Machine Learning Engineer Nanodegree

Capstone Proposal

Benjamin Rouillé d'Orfeuil

January 18, 2018

Domain Background

Convolutional Neural Networks (CNNs) have successfully been applied in the field of image recognition. These algorithms have proven to be incredibly efficient at classifying images and often outperforms other machine learning algorithms at this task. To illustrate, very accurate predictions can be achieved on the well known MNIST database of handwritten digits[1] and the CIFAR-10 dataset[2] using very simple network architectures. These datasets, however, are relatively simple. Not only the number of classes is very low (10 digits for the MNIST database and 10 object categories for the CIFAR-10 dataset) but each class is very different from one another.

Image recognition on real world images, on the other hand, requires the building of complex models. The deep CNNs developed for the ImageNet Large Scale Visual Recognition Competition[3] (ILSRVC) are a perfect illustration. Unlike the MNIST and the CIFAR-10 databases, the ILSRVC dataset has a large number of classes and the difference between some of the object categories can be very tenuous. The algorithm needs to be able to discriminate between different breeds of dog or types of snake. It is hence not surprising that the ResNet50 model[4] developed by Microsoft and that won the 2015 ILSRVC edition has 50 convolutional layers and a total of 168 layers.

It is common practice today to use pre-trained state-of-the-art models to classify photographs. CNNs that have been pre-trained on a large and diverse dataset like ImageNet captures universal features in its early layers that are relevant and useful to most classification problems. The weights of the pre-trained CNNs can then be fine-tuned by continuing training it on the dataset under study.

Problem Statement

Yelp is a social networking site that publishes crowd-sourced reviews about local businesses. About two years ago, Kaggle hosted the Yelp Restaurant Photo Classification challenge (see `https://www.kaggle.com/c/yelp-restaurant-photo-classification`). Labels are optional during the review submission process, some restaurants can be left uncategorized. For this reason, Yelp asked contestants to build an algorithm that automatically predict attributes for restaurants using their user-submitted photographs. The goal of this project is to develop such a model.

[1] The MNIST database is available at `http://yann.lecun.com/exdb/mnist/`. A quick analysis of this dataset can be found here.

[2] The CIFAR-10 dataset can be found at the following url: `https://www.cs.toronto.edu/~kriz/cifar.html`. Predictions on this dataset are presented here.

[3] The 2017 challenge is described on the ImageNet website: `http://image-net.org/challenges/LSVRC/2017/index` and on Kaggle: `https://www.kaggle.com/c/imagenet-object-localization-challenge`.

[4] The Microsoft team presents their model in the following paper: `https://arxiv.org/pdf/1512.03385.pdf`.

Datasets and Inputs

The photographs and attributes for this project can be found in the data section of the Kaggle competition webpage: `https://www.kaggle.com/c/yelp-restaurant-photo-classification/data`. Yelp provides for this competition a training dataset (234 842 photographs) and a test dataset (1 190 225 photographs). Each image is mapped to a business identification number. There is a total of 2000 businesses that can be tagged with 9 different attributes. The labels are listed below:

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

It is worth noting that the dataset are quite large. Both the compressed archive training and testing files are about 7 GB.

Solution Statement

A convolutional deep learning network will be built to tag the restaurants. For this purpose, the Keras[5] Python library will be used with TensorFlow[6] as a backend. The CNN will not be built from scratch. Instead, a pre-trained model will be used either as an initialization or a fixed feature extractor. Some popular models for image classification along with their weights trained on ImageNet are available in Keras. Each model available will be considered and thoroughly evaluated.

The model will be trained on the training dataset and its accuracy will be evaluated on the test dataset. A simple F1-score will be used to evaluate the performance of the algorithm. This evaluation metric will also allow for comparison with models from other contestants.

Benchmark Model

Contestants submit a set of predictions and their model is scored using the predefined evaluation metric. As one can see on `https://www.kaggle.com/c/yelp-restaurant-photo-classification/leaderboard`, the best model has a score of 83.177 % (as reported on the private leaderboard[7]) and models developed by the top 144 ranked contestants are all above 75 %. The goal for this project is to develop a deep learning network that reach a performance greater than 75%.

[5] See `https://keras.io/`.

[6] See `https://www.tensorflow.org/`.

[7] The private leaderboard remains secret until the end of the competition and determines the final competition winners. The purpose of this division is to prevent people from winning by overfitting to the public leaderboard. The public and private leaderboards enclose approximately 30 % and 70 % of the test data, respectively.

Evaluation Metric

As mentioned previously, the evaluation metric will be the F1-score, i.e., the harmonic mean between precision and recall:

$$F1 = 2\frac{p \cdot r}{p + r} \text{ where p} = \frac{\text{tp}}{\text{tp} + \text{fp}} \text{ and r} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

This means that a good performance on both precision and recall will be favored over extremely good performance on one and poor performance on the other.

Project Design

Amazon Web Services (AWS) will be used to conduct this project. The data will be first stored in an Amazon S3 (Simple Storage Service) public bucket and the analysis will then be carried out on an Amazon EC2 (Elastic Compute Cloud) instance using an AMI (Amazon Machine Image) that has all the pre-configured environments for building deep learning applications.

An exploratory data analysis will be first performed. The idea is to get familiar with the dataset and summarize its main characteristics. This would also be a good opportunity to create a validation dataset and, if necessary, pre-process the images that will be later fed to the deep learning model. The results will be presented with a Jupyter Notebook.

The Keras library will be used as a deep learning framework and transfer learning techniques will be applied to build the model. As previously mentioned, powerful image classification models can be built by using a network that has been previously trained on a large dataset. Such a network would have already learned features that are useful for most computer vision problems, and, for this reason, leveraging such features would allow us to reach a better accuracy than any method that would only rely on the available data

State-of-the-art deep learning models such as Xception, VGG16, ResNet50 and InceptionV3 are made available in Keras. These models have achieved excellent results in the ILSRVC. To illustrate, the Xception model reaches 79% (top-1) and 94.5% (top-5) accuracy on the ImageNet validation dataset. Note that for each of these models, Keras not only provides the network architecture but also the associated weights trained on ImageNet.

In order to reduce training time without sacrificing accuracy, the bottleneck features of these models, i.e., the last activation maps before the fully connected layers, will be first calculated and saved. A small fully connected model will be then trained on top of the stored features. At this point of the analysis, it will possible to assess which model performs the best and make an educated decision.

To further improve the performance of the algorithm, we will try to fine tune the last convolutional blocks of the chosen pre-trained deep learning model alongside the top-level classifier. Finally, the small CNN could also be optimized. For instance, more aggressive dropout layer could help to reduce overfitting.