Capstone Proposal

I-Chun Liu
September 6th, 2018

Proposal

Domain Background

The ability to understand the context of an image is extremely useful. For instance, it could be used to reduce and optimize search results for pictures, adding tags automatically to crowdsourced photos on various social platforms, and allowing images to have "words." However, to understand the context of an image, the program must understand the nuances or the "depth" of the image. In Simonyan and Zisserman's paper, they used convolutional neural networks to perform large-scale, multi-label image classifications and concluded that deep CNNs have superior performance than the ones that had fewer convolutional layers [1]. I think it is an exciting problem with many applicable uses for a system that could perform well in multi-label image classification tasks in the real world.

Problem Statement

Yelp held a photo classification competition on Kaggle two years ago. It asked Kagglers to build a model that automatically tags user-uploaded photos with multiple labels, nine labels to be exact. In this capstone project, I would be working on designing and building a Convolutional Neural Network to try to achieve or better the highest benchmark score. The goal of this project is to assign a set of labels to each photo correctly.

Datasets and Inputs

The labels provided by Yelp are the following:

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: ambience_is_classy

8: good_for_kids

The datasets can be found at https://www.kaggle.com/c/yelp-restaurant-photo-classification/data

The datasets consist of the following files:

sample_submission.csv.tgz

test_photo_to_biz.csv.tgz

test_photos.tgz

train.csv.tgz

train_photo_to_biz_ids.csv.tgz

train_photos.tgz

sample_submission.csv.tgz shows how the submission file is supposed to look like. There are two columns in this CSV file. The first column contains all the business IDs, and the second column contains each business ID's corresponding labels in digits from 0 to 8. If there are multiple labels in a business ID, they are separated by spaces.

test_photo_to_biz.csv.tgz and **train_photo_to_biz_ids.csv.tgz** contain the mapping from each photo to its associated business ID. In other words, this mapping allows a business to have more than one pictures, and we're able to tell which photos belong to which store.

test_photos.tgz and **train_photos.tgz** contain the actual photos. Photo ID is in each photo's file name. All the images are user-uploaded, meaning they are non-uniform in sizes and color images.

train.csv.tgz contains each business ID with its associated correct/truth labels. There are 2000 distinct businesses.

Note: The test datasets do not have each business' truth labels. Since only the training data is provided with labels, I plan to randomly divide 25% of the training data into the validation set and 75% into the training set. To obtain the final accuracy for the test dataset, I will submit my test prediction results to Kaggle.

Solution Statement

Since this task requires understanding the depth of images, a Convolutional Neural Network would be appropriate for this task. In addition, according to a Yelp blog post, Yelp also utilized convolutional neural networks to classify business photos [2]. The training and test photos compose of various kinds of images, ranging from food, storefronts, selfies, menus, restaurant interior, company logos, etc. Furthermore, because of the varieties of photos, pre-trained convolutional neural networks may be a great candidate for this task.

Benchmark Model

In 2015, Yelp published an introductory blog post on this challenge, where it went over the metric that is used to assess the performance of a model and the performances of two benchmark models. The evaluation metric for this challenge is the mean F1 score, which will be discussed thoroughly in the Evaluation Metrics section.

Furthermore, the first benchmark model is based on a random guesser, where each label has equal probability. This model results in a score of 0.4347. The second benchmark model "calculates the color distribution of all the images of a test business, compares that to the average color distribution of businesses with positive attribute values and negative attribute values respectively, and assigns the value with a more similar color distribution to the test business." This model achieves a score of 0.6459. My goal for this project is to come up with a model that scores higher than the benchmark model that utilizes color distribution [3].

Evaluation Metrics

The F1 score measures the accuracy by using precision p and recall r. Precision is the ratio of the number of true positives (tp) to the total number of elements being classified as positive (tp + fp). Recall is the ratio of the number of true positives to the number of elements that should be classified as positive **(**tp + fn). Since this task has multi-label, we would compute the mean of F1 score, which is the weighted average of the F1 score of each class label. The following is the formula for the F1-score:

$$F1 = 2\frac{p \times r}{p + r} \; where\, p = \frac{tp}{tp + fp} \; and\, r = \frac{tp}{tp + fn}$$

The F1 score gives equal weights to recall and precision. To have a high F1 score, one must have both high recall and precision scores. Having a low recall or a low precision score would result in a low F1 score [4].

Project Design

1. Load datasets into data frames.
2. Pre-processing:
    a. For each business' training labels, I will perform data manipulations to have a vector of 9 elements, where if a label is True it would have a 1 at that corresponding index. Currently, the CSV file only contains the digits of the labels. For instance, business ID 1001 has labels "0 1 6 8." After data manipulations, it would have [1 1 0 0 0 0 1 0 1].
    b. Normalize images, so their RGB values are between 0 and 1.
    c. Re-size images, so their sizes are uniform.
    d. Perform image augmentations: rescale, rotate, and zoom.
3. Research pre-trained models (Xception, VGG19, ResNet50, NASNet) and try to connect them to a fully connected layer and compare their performances.
4. Research and perform feature extractions.
5. Tune models:
    a. Try adding more layers or removing layers.
    b. Try various values for the hyper-parameters, including learning rate, number of hidden nodes, number of filters, batch size, activation functions, number of epochs, etc.
    c. Try freezing weights in the pre-trained networks.
    d. Try adding more or removing max-pooling layers and dropout layers.
6. Compare each model’s performance and decide which model to use.

7. Repeat steps 4 to 6 if necessary.

References

[1] Karen Simonyan, Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR 2015.

[2] Wei-Hong C. How We Use Deep Learning to Classify Business Photos at Yelp. Yelp, October 2015. https://engineeringblog.yelp.com/2015/10/how-we-use-deep-learning-to-classify-business-photos-at-yelp.html

[3] Daniel Y. Introducing the Yelp Restaurant Photo Classification Challenge. Yelp, December 2015. https://engineeringblog.yelp.com/2015/12/yelp-restaurant-photo-classification-kaggle.html

[4] Yelp Restaurant Photo Classification. Kaggle. https://www.kaggle.com/c/yelp-restaurant-photo-classification#evaluation