BINARY CLASSIFICATION OF YELP RESTAURANT IMAGES

Md. Navid Akbar & Nishith Burman

Northeastern University

ABSTRACT

The objective of this project is to classify images into binary classes using a convolution neural network. Next, the performance of our model is compared with that of an industry benchmark model, by employing transfer learning. Using a five-fold cross validation, the mean accuracy of our model is 76.4%, while the accuracy of the pretrained model was 87.0%. Our proposed model, however, had less number of parameters and was less complex. Thus, in a performance versus complexity analysis using both the Bayesian and the Akaike Information Criteria, our model scored better than Google's Inception V3.

1. INTRODUCTION

Image classification has become increasingly important due to its wide application within the industry. From object detection to multi-class labeling to mapping spatial data it is now being used almost everywhere. Humans love to share their experiences with others. With the dawn of the digital age, we are more frequently sharing our experiences in social media. Each and every minute thousand of people are sharing their experience in social media. Yelp is one of those websites where people share their experiences by uploading pictures of their activities. Now, these Images contain valuable information not only for companies but also for the customers. For instance, with the help of the uploaded images, a customer can roughly get an idea of the place they are visiting or an item they are purchasing.

For reasons such as this, it is quite essential that an option exists in the reviews of the restaurant on the Yelp page that groups images by such features. This would help users find what they need much more convenient manner.

We are using the convolution neural network (CNN) for this image classification task. The entire Yelp dataset on Kaggle has 200,000 images, belonging to six different classes. Among those, we chose about 8000 images, belonging to the binary categories: food and drinks. We split this data, and trained and tested on it using a five-fold cross validation. We also wanted to check how our model is performing against the pretrained model and observe the tradeoff between model complexity and model performance.

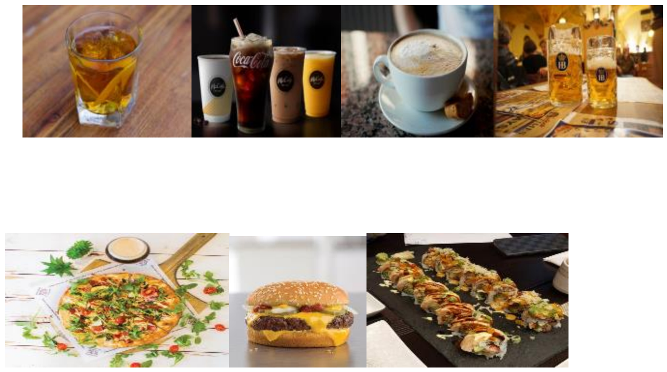

Figure 1. Images of Food and Drinks shared by users on Yelp

2. PROPOSED WORK

2.1. PREVIOUS WORK

A lot of work has been carried out for this classification and a no. of models have been built for this task previously. However, this task For instance, the winner of the Kaggle competition Dimitri Tsybuleyskii used a combination of pretrained weight, embedded spacing, Binary relevance and Ensemble of classifier chain. He achieved a mean F-Score of 0.8318, testing on all six classes of image. Similarly, the second-place winner Thuyen Ngo used pre-trained weights to train his model. In combination with the pre-trained weights, he used Nesterov's accelerated SGD for weight optimization and cross-entropy as the loss function. Thuyen achieved a mean F-Score of 0.8317, testing on all six classes of image. But the major difference between both these model is Dimitri used pre-trained weight from Google,s Inception V-3 model [4], on the other hand, Thuyen used pretrained weights from Facebook's ResNet[5].

2.2. OUR APPROACH

The problem that we addressed through this project is the performance issues of an image classification model based on the no. of model parameter. We also wanted to study the tradeoff between model complexity and model performance in image classification tasks. In order to study this relationship, we built two convolution neural networks. In the first network, we used pretrained weights of the inception network devised by GoogLeNet. The inception model has a large number of parameters approximately 13,000,000 in our case and it is very effective in approximating sparse CNN with normal dense construction. In our second model, we used normal method of training CNN. Both of these models are fundamentally similar, the primary difference resting in their complexity, or number of parameters.

In Figure 2. Below, we see the visualization of the intermediate layers. The visualization of different layers of the network has been included in order to explain the inner working of the models. We can observe the effect of different filters on the image depicting a burger. In the primary layers, only lines and edges are detected. As we go deeper into the convolution layers, we observe higher level features being captured by the different kernels.

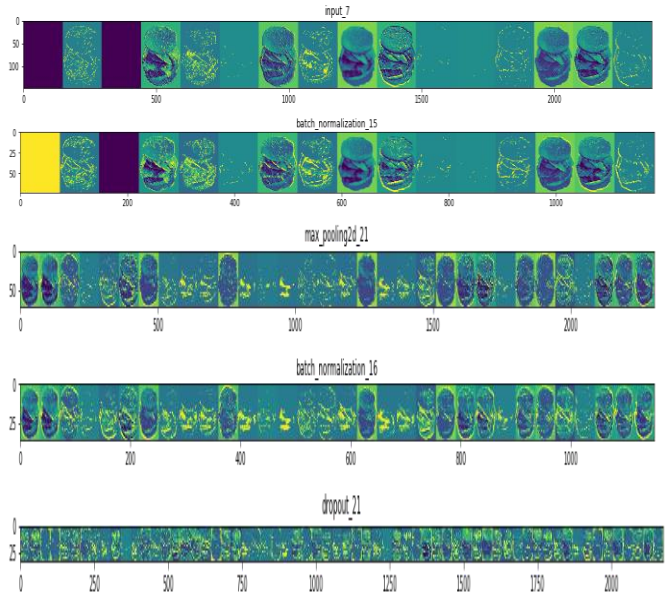

Figure 2. Visualizing intermediate layers of the network

3. MODEL FRAMEWORKS

3.1. CNN with Batch Normalization and Early stopping

This model is a simple deep net model with 3 convolution layers and one fully connected dense layer. The model is trained with 5700 images, keeping about 600 images for validation. This validation is chosen at random from the train set, and is subsequently used for purposes of early stopping, reducing learning rate on plateau and saving the best model. In early stopping, we waited for 4 epochs, if the validation accuracy did not improve by 0.1%. In this duration, we also decreased the learning rate by a factor of 0.75, after waiting for the first two epochs. Finally, we only saved a model when the validation accuracy improved. After the training was complete, we loaded that best model, and subsequently evaluated on the test set.

The trained model is then tested on approximately 1700 images. For updating the weights of the networks, RMSprop optimizer has been used with a learning rate of 0.001, and with ρ=0.9. Binary Cross Entropy has been used as the loss function for this model. Finally, to tackle the problem of overfitting, we introduced dropout after each and every layer. After the convolution layers, the dropout rate was 0.33. After the dense layer, the dropout rate was 0.4. Also, we used a combination 32, 64 and 128 numbers of filters for the three convolution layers, successively. We used a width of 64 nodes for the fully connected dense layer. The total number of parameter for this CNN model was nearly 498,977.

Now before training our data set, we resized the dimensions of the images to 150*150*3. We also used a min-max scaling, so that the pixel values of the images are between 0 and 1. Also, for both training and testing our model, we used a batch size of 64 images. While working with these above combinations we found out that our model was still overfitting after nearly about 10 epochs. So we incorporated the early stopping method to tackle the issue of overfitting.

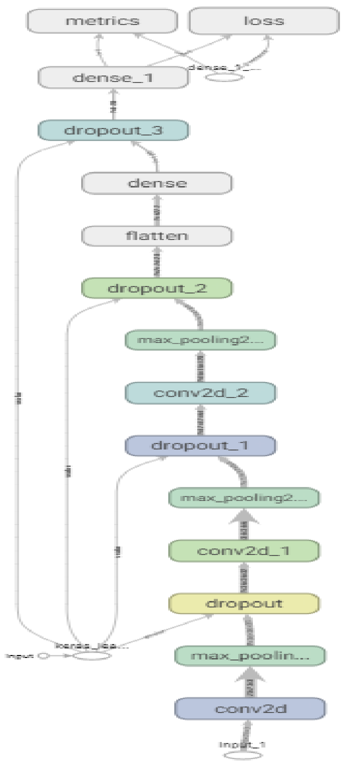

Figure 3. Tensorboard visualization of our model.

3.2. CNN with pre-trained weights from Inception Model

Structure wise this model is quite similar to model described in the section 3.1. Similar to model used in 3.1., it has a combination of convolution layer and dense layer. However, for this model we are using pre-trained weights from the inception model and the model is far more complex than our previous model. It has nearly 13,000,000 parameter. While implementing this model we again faced the issue of overfitting as soon as 5th epoch. So we had to again apply early stopping.

Figure. 4 is depicting just a small component of this network. One can clearly infer the complexity of this model by looking at the small fraction of this model.

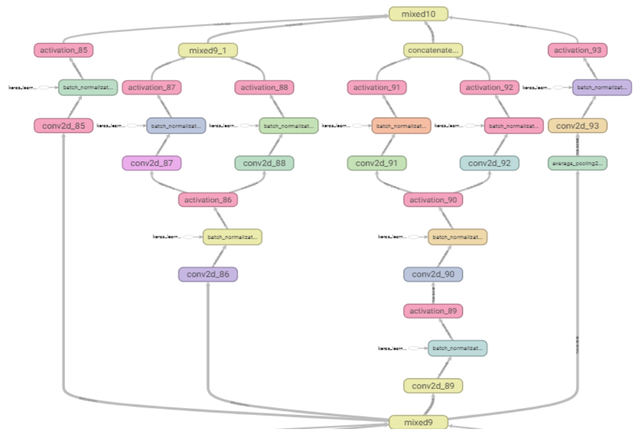

Figure 4. Partial structure of Inception v3.

4. RESULTS

4.1 MODEL EVALUATION

Out of all the available model evaluation parameters we choosed accuracy as the metrics for the model performance. Figure 5. belows the accuracy of the CNN model for the training and the validation set, for one of the folds. Figure. 6 below shows the losses of the training and test sets.

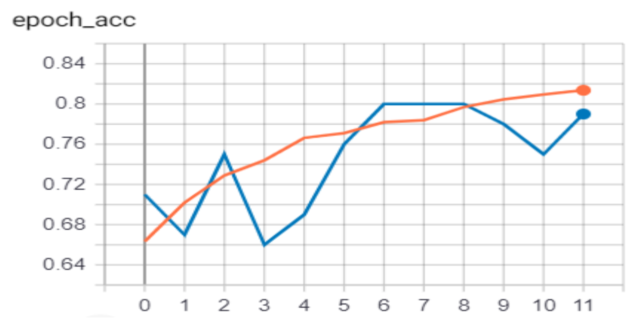

Figure 5. Plot of Training and validation accuracy.

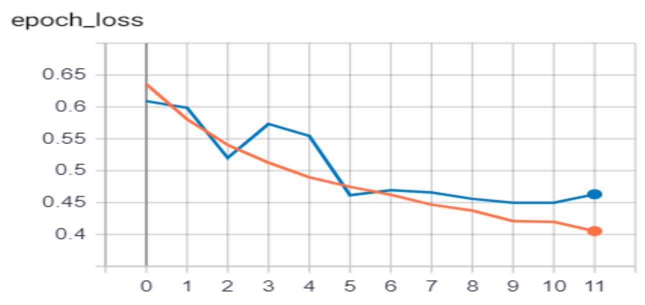

Figure 6. Plot of Training and Test Losses

Similarly, Figure 6 and Figure 7 are indicating the accuracy and loss of training and validation sets for the model with pre-trained weights (described in 3.2), in one out of the five folds. It is evident from figure 7 that this model is achieving the maximum training accuracy of nearly 94 percent. We also see the model is getting overfitted after 3rd epoch only and early stopping is required to tackle this issue.

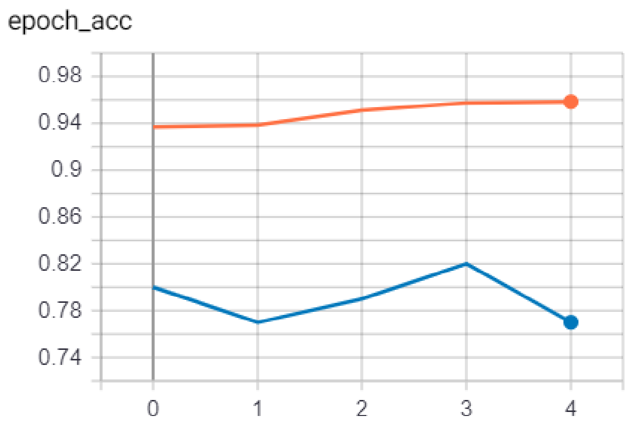

Figure 7. Plot of Training and Test Accuray (pre-trained weights)

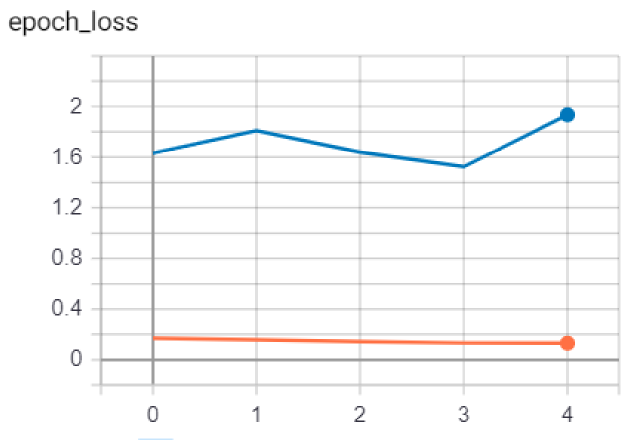

Figure 8. Plot of Training and Test losses (pre-trained weights)

We also implemented 5-fold cross validation to check if the accuracy that we are getting are consistent with test sets with different bunch of images. After implementing 5-fold cross validation, we found out that the average accuracy was 76.3% with a standard deviation of 2.45%. For the Inception model, this mean accuracy was 87.0%, and the standard deviation is 2.86%. This result is shown in the error bar plot of Figure 9.

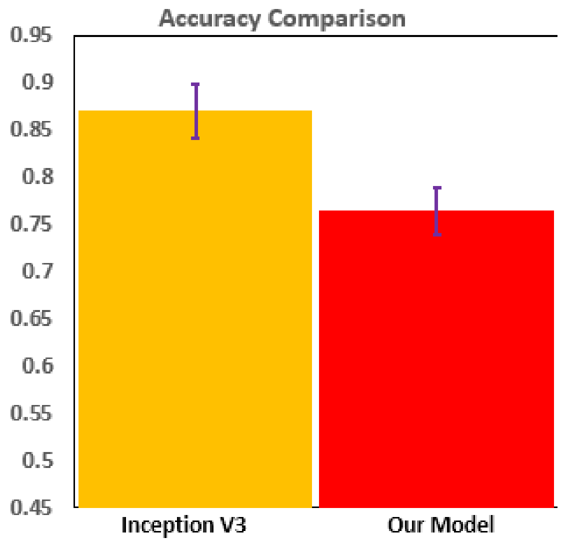

Figure 9: Error bar plots for the mean accuracies of both models.

4.2 PERFORMANCE VS. COMPLEXITY TRADEOFF

CNNs are generally over-parameterized. However, it is often essential to know how the model performance fares against its complexity. Thus, we present the Bayesian Information Criteria (BIC) and the Akaike Information Criterian scores for both the model, using the following relations obtained from [7]:

$$\mathrm{BIC} = N \ln \hat{\sigma}^2 + p \ln N$$

$$\mathrm{AICc} = N \ln \hat{\sigma}^2 + \frac{N(N+p)}{N-p-2},$$

where N is the total number of training images (5700) multiplied by number of classes (2), p is the number of model parameters, and σ^2 is the mean squared training error per image. The results are displayed in Table 1. Recalling that the lower the score, the better the model, our model outshines Inception v3 in terms of performance to complexity ratio.

	BIC	AIC
Our Model	4,628,177	965,003
Inception v3	106,305,931	22,732,958

Table 1: BIC and AIC values for both the models.

5. CONCLUSION

On studying the above two models we found out theie is an obvious tradeoff between between model complexity and model performance. The model that we built was far more simple than the model in which we used pre-trained weights. Hence, training our model was speedier and easier, as there were far less number of model parameters than the model with pre-trained weights. However, in terms of raw accuracy, the model with pre-trained weights was performing better than our model. In the end, we demonstrated how our model is a better choice when taking the BIC and AIC scores into account, indicating it has a better performance to complexity ratio.

6. REFERENCES

[1.] Yunpeng Li, David Crandall, and Daniel Huttenlocher “Landmark Classification in Largescale Image Collections. 2009, IEEE. 1957-1964”.

[2.] Li, F. (2017, January 5). Convolutional Neural Networks (CNNs/ConvNets).http://cs231n.github.io/convolutionalnetworks/.

[3.] Rifkin, R. (2008, February 25). Multiclass Classification. Retrieved November 20, 2017 from http://www.mit.edu/ 9.520/spring09/Classes/multiclass.pdf

[4.] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D.&amp; Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[5.] He, K., Zhang, X., Ren, S., &amp; Sun, J. (2016). Deep residual learning for image recognition.In Proceedings of the IEEE conference on computer vision and pattern recognition.

[6.] Matthew D Zeiler, Rob Fergus “Visualizing and Understanding Convolution Neural Networks” 12 Nov 2013

[7] Stoica, Petre, and Yngve Selen. "Model-order selection: a review of information criterion rules." IEEE Signal Processing Magazine 21.4 (2004): 36-47.