Machine Learning Engineer Nanodegree

Capstone Proposal

Luis Cunha
November 7th, 2018

Proposal

Domain Background

Flower, as well as living organism classification is an important task used both for scientific purposes (taxonomy in research or applied sciences, such as agriculture) and leisure. It has traditionally been cumbersome, requiring a certain knowledge level, as well as careful and time consuming analysis of anatomical characteristics. Not long ago the knowledge was kept in books, requiring either carrying these to the field, or collecting specimens. More recently, this knowledge could be more easily transported to the field in a computer, or even a smartphone. However, the process still remained the same as centuries ago, in which a person had to compare physical characteristics of the subject with the knowledge stored. This process could be greatly improved by automation, delegating the classification task to a computer. Image classification is a common and important application of Machine Learning (ML). It is used on a wide range of applications, from individual identification for security proposes, to identifying road signs in the live video feed of self-driving cars. I will develop a ML model to classify flowers from images, while evaluating performance and cost (time and money) tradeoffs.

Problem Statement

A flower classification needs to be built, which could be used as the basis of a flower classification system (such as an APP, or a REST API) by directly uploading an image and receiving the flower identification. The choices to be made to train such a model are numerous, from the datasets, to the training techniques, to the hardware choice. Deciding to train a model is just the first step. Many decisions must be made to balance the desired accuracy with the cost of training. With unlimited time and budget, one could collect a rich, high-quality, balanced dataset of flower images and fully train a neural network. However, that's seldom the case. Collecting images laborious and expensive. Fortunately, several flower datasets of different size and nature are available online. I've chosen to classify images typically found in the UK, using a dataset a dataset provided by the Oxford University consisting of images of flowers from 102 distinct plant species. There exist other, larger datasets available, but training become costly on a cloud provider. I will explore the process of building a model to achieve the best accuracy (% if correct perditions in a test dataset), and training cost (time and money) tradeoffs.

Datasets and Inputs

There exist several datasets of images of flowers. The Kaggle 2018 FGCVx flower classification challenge dataset includes 1000 species and over 600000 images. Preliminary attempts to use this dataset shows it to be cost prohibitive to train on AWS (e.g. accuracy on a modern CNN architecture was only 70% after 2 days of training, for a cost of ~$50). For someone with their own GPU hardware , such as a consumer nvidia 1080 or 2080 graphics card, this would be a fine dataset. However, I'll focus

on the needs of cloud training and its associated cost. The university of Oxford provides 2 smaller datasets, one of 17 species common in the UK, and a larger dataset of 102 species, with 40-258 images per category. I will use the latter. The species labels are not provided by Oxford (only integer coded categories), but the true labels were deduced by Github user m-co, and will be used. Usage of runtime data augmentation will be evaluated.

The oxford102 category flower dataset can be downloaded from:
http://www.robots.ox.ac.uk/~vgg/data/flowers/102/102flowers.tgz

The original image labels can be downloaded from:
http://www.robots.ox.ac.uk/~vgg/data/flowers/102/imagelabels.mat

Actual flower class labels can be downloaded from:
https://raw.githubusercontent.com/jimgoo/caffe-oxford102/master/class_labels.py

Solution Statement

A deep learning approach will be taken to predict the flow species in an image, using tensorflow and keras frameworks, and the python programming language. Different architectures will be evaluated. A baseline model will be built on a simple CNN. Challenger models will use transfer learning with several imagenet (http://www.image-net.org/) architectures. I will compare the performance (accuracy) vs training speed and monetary cost on a cloud providers (aws) for different CNN architectures (e.g. VGG, InceptionV3, resnet50, inceptionResMetV2, Xception, MobileNet, MobileNetV2, DenseNet, NasNet). Hyper parameter optimization will be performed for network parameters such as optimizer choice, and the value of using image augmentation techniques. I'll also evaluate training only bottleneck layers vs additionally fine tuning the n top layers (the top layers identify more granular features than the bottom layers, which identify fine features such as lines and edges). Finally, I will evaluate the time and cost on training on P2 vs P3 instances on AWS. P3 are advertised as more powerful, but are more costly. I'll evaluate the time to cost ratio of P2 vs P3.

Benchmark Model

As mentioned, a simple neural network consisting of a few convolutional layers and one or a few dense layers (without or without dropout/pooling) will be used as the model benchmark. Challenger models will be built to outperform this benchmark model.

Evaluation Metrics

The evaluation metric to be used both with the benchmark and challenger models will be the categorical accuracy, which reflects the mean accuracy rate across all predictions.

Project Design

The following steps will be taken:

1. Download data
2. Visualize data - inspect a few random images per category
3. Split dataset into named directories for convenient feeding into keras using flow_from_directory function
4. Data will be split into train, validation, and test datasets. Note that train and validation can be hardcoded into separate physical distinct datasets on disk, or the split can be performed at runtime by keras

5. Model evaluation and hyperparameter optimization, using transfer learning (with training of only the bottleneck features)
    a. Models:
        i. Benchmark model (a simple CNN)
        ii. Challenger models based on Imagenet models' architectures
    b. Parameters:
        i. Data augmentations (yes / no)
        ii. Optimizer choice
        iii. Optimizer specific hyperparameters
    c. Fine tuning (train n top layers of the network)
6. Evaluate speed vs cost of P2 vs p3 AWS instances
7. Evaluate performance on the testing dataset