Machine Learning Engineer Nanodegree

Capstone Proposal

Puneet Sharma

November 26th, 2017

Proposal

Domain Background

Object Recognition in Images has been an area of extensive research for a long time. During the last decades, a large number of algorithms have been proposed. The rapid development of computer hardware has enabled the usage of automatic object recognition in more and more applications ranging from industrial image processing to medical applications as well as tasks triggered by the widespread use of the internet, e.g., retrieval of images from the web which are similar to a query image. Alone the mere enumeration of these areas of application shows clearly that each of these tasks has its specific requirements, and, consequently, they cannot be tackled appropriately by a single general-purpose algorithm. Here is the link of CIFAR-10 dataset on which I will perform ml algorithms - https://www.cs.toronto.edu/~kriz/cifar.html . And here is link where machine learning was applied to this type of problem - https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf .

Problem Statement

An image recognition algorithm (an image classifier) takes an image as input and outputs what the image contains. In other words, the output is a class label (e.g. “human”, “dog”, “car” etc.). Well, we have to train the algorithm to learn the differences between different classes. If we want to find dogs in images, we need to train an image recognition algorithm with thousands of images of dogs and thousands of images of backgrounds that do not contain dogs. Needless to say, this algorithm can only understand objects / classes it has learned.

Datasets and Inputs

In this project we will be using **The CIFAR-10 dataset** for our problem. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks. There are 10 different label classes for images in the dataset. Here is link of dataset - https://www.cs.toronto.edu/~kriz/cifar.html .

Solution Statement

I will try to apply some naive classifier, like decision trees or perform grouping or clustering algorithms and try to solve classification problem this way. I will also try applying deep learning algorithms like CNN to this problem and see which performs the best. I will use some pre-processing techniques like reducing dimensions, converting images from RGB to grayscale to identify different patterns within images, normalization or standardization and so on.

Benchmark Model

I plan to compare to the Vanilla CNN caffe model. Here is the link for the model - https://gist.github.com/ishay2b/58248e5f3c3bf575ac40 . I will first train my own model then I will train the Vanilla CNN caffe model on the same CIFAR-10 dataset and I will use the “evaluate” as my evaluation metrics in both my model and the benchmark model.

Evaluation Metrics

Here the Evaluation Metrics that is used in the benchmark model to evaluate the predictions of CNN on the CIFAR-10 test images is “evaluate” and I will use the same Evaluation Metrics to evaluate my predictions. This metric Returns the loss value & metrics values for the model in test mode. The computation is done in batches and we can specify the batch size (If unspecified, it will default to 32).

Project Design

At first we will load the **The CIFAR-10 dataset**. Keras deep learning library provides a convenience method for loading the CIFAR-10 dataset. The dataset is downloaded automatically the first time this function is called and is stored in your home directory. Then we will check the number of images in training and testing dataset and plot some of the images to confirm that the dataset is completely loaded. Now comes the preprocessing of data, we will check the dimensions of images so that we can decide whether the images need the reshaping of dimensions or not. Then we will do normalizing step which is required in most of the image classification problem before giving input to CNN. In this step we will divide every pixel by 255 as for grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of gray. And therefore we can very quickly normalize the pixel values to the range 0 and 1 by dividing each value by the maximum of 255. Then we will check if the labels need preprocessing or not and if it needs we will do the corresponding steps. Then we will look for data augmentation to rotate the images at different angles to provide variety of inputs to our model. Then we will try different CNN architectures by trying different combinations of convolutional, pooling and activation layers on the inputs. I am thinking of applying an approximate of 5 convolutional layer with some maxpooling layers between them and after fitting the model to the training data (followed by compiling in which we define a loss function and an optimizer), we will try to find which architecture gives the maximum accuracy on the test set by using evaluate function which tells us the accuracy and error rate. Then for improving the results we can use some regularization method like adding some dropout layers to prevent overfitting. We can also use ELUs in place of ReLUs, especially with Maxout. ELUs are smoother versions of ReLUs that speed up convergence and classification accuracy. We can also convert the images from RGB to Gray this will help the architecture in identifying patterns within images.