Sentiment Analysis on Images by CNN

Yimei Tang

***Abstract*—In my project, I use 400 labeled images of a hotel from TripAdvisor taken by hotel guests.I explore two CNN models in performing the sentiment analysis but the accuracy result in the test set is only 40% to 45%. Although the model performance is not too good. I learnt how to fine tune a model in Tensorflow by applying regularization, optimizing the gradient descent by using ADAM method, normalizing the input batch and using different size of mini-batch.I then use the last fully connected layers' output from the best model as input in Kmeans clustering and PCA analysis. It turns out that 50% of the variance of the 1024 outputs could be represented by 9 PCs. I hope to continue exploring in that direction.**

***Keywords*—Sentiment Analysis, CNN, Hospitality,**

I. Introduction

A. Research Question

In this project, my research question is: is it possible to use images alone to perform sentiment analysis?

Currently, most of the sentiment analysis use text data as input. However, with the fast development of CNN and the increasing size of database image, the subject of applying image in CNN to perform sentiment analysis is worth studying. Not only could we supplement image sentiment analysis to text sentiment analysis, we could also find latent factors in images that determines the sentiments that is hard to find in text.

I start from a small dataset of 400 labeled images of a hotel. The images are all taken by guests(no professional pictures are included since they don't have ratings) with a rating ranging from 1 to 5, 1 as the lowest. Since no similar researches have been done in this area, I use human-error as the benchmark for my model accuracy score. However, it is hard to get the human-error since there is no prior knowledge of this subject. Since it is a five-class classification problem(rating 1 - 5) and the sentiment analysis is subjective, I use 50 accuracy score as the human-error for now.

Using images alone to perform sentiment analysis is hard since it is subjective, unlike image recognition. I will use this project as a starting point and continue my research in future work.

B. Inspiration and Application

Using image to perform sentiment analysis could help in these following areas:

1) When people take a picture on their social media, before they type any text, we could predict their ratings.
2) When a lot of images are uploaded to the database without labeled ratings or captions, by predicting the ratings or captions, we could improve information retrieve accuracy.
3) Similar images might have opposite rating(such as one image is given rating 1 while the other is given rating of 5), the difference between these images is worth studying.

II. Data set, Model and Methodology

A. Data set

I downloaded 1000 images from the hotel on TripAdvisor(using Image Downloader a Chrome extension), filtered out photos taken by Experts and hotel management, then hand labeled the remaining images with ratings. However, I couldn't match all pictures with the ratings on the website, therefore, I eventually have 400 pictures with labeled ratings.

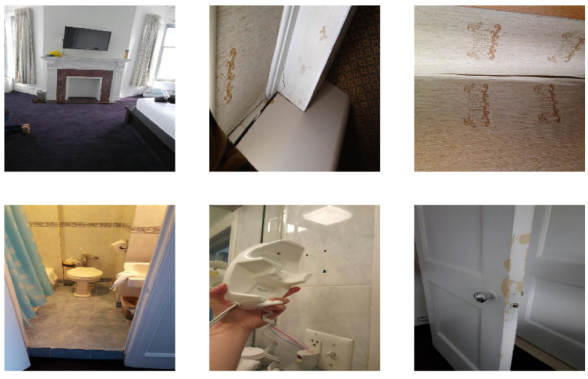
Below is the rating distribution:

```
Number of Images of Rating 1: 69
Number of Images of Rating 2: 81
Number of Images of Rating 3: 75
Number of Images of Rating 4: 138
Number of Images of Rating 5: 37
Total Images: 400
```

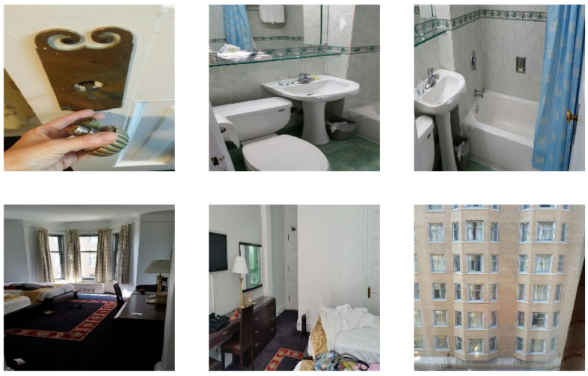
Although it is unbalanced with more data in rating 4 and less data in rating 5, the size of the data set limits me to trim the data to a balanced data set. In my future work, I will use AMT(Amazon Mechanical Turk) to get more data and have a balanced data set.

After getting the data labeled, I converted each image to 3-d numpy array of their RGB values.However, since each picture has different size, after exploring some classic CNN models, most of the input pic size is no larger than 227*227. Therefore, I re-sized all the images to 227*227. Now each image is represented by a numpy array, shape of (227,227,3). Since I only have 400 data set, I will not use validation set but only use 10% as the test set and the remaining 360 images as training set. I randomly shuffled the 400 images before splitting into training set and test set.

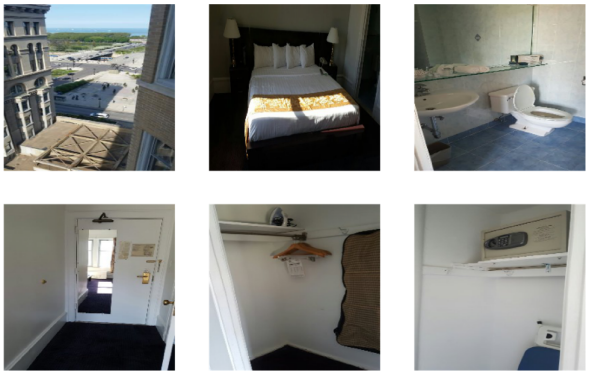
Images of Rating 1:



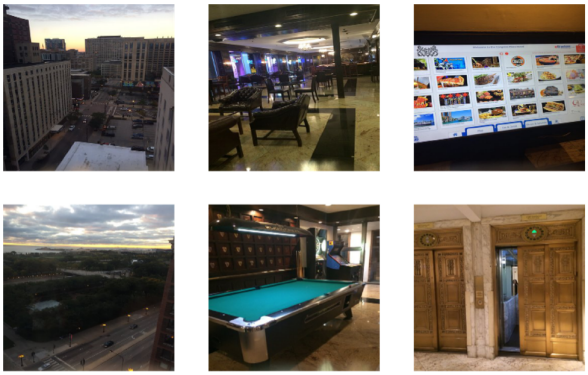
Images of Rating 2:



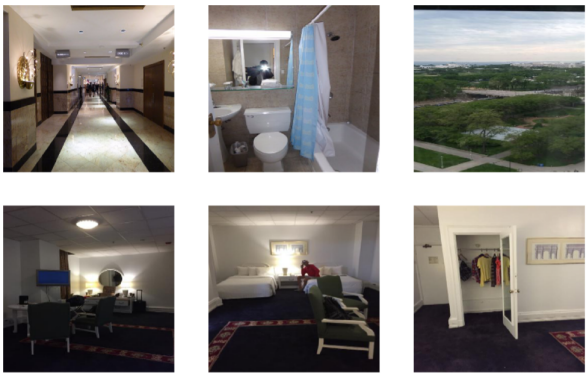
Images of Rating 3:



Images of Rating 4:



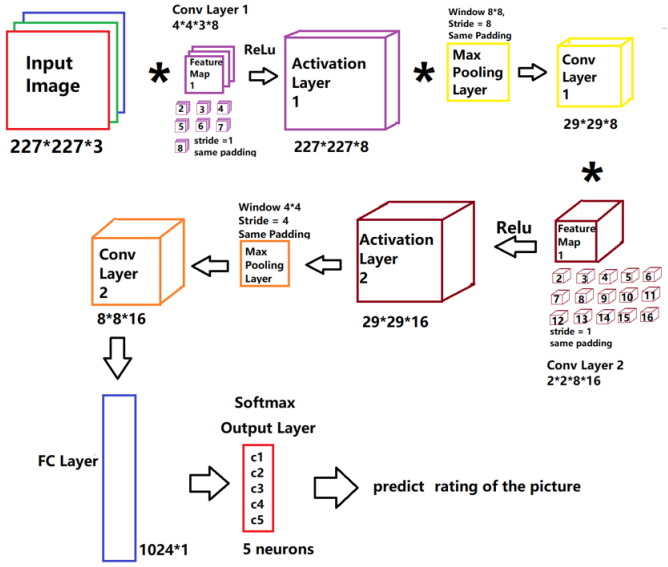
Images of Rating 5:



As you could tell from these images, it is hard for a human to classify which rating does this image has. However, it is relatively easy to spot "Rating 1" images than images from other ratings. Rating 1 images usually have stains or broken objects in the images.

B. First Model

I first built a very simple Conventional Neural Network on Tensorflow. The structure of the CNN is as follows: [1]



In this model, I use the following functions in Tensorflow:

1) tf.nn.conv2d
2) tf.nn.max_pool
3) tf.nn.relu
4) tf.contrib.layers.flatten
5) tf.contrib.layers.fully_connected
6) tf.nn.softmax_cross_entropy_with_logits
7) tf.train.AdamOptimizer
8) tf.nn.l2_loss

I learn there are different methods to optimize Gradient Descent:

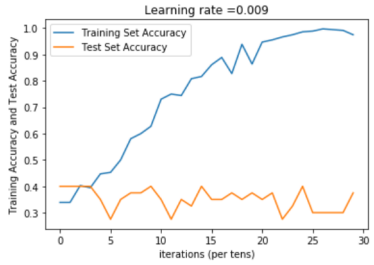
1) Gradient Descent With Momentum:
   When using normal gradient descent, we might find the cost oscillate to the local minimum but waste many iterations going back and forth on the orthogonal axis. Gradient descent with momentum helps the cost to move towards to local minimum by moving less in the orthogonal axis by taking derivatives during past iterations of gradient descent into considerations. It only uses 1 parameter.
2) RMSprop:
   Similar to Gradient Descent With Momentum, we use past derivatives of weights as factors in calculating this iteration's derivatives of weights to ensure the loss moves faster towards the local minimum. It only uses 1 parameter.
3) Adam:
   Adam is short for Adaptive Moment Estimation, it is a combination of the Gradient Descent With Momentum and RMSprop by using both of the parameters from these two methods.

I also understand the importance of normalizing the batch. In my implementation, I normalize the input by dividing them by 255 (min max normalization). After normalizing the input, it makes the cost function to optimize faster and it also has regularization effect.

C. Performance

First, I run the model with mini-batch size of 64 and 30 epochs.

```
Cost after epoch 0: 1.886790
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 5: 1.603103
Train Accuracy: 0.452778
Test Accuracy: 0.275
Cost after epoch 10: 1.150919
Train Accuracy: 0.730556
Test Accuracy: 0.35
Cost after epoch 15: 0.601573
Train Accuracy: 0.861111
Test Accuracy: 0.35
Cost after epoch 20: 0.337171
Train Accuracy: 0.947222
Test Accuracy: 0.35
Cost after epoch 25: 0.119698
Train Accuracy: 0.988889
Test Accuracy: 0.3
```




```
Train Accuracy: 0.975
Test Accuracy: 0.375
```

The model has achieved 98% accuracy on the training set at epoch 25 but it is doing poorly on the test set. Therefore, the model is complex enough to handle this data set but the problem is over fitting.

In order to solve the problem of over fitting. I have four options:

1) Get more data:
   Sentiment analysis is relatively subjective(unlike classification which is objective), therefore, I cant perform usual data augmentation methods such as : mirroring, random cropping, rotation, shearing, local warping and color shifting since all of these critical factors of an image is influenced by the photo takers mood, which is correlated with his rating.
2) Regularization:
   The data set is too small to have dropout method, therefore, I will use L2 regularization:

$$\underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l}\sum_{k}\sum_{j}W_{k,j}^{[l]2}}_{\text{L2 regularization cost}}$$

3) Use better CNN model:
   I will build a model similar to that from the research paper on sentiment analysis on Flicker and Twitter Images.(You et al., 2015) [2]
4) Use smaller size of mini batch:
   Using a smaller batch might help the model better generalize to the training set.(Keskar et al., 2017) [3]

D. Performance of First Model with Regularization

Below is the performance after regularization of different parameter values:

Lambda	Mini Batch Size	Best Accuracy Score of Training	Epoch Time		Best Accuracy Score of Test Set	Accuracy Score of Training Set	Difference	Epoch Time
0	8	1.00	90		0.375	0.62	0.245	5
0	64	0.99	25		0.350	0.73	0.380	10
0	360	1.00	85		0.450	0.486	0.036	20
0.001	8	0.98	25		0.400	0.9111	0.511	10
0.001	64	1	30		0.400	0.461	0.061	5
0.001	360	0.98	100		0.450	0.48	0.030	15
0.005	360	0.98	100		0.400	0.338	-0.062	5
0.005	360	0.98	100		0.400	0.383	-0.017	10
0.01	360	0.98	95		0.425	0.461	0.036	15
0.05	64	0.89	30		0.350	0.56	0.210	10

lambda=0.01, mini-batch size = 360

First, smaller mini batch size doesn't help in my case. The reason might be that my data set is too small(¡1000) so have smaller mini-batch will not help solve the problem of regularization.

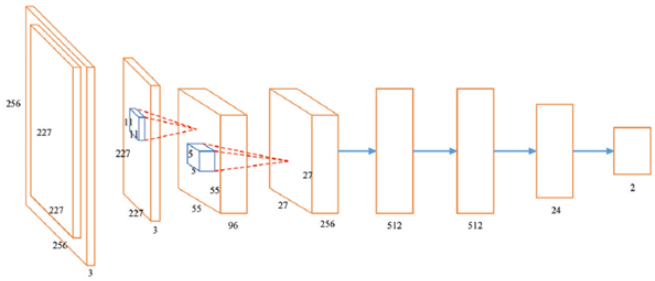
Second, by using my whole training set as a batch(when mini-batch size is 360), my model's performance on the test set has improved a lot, with the accuracy score increased from 0.375 to 0.450. The reason might be that using the whole training set instead of mini-batch makes the model robust to the noise, so it is generalizing better to the training set.

Third, by using lambda of 0.001 in L2 regularization, the model's accuracy score of test set is the same but the difference between that of the accuracy score of training set has decreased. This says that the regularization does help in my case.

Fourth, if I use larger lambda values, such as 0.01 or 0.05, the accuracy score of both training set and test set will hurt and the model is being "penalized" too harsh that it can't learn well on the training set, let alone perform well on the test set.

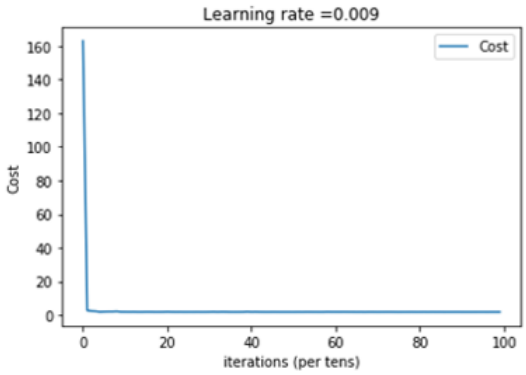
E. Second Model and Performance

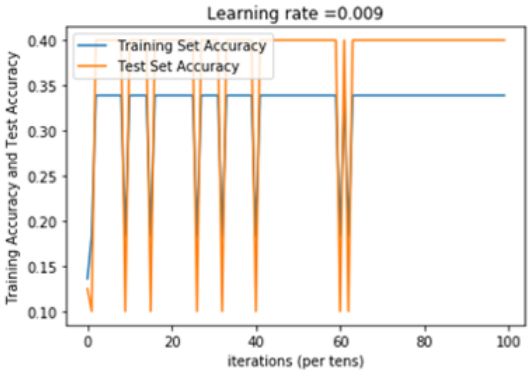
The CNN model of the research paper is as follows: [2]



The author didn't specify the max pooling layer's window size so I apply max pooling layers from my first model. Also, I added ReLU layer following each max pooling layers in the model.

Below is the performance of the model, with mini-batch size of 64 and 100 epochs.






```
Train Accuracy: 0.338889
Test Accuracy: 0.4
```

The second model is worse than the first model on the training set. I think the reason is that the second model has more layers than the first model, the vanishing gradient problem is stopping the cost to decrease to the local minimum. This is apparent in that the cost score stops decreasing from 1.85 since epoch 65.

```
Cost after epoch 0: 162.876251
Train Accuracy: 0.136111
Test Accuracy: 0.125
Cost after epoch 5: 2.015688
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 10: 1.915137
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 15: 1.908017
Train Accuracy: 0.183333
  ......
Cost after epoch 50: 1.844074
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 55: 1.857531
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 60: 1.891161
Train Accuracy: 0.183333
Test Accuracy: 0.1
Cost after epoch 65: 1.848648
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 70: 1.847348
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 75: 1.836810
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 80: 1.846296
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 85: 1.856467
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 90: 1.840904
Train Accuracy: 0.338889
Test Accuracy: 0.4
Cost after epoch 95: 1.846269
Train Accuracy: 0.338889
Test Accuracy: 0.4
```

The reason that the second model is not doing well on my data set might due to:

1) My data set has only 400 images, the Flicker training data that You uses has half a million images.
2) My data set is significantly different from the Flicker data which contains many other categories of images:

Therefore, I learn that using other CNN models could be helpful, but we need to take into consideration of dataset difference.

F. K-means Clustering and Principal Component Analysis

Although the best model could only achieve 0.45 accuracy score on the test set, I could still use the result of the last fully connected layer(1024 outputs) as input in conducing Kmeans Clustering and Principal component analysis.

I first just use the 1024 outputs in the K-means clustering(clusters = 5). However, the outcome is not satisfying.

After conducting Principal component analysis, I found that over 50% of the total variance in the 1024 features could be explained by 9 principal components.

	1024 Features	9 PCs
Completeness Score	0.066744	0.069952
Homogeneity Score	0.064897	0.066784

By using the 9 principal components instead of the 1024 features in clustering, the result has improved a little. Therefore, we know that the 9 principal components are useful in performing sentiment analysis.

III. Discussion of the Results

Although the best model's accuracy score is only 0.45, but it shows that there is possibility that we could use hotel images alone to predict ratings. Especially a simple model could achieve 100% accuracy on training set with only 360 images.

In the k-means clustering and Principal Component Analysis results, we know that over 50% of the total variance from the 1024 features could be represented by only 9 principal components, therefore, it is feasible to use the output from the last fully connected layer in other classification methods, such as LDA and SVM.

IV. Conclusion

I should continue the research with following improvements:

1) Get more labeled data:
   Use AMT(Amazon Mechanical Turk) to get more data.

2) Learn from other researches:
   Expedia has conducted similar research on ranking hotel images. They use 100,000 images and the MAPE on the test set is 12.5%. [4]
3) Explore following classic CNNs:
   LeNet will be a good learning model since it has similar structure of my first model - two conventional layers in the network. Also, I should study the AlexNet which is more complicated but has been proven successful in many research areas. Transfer learning is a good approach since training a neural network takes time and efforts, it might be better to use pre-trained weights to start with then fine tune the parameters.

REFERENCES

[1] Andrew Ng's CNN Model from Convolutional Neural Networks at Coursera

[2] Q. You, J. Luo, H. Lin, and J. Yang, Robust Image Sentiment Analysis Using Progressively Trained and Domain Transferred Deep Networks, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.

[3] N. S. Keskar, D. Mudigere, , J. Nocedal, , M. Smelyanskiy, , and P. T. P. Tang, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, ICLR 2017.

[4] https://news.developer.nvidia.com/expedia-ranking-hotel-images-with-deep-learning/