Monday, May 7, 2018

Henok Mengistu

#### In this practicum challenge I was asked to develop a classifier that classifies if an image is a business or it is something else. Because the dataset from this challenge is very small (only 3119 images), it is not a good idea to train a classifier from scratch. Thus, I approached the challenge through transfer learning technique. Transfer learning is a method in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. I used a pre-trained model that was trained on places database and imagenet. I grabbed this pre-trained model from this link: https://github.com/GKalliatakis/Keras-VGG16-places365. The reason I choose this data set was because it has similar features with my data set which also involves places such as restaurants, houses, roads etc.

#### My approach involves extracting features from the earlier layer of the pre-trained model and train a classifier on top of those features. Here are the steps:

* First extract features by running all images through the pre-trained model which is without the fully-connected layer. The features are the output of the last conv layer of the pre-trained model. The extracted features are then saved and used to train a 3 layer fully-connected neural network which works as a classifier.
* overall, the approach is to represent each image into a feature space and use that feature space to train a classifier
* The first attempt of this method was without data augmentation
* Max classification accuracy was 80.4% and 76.1% with data augmentation. I thought augmentation would help but it didn’t
* Running instruction
  1. First run feature\_extraction.ipynb. This script has an option specify whether to do data augmentation or not. This python scrip also implements the data preparation task asked in the practicum
  2. Then run train\_features.ipynb. This script implements the modeling part of the challenge

Both of these python notebooks are commented to further explain their inner workings

Deployment: I choose to deploy the trained classifier through Flask.

Steps

* Install Flask web framework
* Run deploy.ipynb
* In the command line run this command: curl -X POST -F image=@farm.jpeg <http://localhost:5000/predict>

This command will make a post request passing farm.jpeg

* The output is the predicted class and the probability of the class assignment

With this submission, I also included plots that describe the training behavior for both with and without data augmentation.

Inside the plots directory run plot\_results.ipynb

Discussion

I expected that data augmentation would help increase the classification accuracy, but it didn’t. I think the reason is that most of the original images are so similar to each other that data augmentation couldn’t inject the required data variation that would increase accuracy.

The plots show that validation accuracy fluctuates throughout the training period. I think the reason is that the validation set is very small. However, more investigation is needed to justify this assertion.

Overall, I must admit that 80.4% classification accuracy is not that much good result but given the fact that there is small data set I would say it is not too bad. With more data set I expect the accuracy to increase.

Furthermore, I noticed that some of restaurant images do not look as such. Take a look at the below images that are labeled as restaurants/ retail. For me they don’t look like restaurants/retail. I mentioned this out of curiosity.



I also noticed class label inconsistency. such as restaurant/retailvs restaurnt/retail. But that is a quick fix.

Further works

I see a couple of approaches that might improve the classification accuracy. At least I would start by trying out these two follow-ups

* Have more data. The more data, the clearer the pattern will be. Meaning, more data set allows to best expose the structure of the underlying problem to the learning algorithms
* Train two or more models that uses different internal representation of the problem and use the technique called Staking to best combine their predictions. Stacking is an ensemble technique that uses a new model to learn how to best combine the predictions from two or more models trained on dataset.

Dependencies: Have the following libraries installed to successfully replicate my results

* Python 3
* keras2.1.1
* Flask 0.12.2
* Pandas0.20.3
* Matplotlib.pyplot