Object Detection Using Regression-type Model vs. Faster Region-Convolutional Neural Network

Alfred Cheung, Yang Zhao and Yuetao Lv

Introduction

While modern convolutional neural network architectures have been very successful at this task of single object detection, in autonomous driving, it not sufficient to just be able to detect an object accurately. In addition to accuracy, Speed is also a major factor. One can imagine an autonomous vehicle collecting live footage of its surroundings constantly as it moves. It is necessary to process this live stream of frames and detect objects in them efficiently.

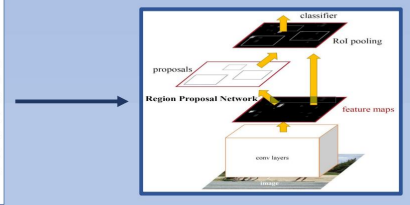
One algorithm aimed at overcoming this challenge is called faster region-convolutional neural network (faster R-CNN), which falls under the category of region-based methods. These methods rely on sampling only a relatively small subset of proposed regions out of all possible regions in an input image to detect potential objects. Then, on each region, an image classification using a CNN architecture is performed (with a softmax output layer) to determine what object is in that region (cat, dog, person, etc.). This approach is what is meant by "region" CNN and it has the benefits of: (1) being able to detect many objects in a given image, and (2) being able to perform the detection task accurately.

The goal of this project is to compare the speed of this algorithm to simple regression-type object detection baseline algorithms. Regression- based baselines frame the object detection task as a regression problem – the goal is to predict the x and y coordinates of the center of an object as regression variables. These baselines involve no region-proposal steps and can be very fast.

How it works

A main time bottleneck of region-proposal based approaches to object detection is in proposing a good set of regions that might have objects in them. Each of these proposed regions must then be fed into a classifier to determine if there is an object in it. The classifier itself tends to be a large, complex CNN that can extract relevant features for object classification. Thus, there are ***two*** computationally heavy parts: (1) a region proposal computation, and (2) a feature extractor/classification computation.

The central premise behind the faster R-CNN algorithm is the following: have a convolutional neural network which is able to perform both tasks (1) and (2) simultaneously. This "dual purpose" network is called a Region Proposal Network (RPN). Essentially, it is a CNN which produces as output key features from an input image. The same features are then used to make region proposals and to use in the classification task for each proposed region. By having a single RPN which accomplishes both of these tasks, time can be saved. A schematic diagram of this method is shown in the figure to the right.



Data and Preprocessing

We obtained images and true labels for each of them from Kaggle. The "true labels" for each image is given as four numbers, the x and y coordinates of the top left corner of the bounding box (x_{TL}, y_{TL}), and the x and y coordinates of the bottom right corner of the bounding 1 box (x_{BR}, y_{BR}). By taking the average of these two coordinates, we can find the center(x_{center}, y_{center}) of where the object is located:

In addition to this data, for the faster R-CNN, we will use an existing model that is pre-trained on data from the PASCAL Visual Object Classes Challenge 2007. Thus, we are implicitly using data from this dataset.

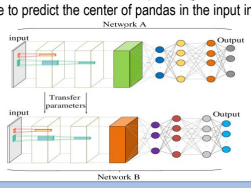
Baseline Models

Baseline Model 1: Fully-Connected Neural Network

1. Flatten input image into a one dimensional array. I.e. if the original input image shape is (n_x, n_y, 3), the resulting flattened input shape is ($n_x \times n_y \times 3$, 1).
2. First fully connected hidden layer with 128 neurons using Relu activation. The result is batch normalized.
3. Second fully connected hidden layer with 64 neurons using Relu activation The result is batch normalized.
4. Final output layer with two neurons. These represent the predicted center coordinates for the object. They are also passed into a Relu activation.

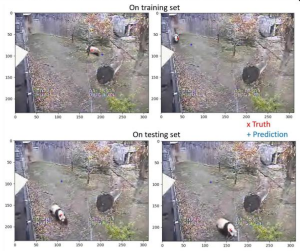
Baseline Model 2: CNN with Transfer Learning

We tried to use VGG-16, a well-established CNN for classification tasks to conduct transfer learning. The weights are pre-trained from the data set "ImageNet". The way we conduct transfer learning is to use the output from the second-last fully connected layer of VGG16 as encoded features of the input image. We added a 2-neuron fully connected layer there to predict the center of pandas in the input image.



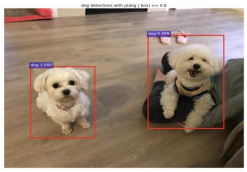
Results and Analysis

We trained the baseline model on 5500 training example images and evaluated on 500 testing images. The training loss was around 500 while the testing loss was around 5000, giving a factor of 10 difference. This seems to indicate overfitting of our baseline model.



Originally, we planned to test on the same set of panda images as for our baseline models. However, we encountered a major issue: the pre-trained faster R-CNN model was not trained on pandas and thus did not have panda as a classification category. We could not train in new model in time. For this reason, we regrettably could not make a direct comparison with the baselines using the panda images. We will instead predict on some dog images and comment on the speed of the faster R-CNN implementation as compared to the baselines.

203 regions were proposed and tested for the presence of a dog. The pre-trained model is able to successfully predict both dogs. We performed the same task 100 times and averaged the prediction time over those trials. On average, the prediction task took 0.31 seconds. For regression-based the baseline model II, the prediction task took 0.011 seconds. It appears that the baseline is much faster than the faster R-CNN. However, we must remember that for a single input image, faster R-CNN proposes many regions and does classification on each of them. Thus, 0.31 seconds represents the total time taken to make the proposals and classify all potential objects in each one, at the same time achieving very high accuracy. Through this comparison, we see how good the performance of faster R-CNN is in both speed and accuracy.



Acknowledgements

We thank our TA ***Farzan Farnia*** for many good suggestions throughout this project.