Yelp Restaurant Photo Classification

Rakshit Gautam
111168264
Grad Student, CSE, Stony Brook University
rgautam@cs.stonybrook.edu

Namita Mhatre
110929172
Grad Student, CSE, Stony Brook University
nmhatre@cs.stonybrook.edu

Abstract

Yelp Restaurant Photo Classification is a challenge on Kaggle. The challenge requires us to predict attribute labels for restaurants using user-submitted photos. The problem of predicting labels is a multi-label and a multi-class classification problem. We have used a pretrained VGG-16 neural network model and an Inception V3 pretrained model from Tensor-flow for computing the image features and have used these features for training a multi-label multi-class SVM (Support Vector Machine) classifier using the Onevs-Rest from the Sklearn library in Python (after merging these features to generate business level features).

1. Introduction

Yelp Restaurant Photo Classification is a Kaggle challenge that requires us to predict labels for a business, based on user submitted photos and tags for a business. The training data-set given to us by Yelp comprises of 2000 businesses assigned multiple labels from 0-8. Each of the label represents a tag associated with a business. For these 2000 businesses, we have 2,34,842 images which are labeled. Yelp has also provided a data-set for testing, but since the labels for that data are not available to us, we have not used the test data-set and split the train data-set given to create new training and testing data-sets. The labels associated with the businesses are -
0: good_for_lunch
1: good_for_dinner
2: takes_reservations
3: outdoor_seating
4: restaurant_is_expensive
5: has_alcohol
6: has_table_service
7: ambiance_is_classy
8: good_for_kids.
A business can be assigned multiple labels from these labels.
Multiclass classification means a classification task with more than two classes; e.g., classify a set of images of fruits which may be oranges, apples, or pears. Multiclass classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time.
Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A text might be about any of religion, politics, finance or education at the same time or none of these.
This paper shows how the challenge was solved using two methodologies. The first used a VGG-16 pretrained network while the second used Inception V3 from Tensor flow. Then a SVM classifier was used to generate the model and to predict the test data. Also, both models were compared based on their F1 scores.

1.1. Data-Set

The data-set obtained from Kaggle comprised of 2,34,842 images and is a huge data-set which meant that we were required to do the computations on a GPU. We had a 2GB AMD Radeon Graphic card available with us, and were willing to use this with python libraries for the task. The python libraries like Theano and Lasagne have GPU support for Nvidia graphic cards and use CUDA. These libraries are in development stage for use of OpenCL, and do not provide much support for AMD graphic cards at this point of time. Thus, we decided to reduce the amount of data-set to work on.

We decided to train the model on 100 businesses and chose these businesses randomly (following a uniform probability distribution) from the yelp businesses while ensuring that they cover all of the 9 labels. Hence, the training data-set used by us, comprises of 100 businesses and 14133 images. We have randomly chosen 50 businesses from the remaining 1900 businesses of Yelp data-set (following a uniform probability distribution), and we have 5694 images for testing.

The Fig 1. shows some sample images for business ID 9

Figure 1. Sample images for business ID 9

whose labels are : 1, 2, 4, 5, 6, 7. i.e. good for dinner, takes reservation, is expensive, has alcohol, has table service, and ambiance is classy. These are 6 sample images. Every business have approximately 100 images from which features are extracted and the mean is taken for every business.

1.2. Approach

Features for images were generated using the pretrained VGG-16 model and Inception V3 model. As we need to predict the labels for every business, we need to have collective features for each business. Thus, we need to aggregate the features of images corresponding to the same business. Thus, after getting the feature vectors for each image, the mean of the features were calculated for each business. The SVM classifier was then trained using these 100 feature vectors corresponding to 100 businesses. After training the model, the test cases were predicted and then the F1 score was calculated as a measure of performance. Then the VGG-16 and Inception V3 models were compared based on this score. We also computed the VLAD descriptors to represent a business, but later decided to drop them as they were less informative (The values were very close to zero which meant that the feature vector values for a business were very close to the mean feature vector)

2. Algorithms and Models

2.1. VGG-16 Architecture

VGG-16 neural network comprises of 16 Weight Layers. Figure 2 shows the architecture of VGG-16. The following is the sequence of layers in a VGG-16 network.

- InputLayer(shape=(None, 3, 224, 224))
- 3x3 ConvLayer- 64 channels (pad=1)
- 3x3 ConvLayer- 64 channels (pad=1)
- MaxPoolLayer (2x2 window, stride=2)
- 3x3 ConvLayer- 128 channels (pad=1)
- 3x3 ConvLayer- 128 channels (pad=1)
- MaxPoolLayer (2x2 window, stride=2)
- 3x3 ConvLayer- 256 channels (pad=1)
- 3x3 ConvLayer- 256 channels (pad=1)
- 3x3 ConvLayer- 256 channels (pad=1)
- MaxPoolLayer (2x2 window, stride=2)
- 3x3 ConvLayer- 512 channels (pad=1)
- 3x3 ConvLayer- 512 channels (pad=1)
- 3x3 ConvLayer- 512 channels (pad=1)
- MaxPoolLayer (2x2 window, stride=2)
- 3x3 ConvLayer- 512 channels (pad=1)
- 3x3 ConvLayer- 512 channels (pad=1)
- 3x3 ConvLayer- 512 channels (pad=1)
- MaxPoolLayer (2x2 window, stride=2)
- DenseLayer (Fully Connected)- 4096 channels
- DenseLayer (Fully Connected)- 4096 channels
- DropoutLayer
- DenseLayer (Fully Connected)- 1000 channels
- SoftMax layer

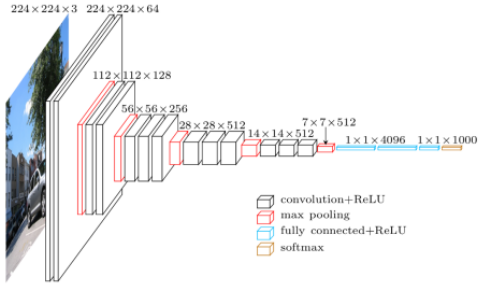

Figure 2. VGG-16 Architecture

2.1.1 Convolution layer

During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input.

2.1.2 Maximum Pool Layer

It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features. The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control over-fitting.

2.1.3 ReLu (Rectified Linear Units)

It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

2.1.4 Softmax Layer

The loss layer specifies how the network training penalizes the deviation between the predicted and true labels and is normally the last layer in the network.

2.2. Inception V3

The inception V3 has three more different types of layers than VGG-16. We have Average pool layer, Concatenation layer and a dropout layer. The Inception V3 has already been trained for ImageNet Large Visual Recognition Challenge, 2012 data-set with an effective batch size of 1600 across 50 GPU's, where each GPU ran a batch size of 32. We are using this pre-trained model from tensorflow for classification.
Inception V3 reaches a 3.46% on the top-5 error where as the VGG-16 yields a 6.8% on the ILSVRC-2012 data-set.
Also, the computation cost of VGG-16 is much higher as compared to Inception V3. The only drawback of Inception V3 is its architectural complexity. We can see from Figure 3 the highly complex Inception V3 architecture.

2.3. SVM

This strategy, also known as one-vs-all, is implemented in OneVsRestClassifier. The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n_classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy and is a fair default choice.

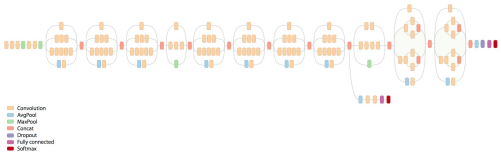

Figure 3. Inception V3 Architecture

3. Methodology

3.1. Feature Computation

The VGG-16 network is pre-trained on ImageNet data-set. We have acquired the pre-trained VGG-16 network and Inception V3 as a pickle file. The predictions for our training images on this pre-trained network will be used as features for modeling our classification problem. We take the output of the last Dense layer from VGG-16 with 4096 channels and Avg pool layer with 2046 channels as feature vector for an image. These features are then fed to the Neural network for training.

3.1.1 Feature Computation for VGG-16

The VGG-16 network is pretrained on ILSVRC-2012 data-set. We have acquired the pretrained VGG-16 network as a pickle file. The predictions for our training images on this pretrained network will be used as features for modeling our classification problem. We take the output of the last Dense layer with 4096 channels (written in red) as feature vector for an image. The images have to be converted to a standard size of 224x224x3 for passing into the neural network. For doing this, we have re-sized the images such that the smaller dimension of images is 224 while maintaining the aspect ratio. Then, we crop 224 pixels from the center along the bigger dimension. We restructure these image into 224x224x3, subtract mean VGG network image from these images and feed them into VGG-16 to obtain the output of last dense layer with 4096 layers as feature vector for every image.

3.1.2 Feature Computation for Inception V3

Inception V3 is also a pretrained network on the ILSVRC-2012 data-set. The features were extracted from the last

Average pool layer of the network. This layer has 2046 channels. The images were reconstructed to 299x299x3 as in the previous case before feeding them to the network.

3.2. Business Level Features

We have taken the mean of all the image features belonging to a particular business as the representative feature vector for that business. We also computed the VLAD descriptor for each business, but the range of these values was of the order of 0.001 meaning that the individual features varied by a lesser amount from the mean and that the mean could be a better representative of a business.

3.3. Classification

For classification, we have used the multilabel OneVsRest Classifier from sklearn package in python. The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. It uses the SVM (Support Vector Machine) classifier. N classifiers are generated for N different labels and are then integrated to form a single model. In our case, we have 9 labels. So, 9 individual classifiers are generated and are combined into a single model. We use this trained model for testing the data.

3.4. Performance of Model

To judge the performance of the model, we used the F1 score. F1 score is the trade-off between precision and recall. Precision is the fraction of retrieved instances that are relevant and recall is the fraction of relevant instances that are retrieved. We computed the mean F1 score by computing a weighted mean of individual F1 scores of all classes. We then compared this mean score for VGG-16 and Inception V3. The F1 score is weighted with the number of businesses in each label.

The $F1$ score is -

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

where

$$precision = \frac{truepositives}{truepositives + falsepositives}$$
$$recall = \frac{truepositives}{truepositives + falsenegatives}$$

4. Code Flow - Implementation Details

4.1. Store the Mappings

For solving the problem, we have created initial dictionaries to store the mappings. The following dictionaries have been created -

- image_to_bid - gives us the BusinessID for an image
- bid_to_image - gives us the list of images for a BusinessID
- bid_to_label - gives us the list of labels associated with a BusinessID
- bid_train - is a list of chosen BusinessID's for training set
- bid_test - is a list of chosen BusinessID's for test set
- images_train - is a list of images in training set
- images_test - is a list of images in test set

4.2. Feature Vector Computation - VGG-16

4.2.1 Loading VGG-16 Network

For loading the VGG-16 network we perform the following

- Load the pickle file for pretrained VGG-16 neural network. This file contains information about Weights, B values and Image Mean.
- Create an empty network using lasagne.layers module. We create a dictionary to map each layer number with empty layers created using lasagne.layers.
- Load the parameters from pickle file into the empty neural network by using set_all_param_values(last layers of empty network, all the parameters)

4.2.2 Computing Features

- Create a function getF(input) that takes a 4-D Tensor and returns the output of Dense Layer. We do this using lasagne.layers.get_output(layer,input).
- For each image in images_train, convert it into 3x224x224 size image, subtract the mean VGG image from this and call getF() on this to get the Feature Vector for each image in training set. Do the same for all images in test set.

4.3. Feature Vector Computation - Inception V3

4.3.1 Loading Inception V3 Network

- Pretrained Inception V3 neural network is obtained in the form of a graph file (.pb extension).
- An empty graph is created using tensorflow.graph() and graph definitions are loaded into this graph by reading the above graph file

```
model_fn = 'classify_image_graph_def.pb'
graph = tf.Graph()
sess = tf.InteractiveSession(graph=graph)

with tf.gfile.FastGFile(model_fn, 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
tf.import_graph_def(graph_def)
```

4.3.2 Computing Features

- The following code stores the InceptionV3 features for image.jpg in fv

```
with tf.Session() as sess:
  rep = sess.graph.get_tensor_by_name('import/pool_3:0')
  im = tf.gfile.FastGFile('image.jpg', 'rb').read()
  fv = sess.run(rep, {'import/DecodeJpeg/contents:0': im})
  fv = fv.reshape(1,2048);
```

4.4. Business Level Features

Initially we computed the VLAD descriptors for a business to use a feature vector for a business. On computing the VLAD descriptors for 100 businesses, we found that the values of these descriptors were considerably small (with maximum value being 0.008 and most of the values being zero when computed for VGG feature vector data). Since, VLAD descriptors carry very less information about the business as their low values represent that most of the business images are very close to the business image mean, we decided to use the mean feature vector for each business as the representative of a business. So, we represent a feature vector for a business as the average of all the image feature vectors for that business.

4.5. SVM Classification. Model Training

- Given the mapping of each business id to its labels, we compute the output values as a list of list of labels. And then use MultiLabelBinarizer() from sklearn package in python as follows -

```
Ytrain=MultiLabelBinarizer().fit_transform(Ytrain)
```

- Then we create an object of OneVsRestClassifier in python and pass the data-set (shape=[number of training businesses, size of feature vector]) to fit() and dump the classifier into pickle file as follows -

```
clf = OneVsRestClassifier(SVC(kernel='linear'))
clf.fit(arr, Ytrain)
with open('classifier.pickle', 'wb') as handle:
        pickle.dump(clf, handle)
```

4.6. Prediction

- We use the classifier created above to predict the labels on the test data-set. The variable test_data has shape (Number of Test Businesses, Size of Feature Vector)

```
with open('classifier.pickle','rb') as handle:
    clf = pickle.load(handle)
pred = clf.predict(test_data)
```

5. Results

The results on the test data-set of 50 businesses are -

Class	VGG-16 F1 score	Inception V3 F1 score
0	0.5882	0.6285
1	0.7586	0.7407
2	0.76	0.7916
3	0.5128	0.6521
4	0.6666	0.6206
5	0.8196	0.807
6	0.8857	0.9117
7	0.6666	0.8
8	0.7936	0.7666

Figure 4. F1 scores for individual classes

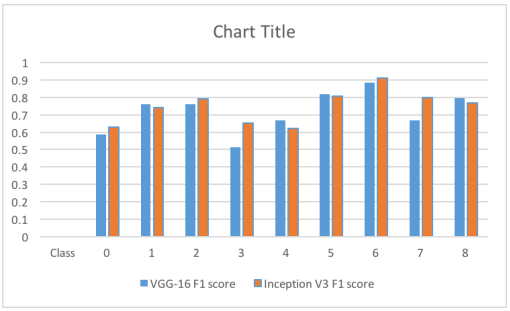

Figure 5. VGG-16 vs Inception V3

We got a mean F1 score of **0.7425** using VGG-16. We got a mean F1 score of **0.7670** using Inception V3.

We can see that the overall mean F1 score is better for InceptionV3 as compared to VGG-16 neural network. We can see that there are some classes where VGG-16 seems to perform better than Inception V3 network. These are classes 1,4,5 and 8. Inception V3 has better performance overall and in majority of classes.

Even though the overall mean F1 score of 0.767 is achieved, we have F1 scores for individual classes that are way above this score. Inception V3 gives an F1 score of 0.9117 for class 6 which means that it could classify labels of class 6 with very high accuracy, while we achieved

minimum class score of 0.6206 for Inception V3 in class 4, which is considerably below the mean score.

6. Testing

For further testing the system, we took pictures of two restaurants in Stony Brook; Curry Club (Figure 4) and Cheesecake Factory (Figure 5). We tested our model on these images. Following were the predicted labels:

Cheesecake Factory:

VGG-16 : 0,1,2,3,4,5,6,8
Inception V3 : 0,1,2,3,4,5,6,7

Labels : Good for lunch, Good for dinner, takes reservations, outdoor seating, expensive, has alcohol, has table service.

From the images, it is evident that the restaurant has alcohol, has table service and is good for lunch and dinner. Also, from our point of view the ambiance seems nice but because the model learned from some pre-labeled images, it could not predict that. Also, VGG-16 predicted that it is good for kids. In the images there is one image with kids. Thus this prediction was fairly good.

Curry Club :

VGG-16 : 1,2,3,5,6,7
Inception V3 : 1,2,3,5,6

Labels : Good for dinner, takes reservations, outdoor seating, has alcohol, has table service, ambiance is classy.

There is one image with alcohol, thus label 5 was predicted correctly. The ambiance, as we can see from the images, is very classy. The label for outdoor seating is erroneous as there is no such image. Other than that, most labels were predicted correctly.

7. Conclusion

Thus, we trained our classifier using both VGG-16 and Inception V3 pre-trained models and from the results it is evident that the Inception V3 is more efficient than VGG-16 and provides a better mean F1 score for this problem.

References

[1] Lasagne Documentation :. https://lasagne.readthedocs.io/en/latest/.

[2] Tensor Flow Documentation. https://www.tensorflow.org/versions/master/tutorials/image_recognition/index.html.

[3] VGG Home. www.robots.ox.ac.uk/~vgg/practicals/cnn/index.html.

[4] VGG in Tensor Flow. https://www.cs.toronto.edu/~frossard/post/vgg16/.

[5] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

Figure 6. Curry Club

Figure 7. Cheesecake Factory