Capstone Project Proposal

Machine Learning Engineer Nanodegree

Jeffen Chen

Domain Background

Compared with text, images can provide more vivid, easy to understand and more artistic information, which is an important source for people to transfer and exchange information. Image classification is to distinguish different types of images according to the semantic information of images. It is an important topic in computer vision, and also the basis of other high-level visual tasks such as image detection, image segmentation, object tracking, and behaviour analysis.

Problem Statement

An image classification algorithm will be created by using CNN in this project. There are many examples of machine learning being used to solve computer vision problems, and it's not hard to find applications that incorporated CIFAR-10 and CIFAR-100 as the dataset. In this project, CIFAR-100 will be used and because each image in the dataset is well labeled, it will be easy to measure whether the result it fit our expectation or not.

Dataset and Inputs

I decided to choose CIFAR-100 dataset after research. This dataset is publicly available at University of Toronto website and it has images in 100 classes containing 600 images each.

There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses.

50000 labelled examples are provided for training, with a further 10000 unlabelled examples used for testing. Each images has 3 RGB colour channels and pixel dimensions 32×32 for an overall size per input of 3×32×32=3072.

Solution Statement

The solution to this project will be training a CNN using transfer learning. There are several pre-computed networks that available on Keras:

- VGG-19
- ResNet-50
- Inception
- Xception

A final CNN architecture will be built upon one of the four networks. To improve the accuracy, image augmentation will be introduced as well.

Benchmark Model

The benchmark model will be a simple CNN consist 3 fully-connected layers and a dense layer that shown in the figure below. This will be relatively simple model to train and probably will get a very low score. Based on the leaderboard in CIFAR-100 competition on Kaggle, I am expect my transfer learning model to get a score that higher than 0.50.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               8645
=================================================================
Total params: 19,189.0
Trainable params: 19,189.0
Non-trainable params: 0.0
_________________________________________________________________
```

Evaluation Metrics

The evaluation metric for this competition is multi-class classification accuracy. For example, the proportion of true class labels correctly predicted.

Project Design

This project will use Python 3.x and Keras with tensorflow backend. For better training efficiency, our model will be trained on AWS. I intend to go through the following steps after successfully established my workspace:

- Normalize input data
- One-hot encoding all the tags
- Construct training set and validation set by ratio 0.8
- Artificially add noise to our raw data such as crop, flip or adjust hue, contrast and saturation.
- Using pre-computed networks, different fully-connected layer and configurations to improve the architecture performance.

- After the model is successfully trained, we can extract the weights and use that to make prediction about any uploaded image.

References

1. Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.
2. Udacity Machine Learning Nanodegree Dog Project
3. Keras Official Documentation