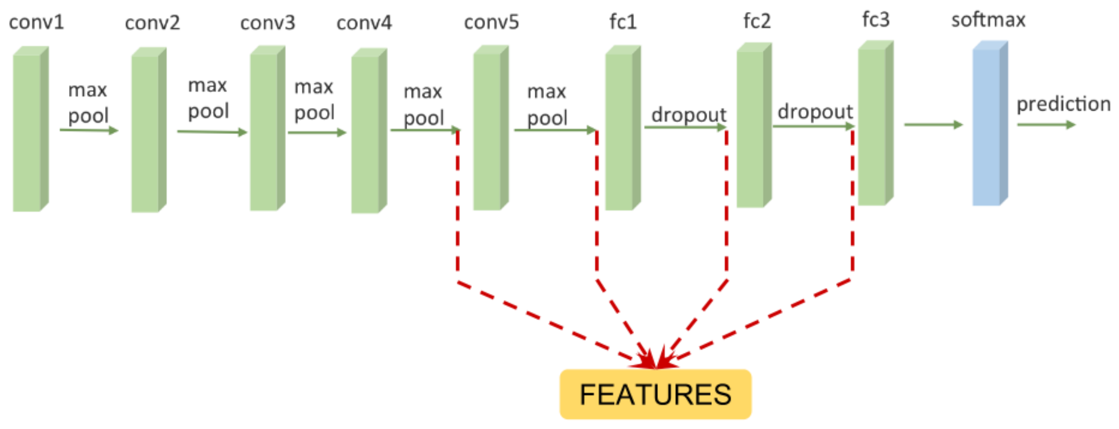


Figure 4: Overview of the VGG16 network (each conv block includes 2 conv layers followed by a relu layer), along with the locations from which CNN codes were used to train baseline SVMs

of the $N \times |\mathcal{Y}|$ activations across the first dimension, averaging out the features for each constituent image. Thus, $X_j^{i*} = \frac{\sum_{k=1}^{N} X_{kj}^i}{N}$ where $X_j^{i*}$ is the final activation over which the loss $L_{ij}$ is computed.

$$L_{ij} = Y_j^i \log\left(\frac{e^{X_j^{i*}}}{1+e^{X_j^{i*}}}\right) + (1-Y_j^i)\log\left(\frac{1}{1+e^{X_j^{i*}}}\right)$$

Note that this is same as the binary softmax loss.

- **Max-** This strategy lets each classifier select the image it responds maximally to i.e. we take the max of the activations over the first dimension to get our input to the classifier. Thus, $X_j^{i*} = \max_k(X_{kj}^i)$. The remainder is similar to the previous strategy of taking the mean.

- **Weak Or-** In this strategy suggested in [4], each image is fed to the classifier and a prediction is made based on each image. The label for the overall restaurant is negative iff the prediction for all the constituent images turns out to be negative. If any of the image leads the classifier to return that the label should be in the label set, then the label is added to the final prediction.

The loss $L_{ij}$ is computed here as

$$\begin{aligned} L_{ij} &= Y_j^i \log(p(Y_j^i = 1|X^i)) \\ &\quad + (1-Y_j^i)\log(p(Y_j^i = 0|X^i)) \end{aligned} \tag{1}$$

Note that this is same as the binary logistic loss or entropy loss. We further define a vector $[y_{1j}^i, y_{2j}^i, \ldots, y_{Nj}^i]$ for the $j^{th}$ attribute for $X_i$, where each element $y_{kj}^i$ indicates whether the $j^{th}$ attribute should be present in $Y^i$ based on the $k^{th}$ image/feature vector $x_k^i$ in $X^i$. Thus,

$$p(y_{kj}^i = 1|x_k^i) = \sigma(x_k^i) = \frac{1}{1+e^{-x_k^i}} \tag{2}$$

$X^i$ is labeled negative with respect to the $j^{th}$ attribute only when all the instances suggest that the label should be negative. Thus,

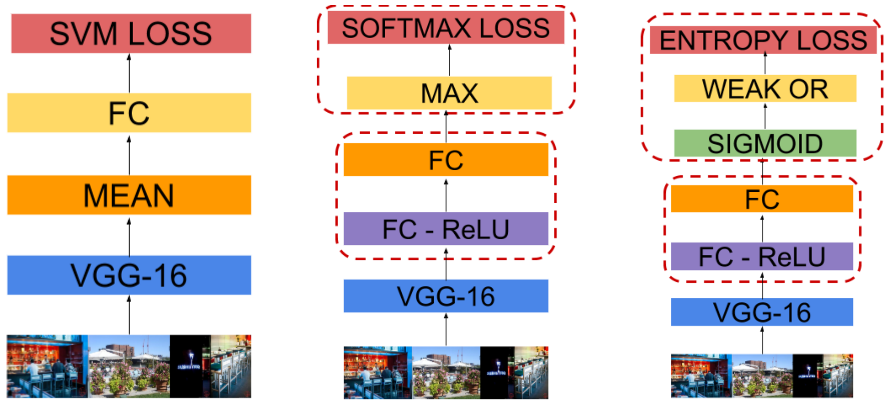
$$\begin{aligned} p(Y_j^i = 0|X^i) &= \prod_{k=1}^{N}(1 - p(y_{kj}^i = 1|x_k^i)) \\ &= \prod_{k=1}^{N}(1-\sigma(x_k^i)) \\ p(Y_j^i = 1|X^i) &= 1 - p(Y_j^i = 0|X^i) \end{aligned} \tag{3}$$

- **Weighted mean-** Taking inspiration from neural attention models [9] being used for a soft search over the input space to discover the inputs most relevant to the downstream task, we implement a very basic version of the technique that takes a weighted mean of the $N \times H$ activations across the first dimension with the following equations

$$c = XW, \alpha = cw, X^* = X^T\alpha \tag{4}$$

where $W$ is a projection matrix, $W \in \mathbf{R}^{H\times H}$, $w$ is a weight vector, $w \in \mathbf{R}^H$, both trainable.

Clearly, we will train different parameters for different labels, since the importance of each inherent feature in the activations for each label would differ.



(a) Position of mean layer for baseline SVM classifier

(b) Position of max/weighted max/mean layer

(c) Position of weak-or layer

Figure 5: Overview of multi-instance example selection models and CNN code + SVM architecture

### 5.5. Putting it together

Figure 5(b) and (c) describe the selection strategies we described in Section 5.4.

For the max, weighted mean and mean selection techniques, we add these operations on top of a two layer feedforward neural network (with final outputs of size $N \times |\mathcal{Y}|$ for the $N$ images) which in turn gets its inputs from the forward pass of the images through the full VGG-16 network **except** the last softmax classification layer which we cut off. Figure 5(b) describes this - an FC-ReLU-FC-input-selection sequence on top of the penultimate $(7^th)$ fully connected layer of the VGG16 network. For classification, as discussed, we use a per-label softmax loss.

The architecture with the weak-or, shown in 5(c), is more or less similar in terms of the two-layer feedforward NN after the VGG16, except the classifier is applied to each of the $N$ images and the probability of each label being in the true label set is computed as described in Section 5.4, followed by a entropy or logisitc loss on this probability distribution, summed over all labels.

Note that for all these models, we treat an entire 'bag' of 16 images as a single entity and pass it through the network for training.

### 5.6. Implementation

We used Torch for implementation, and ran initial experiments on a CPU before moving to an AWS instance with 1 NVIDIA GPU with 4GB memory and 8 Intel Xeon CPU cores, with CUDA 7.5 and CuDNNv3 installed.

While running experiments, since we previously select 16 images per business (owing to memory concerns), we run our models providing as input 16 images of the same business. Since this is just one instance being processed, it is equivalent to pure stochastic gradient descent (SGD) with momentum. We use momentum 0.9 for all our experiments.

## 6. Results

We now go through our results, presented in Table 1, for each set of experiments that we perform, section by section.

### 6.1. CNN code SVMs

We trained one-versus-rest SVMs for each label on the CNN codes on the input activations from the CONV4, CONV5, FC1 and FC2 layers. The results are presented in the first four rows of Table 1.

We observe that an SVM trained on the FC layer features provides a simple yet very competitive baseline for the task. The SVMs trained on FC1 and FC2 layer features achieve good mean F1 scores of 0.77 and 0.79 respectively, achieving 0.8+ F1 scores for 4 of the 9 labels in the task. The activations at both these layers are of hidden dimension 4096 for each image, and mapping these many features to a 9 binary label space is seemingly tractable.

In contrast, the SVMs trained on CNN codes from the later convolutional layers (after CONV4 and CONV5) perform miserably, performing worse than random on 3 of the 9 labels. The activations on these layers are of size $512 \times 14 \times 14 \sim 100k$ and $512 \times 7 \times 7 \sim 25k$ per image respectively. These correspond to image features that are highly sparse and spatially local in nature, and it is very likely that a single linear layer is unable to combine these activations in order to get meaningful semantic information about the image - deeper and more structured computation would be required for that.

Another thing to note is that the difference between the SVMs trained on FC1 and FC2 is minor (0.77 versus 0.79). Other than a big jump in the performance of SVM-FC2 on label 3 (outdoor seating), the two are neck-to-neck in terms of classification F1 scores. This suggests that the convolutional layer activations, after having been put through a single fully connected layer, already begin to represent relevant semantic information that enable the simpler SVMs to classify the images with reasonable accuracy, and that (possibly) one or two FC layers on top of the CONV layers in VGGNet might be similar in performance to the three layer version, which could lead to massive parameter savings : the FC layers contain most of the parameters of the network. This also indicates sparsity in these layers and opens up the possibility to prune the FC portions of these networks using self-pruning techniques [10].

## 6.2. Example selection strategy FFNNs

The next four columns of Table 1 present our numbers on different example selection strategies for multi-instance classification, on top of a 2 layer FC-RELU-FC network. We used a learning rate in the ballpark of $6 \times 10^{-5}$ to $1 \times 10^{-4}$ to train the FFNN, with a relatively strong regularization constant of $1 \times 10^{-3}$ in each case.

Contrary to our expectations, we report the best numbers on plain old averaging of all the image activations (2NN-MEAN), which gives us an F1 score of 0.79, equivalent to our best SVM baseline, but slightly ahead on Hamming loss (0.206 versus 0.211). The max strategy, designed to most clearly separate the classification of each label, performs below par on labels 0 and 1 : good for lunch/dinner. This is quite possibly because of our network underlying the max layer being fairly shallow - we believe a deeper network would allow the max to perform better. The weighted mean model is likely hurt by the same problem - a shallow depth.

The weak-or model, in retrospect, comes with a very strong bias towards adding labels to the prediction. While it does fine on labels with over 50% majority on the validation set, it gives poor numbers on other labels - particularly 0 and 4 (good for lunch and expensive, respectively).

A peek into the internals- Figure 6 shows the training avg softmax loss and train/val mean F1 scores (averaged across labels) versus training epochs, for the 2NN-Mean model. Surprisingly enough, the loss fluctuates a lot (indicating a downward trend, but still very strange), but the training and validation mean F1 scores are reasonable.

## 6.3. Other models

Based on these, we also tried an ensemble of three of our best models (2NN-MEAN, SVM-FC2 and 2NN-WMEAN), employing a majority voting mechanism to make the overall classification decision for each label. However, we could not achieve any significant improvement over the individual SVM-FC2 and 2NN-MEAN models, achieving an average F1 score of about 0.79 using the ensemble.

In addition, we also tried finetuning the VGG16 net for the task - precisely, the fifth and final CONV-CONV-RELU block, by incorporating it into the trainable part of the network, but using at all stages for this part of the network a learning rate which was a hundredth of the learning rate used for the FC layers. This was to ensure that our already-decent conv layer weights were not perturbed too much, especially since the FC layers were being trained from scratch. For various choices of the hyperparameters, however, we failed to extract an improvement in the mean F1 score, and are hence omit its results here.

## 6.4. Individual label performance

In this section we discuss the performance of the classifiers on the 9 individual labels. The classifiers achieve 0.9+ F1 score on labels 5 and 6 (alcohol and table service respectively), while the worst performing labels are 7 and 3 (classy ambience and outdoor seating respectively).

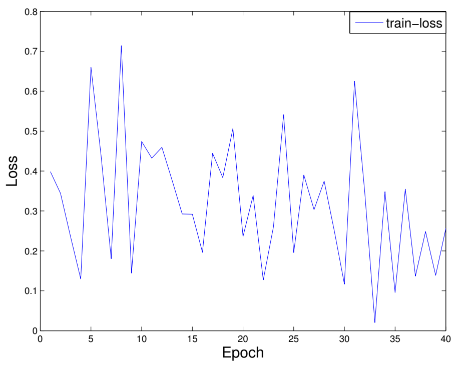
Approximately 70% of the pictures of a restaurant are of food items, a good number of these being shots of various items kept on the table. These, in our opinion, could provide sufficient information, given enough training, for labels 0, 1, 5 and 6 (good for lunch/dinner, alcohol, table service respectively) - since images of food items could be indicative of lunch/dinner, and alcohol and table service might be indicated by the other aspects of a full-table image.

However, the VGG16 net we use is trained on ImageNet categories, which correspond to real-world objects. 6 of the 1000 ImageNet categories correspond to alcoholic items, and 2 of them correspond to tables (a few others for miscellaneous furniture, relevant for label 6). Labels 0 and 1, however, tend to be more abstract in nature, and would require in-depth training to identify food item pictures as good for lunch/dinner.

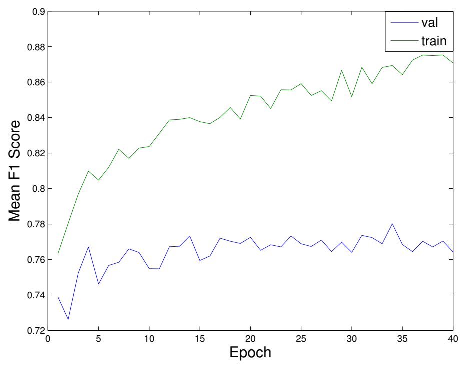
Another important class of input images covers the exterior of the restaurant, more often than not including the restaurant's sign. These provide ample evidence for label 3 (outdoor seating). In spite of this, the results for this label were rather poor (best mean F1 score 0.68). We reason this could also be caused by the ImageNet classes being fine

| Model \| Label | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Avg | HamLoss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SVM-CONV4 | 0.34 | 0.58 | 0.50 | 0.52 | 0.37 | 0.73 | 0.79 | 0.44 | 0.75 | 0.56 | 0.453 |
| SVM-CONV5 | 0.37 | 0.59 | 0.53 | 0.54 | 0.40 | 0.73 | 0.76 | 0.47 | 0.77 | 0.58 | 0.446 |
| SVM-FC1 | 0.56 | 0.77 | 0.81 | 0.57 | **0.82** | 0.87 | 0.90 | 0.62 | 0.83 | 0.77 | 0.224 |
| SVM-FC2 | 0.57 | 0.77 | 0.82 | **0.69** | 0.81 | 0.89 | 0.91 | 0.60 | 0.81 | **0.79** | 0.211 |
| 2NN-WEAKOR | 0.61 | 0.83 | 0.83 | 0.59 | 0.65 | 0.89 | 0.92 | **0.64** | 0.82 | 0.77 | 0.231 |
| 2NN-MAX | 0.54 | 0.77 | 0.83 | 0.58 | 0.73 | 0.89 | 0.90 | 0.63 | 0.83 | 0.75 | 0.226 |
| 2NN-MEAN | **0.68** | **0.83** | **0.84** | 0.64 | 0.79 | **0.92** | **0.94** | 0.59 | **0.84** | **0.79** | **0.206** |
| 2NN-WMEAN | 0.66 | 0.78 | **0.84** | 0.66 | 0.76 | 0.89 | 0.93 | 0.63 | 0.82 | 0.78 | 0.215 |
| Majority | 0.57 | 0.66 | 0.64 | 0.68 | 0.65 | 0.75 | 0.83 | 0.60 | 0.76 | 0.68 | 0.307 |

Table 1: Overview of classification results; columns 0-8 and avg show each label's F1 scores, or the average, and HamLoss shows the hamming loss. All numbers are on the validation set. Omitted - numbers for ensemble and finetuned models



(a) Training loss vs epoch for 2NN-Mean



(b) Train/val accuracy vs epoch for 2NN-Mean

Figure 6: Behavior of loss function and train/val accuracy for 2NN-Mean classifier

tuned to detecting and classifying prominent objects in an image, and outdoor seating shots are not that common in the dataset.

The other labels (reservation-taking, expensive, classy ambience and kid-friendly) have a more abstract nature to them. However, surprisingly, for the labels reservation-taking and kid-friendly, our classifiers achieve respectable performance (0.84 mean F1 score for each), while the other two attributes, which we expect to see a strong correlation between, get mean F1 scores of just $\sim 0.64$.

## 7. Conclusion and Future Work

As part of this project, we explored the MIML classification paradigm. This has previously been applied in the image domain to identify multiple objects (instances) in a single image, but whether or not the same techniques would work for this dataset, with potentially very different images for each entity, remained to be seen. We took a transfer learning approach to this problem using a pretrained VGG16 net, coming up with baselines and investigating strategies for the multi-instance angle of the problem, also experimenting with finetuning and ensemble models (albeit with not-so-great results). We finally report a reasonable mean F1 score $0.79$ on a subset of the full data with a 2-layer FFNN on top of the VGG16 pretrained net.

Training and testing our models on the entire dataset will be the first line of future work, which would enable us to understand the scalability of these models. Many of our models were trained just using the features extracted from a VGG16 network independent of the deep network itself. This allows several other MIML algorithms to be tested, like MIMLSVM and MIMLBOOST as discussed in [3]. It would be interesting to test whether more complicated models like these can perform better on the CNN codes than simple linear and 2 layered NN models. Res-Nets [11] have shown promising results on ImageNet, improving upon all previous models. It would be interesting to fine tune these networks for the MIML problem and see whether they can perform better on these tasks as well.

## References

[1] "Yelp restaurant photo classification," *https://www.kaggle.com/c/yelp-restaurant-photo-classification*.

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

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

[4] O. Yakhnenko and V. Honavar, "Multi-instance multi-label learning for image classification with large vocabularies." pp. 1–12, 2011.

[5] G. Papandreou, I. Kokkinos, and P.-A. Savalle, "Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 390–399.

[6] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *arXiv preprint arXiv:0808.3231*, 2008.

[7] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[8] 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: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[10] S. Han, H. Mao, and W. J. Dally, "A deep neural network compression pipeline: Pruning, quantization, huffman encoding," *arXiv preprint arXiv:1510.00149*, 2015.

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