Introductory Implementation of Object Detection in Images using R-CNN

Jace Mixon
University of Central Florida
jace.mixon@knights.ucf.edu

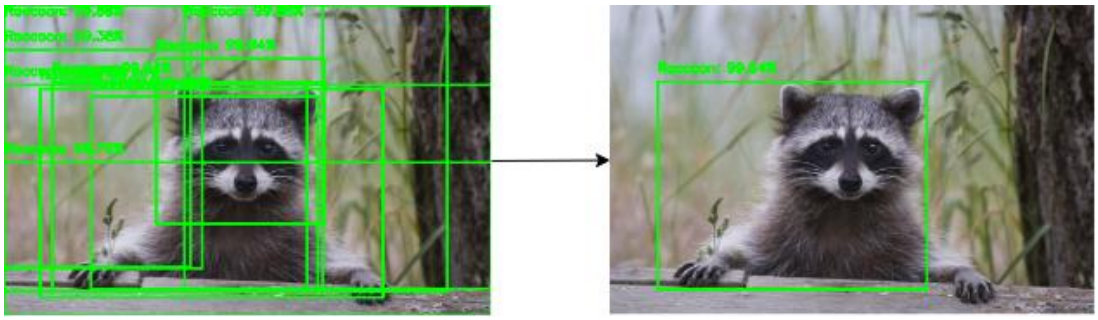

Figure 1: Application of non-maximum suppression on a list of region proposals, outputting the result of object detection.

Abstract

Object detection focuses on two important problems to be identified: what object exists in an input image and where specifically are they located in the image. While it has been solved with convolutional neural networks and SVMs on how to identify which objects are in an image, a difficult task is then being able to identify the localization of the object within the image. The introduction of R-CNNs have shown that convolutional neural networks can be used to solve this problem and then later optimizations of the same network shows improvements to this approach. In this paper, a demonstration is shown on how R-CNNs work at a base line level to gain a better understanding of the fundamentals to then allow for further understanding of the optimizations in future demonstrations.

1. Introduction

Object detection is a Computer Vision problem that is distinct from other object-oriented image tasks, such as object recognition and object localization, in that it requires the objective from both object recognition and object localization to perform its tasks. The task of object recognition is that when an image is observed, the network (or any program) must be able to determine if a specific object exists within the image, based on the classifiers that it knows. For object localization, the network (or any program) must be able to determine the location of different instances of one or multiple objects within the image. When discussing the task of object detection, it must do a combination of the two tasks in that it must be able to locate different instances of one or multiple objects within an image *and* be able to identify which class each of the instances of the objects it belongs to.

Previously, object recognition tasks have been solely reliant on key feature extraction algorithms, such as HOG [5] and SIFT [6], since the algorithms do an effective job at identifying which parts of the input image is a key point to be focusing on for the network and, based on those key points, what kind of object is the network seeing. When it comes to the task of object localization, the only general approach that has been introduced at the time was the concept of the scale pyramid and sliding window method. In this methodology, a bounding box is generated and then "slides" across the image, detecting any objects it can find within the box's boundaries. Since the box's size was fixed, there were considerations that the object may not fit perfectly to the size of the bounding box. To solve this issue, the scale pyramid is used, where the image is scaled down multiple levels and the algorithm repeats itself for each level.

The problem with this approach is that the algorithm is slow since the bounding box must slide across the input image per pixel and must repeat the same process on different scales of the same image. Additionally, the algorithm uses a preset box of a certain dimension, but the object within the image may not have the same dimensional bounds that can fit within the set bounding box of the algorithm.

To address this issue of object localization within the

object detection task, it must be considered that convolutional neural networks, also known as CNNs, were effectively replaced by Support Vector Machines (SVM) in recent times, so the idea of using a CNN was met with some question. However, Ross Girshick et al.'s approach of the Region-based Convolutional Neural Network (R-CNN) [1] showcased the advantages a CNN can have when it comes to doing both object localization and object recognition.

In the approach, the problems that were considered was how objects would be localized in a deep network and how a high-capacity model could be trained with only a small amount of annotated detection data. For R-CNN, it addresses the first problem by creating regions of interest to observe within any image known as the "recognition using regions" paradigm [1], also known as region proposals. For the second problem, the idea of using unsupervised pre-training was considered to train the network, but it ultimately decided on using a supervised pre-training methodology on a large auxiliary dataset, followed by fine-tuning on a smaller dataset.

The R-CNN approach is distilled into three main components. For the first component, the algorithm generates category-independent region proposals, which it does so by using an approach known as selective search, where it generates region proposals based on where objects may likely appear in an image regardless of what object is being detected. Of course, only a select few of these proposals will contain an object of interest in the image. For the second component, the algorithm uses a large convolutional neural network to extract a fixed-length feature vector from each region to find key points to look at for classification. Lastly, the third component of the approach takes the feature vectors that were extracted from the previous component of the algorithm and is classified based on class specific linear SVMs. One main point of the third component of the approach is that multiple proposals may contain the same object at different locations of the image, where one proposal may have full identification of the object while another proposal may have a clipped version of the object. To resolve the issue of having inadequate proposals considered, a greedy non-maximum suppression algorithm is implemented to dispose of extra proposals.

This use of the R-CNN to do object detection has shown to be an effective network to do such a task and reinvigorated the use of convolutional neural network for image tasks once again. However, the R-CNN algorithm does contain flaws that hinder its performance. For instance, having the three components makes the training pipeline more complex than having one continuous pipeline. Additionally, the training itself is expensive for each region because the region proposal itself is passed through the network multiple times and it must do so for each region in the image, which may contain thousands to process. Lastly, the R-CNN approach has the regressor and classifier heads to be separate, which means that training for both heads are separate. Therefore, synchronization issues can arise from having two separate heads of the network.

To resolve this issue, an optimization of the R-CNN was made through the Fast R-CNN [2] implementation. It resolves these issues by having the entire image passed through the network rather than having singular proposals passed through the network. Also, the classifier and regressor heads are part of the same network so that the fine-tuning of the hyperparameters can impact the two heads simultaneously rather than individually. Lastly, the network is trained end-to-end to keep the one entire network structure consistent. This optimization of the R-CNN approach has shown to have a speedup in computation time while also providing a better mAP score between the two.

Afterwards, it was considered that the Fast R-CNN could be improved further since the network does not use region proposals and thus must check the entire image to find the possible locations of the object in question. In the implementation of the optimized version of the Fast R-CNN labeled Faster R-CNN [3], the network can now be able to generate regions of interests for the network to use as region proposals to minimize the time spent on searching the image for possible locations of the object. This is all included in the end-to-end training of the network and does show an increase in mAP score and a speedup in computational time.

While it is enticing to try to implement a Faster R-CNN for the most optimal network possible to do an object detection network implementation, it was decided to focus more on the base line approach to creating an R-CNN for the purpose of object detection to get a better understanding as to how the main parts of the network that will later carry over to Fast R-CNN and Faster R-CNN works. The overall objective of this paper is to create an understanding of how Girshick's R-CNN methodology works in a demonstration of object detection using a singular classification network.

2. Method

To construct a R-CNN for the purpose of this demonstration, the network should demonstrate an understanding of being able to localize objects within an image using a deep network and the training was going to be done using a "scarce" number of images in the dataset. To appropriately train the network, the scarce number of images will be used to build a larger dataset and the scarce number of images is used to fine-tune the network's parameters.

The network is constructed with the three components in

mind: use selective search to gather region proposals for each image, use a network to gather feature vectors for each proposal passed into the network, and use class specific linear SVMs to detect what objects are being seen in those proposals. For the first component of the network, it was considered to handcraft each region proposal based on any research done with region proposals within R-CNN architecture. However, to avoid redoing work that already exists, the network utilizes OpenCV's region proposals that are archived in their libraries. With this functionality, the bounding boxes that represent the region proposals are provided and can be used to feed into the network as inputs.

For the approach of the second component, the network is mainly comprised of an existing network, which is the MobileNetV2 [4]. The network has been trained using ImageNet and was selected because it has become a common network for image classification that it is contained as a model within TensorFlow's libraries. In addition, the model supports the option to remove the fully connected layers of the network so that a custom-made classifier can be used in replacement of the already existing FC layers. The customization option to allow for different fully connected layers and classifiers is what will be used to address the final component of creating the R-CNN, which is to be able to create a class specific linear SVM.

Overall, the approach to the demonstration is as follows: a database with a scarce number of images is initially gathered (for the consideration of this demonstration, having 200 images is sufficient). Afterwards, OpenCV's selective search and region proposals will be used to generate a larger dataset that will be used to train the network to classify objects within the new set of images. Afterwards, we use the network to feed any image through the network and use OpenCV's selective search to find proposals again, but the proposals are not saved into a separate database. Instead, the network will take in each proposal as a separate input and decide if the proposal does or does not contain an object that it knows in its classifier set. After the network has separate the proposals into a group of proposals that the network has a low confidence in there being an object in it (the object may not exist in the proposal) and a group of proposals that the network has a high confidence in there being an object in it (the object may exist in the proposal), then the group of proposals that have high confidence of it containing an object is observed and a greedy non-maximum suppression is used to single out the most confident-scoring box. This box (or boxes) will be considered the result of the network's object detection task.

The network will be crafted using the Keras and TensorFlow machine learning framework to abstract the number of parameters that need to be adjusted for the network's training.

3. Results

The dataset that was used during this demonstration is a raccoon database that was curated by data scientist Dat Tran that was also used to do object detection. The database contained a total of 200 images of raccoons, some of which may contain multiple racoons in the same image, and

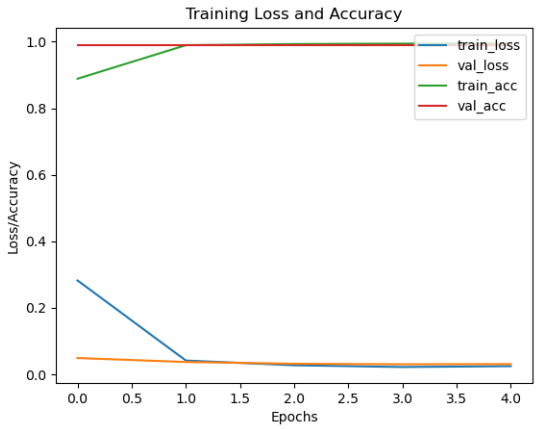

Figure 2: Graph of the classifier model's loss and accuracy through training and validation across four epochs.

contains an annotation folder for the images which contains the ground truth boxes that bound where the raccoons are in the image. This is a helpful assist when it comes to creating a separate large dataset in the beginning because the demonstration can use intersection-over-union to decide whether the given proposal in the given region proposal list overlaps with the ground truth box by enough.

For a listing of the parameters that were included for this demonstration, in creating the new dataset for training the network, a minimum threshold of 70% for the IOU score was used to decided if the proposal was sufficient or not. This led to a total of 1,547 positive results and 2,200 negative results. For proposal counts during the creation of the new database, a maximum of 2,000 proposals were analyzed per image, while during the analysis of proposals during object detection using the fine-tuned network, a maximum of 200 proposals were analyzed per image. For the proposal boxes that were considered out of the 200, only those boxes that had a 99% confidence or higher were to be considered. For the non-maximum suppression algorithm, a threshold of 20% was used to indicate a sufficient overlap between two boxes. If it was the case that two boxes overlapped by 20% or more, then the box with the lower confidence would be dropped.

In creating the new head for the network, an average pooling of 7x7 was conducted from the MobileNetV2's output layer. Then the layer was flattened and brought to a fully connected layer of 128 neurons, all with the ReLU

activation function. A dropout of 50% is used as a regularizer factor and then the final output layer of the head is a fully connected layer with 2 neurons with a softmax activation function. This is to indicate if the raccoon exists or does not exist within the image proposal. To make sure only the head was the only part of the network that was being trained, all the other layers in the MobileNetV2 were frozen so that their weights would not change.

Over five epochs, the network gains an accuracy close to 99% and the overall loss across the training and validation portions were minimized with no spikes upwards, leading to a formidable fit of the model that avoids issues with overfitting.

When conducting object detection on a select few images, it was noticed that even with a confidence threshold of 99% still led to having a vast number of boxes to filter through the non-maximum suppression algorithm to find the highest scoring box. Even after the algorithm was implemented with a threshold that is rather low in comparison to normal threshold amounts, such as 30% and 50%, multiple boxes would still appear in the resulting image of the raccoon. The R-CNN also seems to have trouble with cases where the raccoon took up most of the image. Regardless, the simple R-CNN demonstration showcases an effective object detection network for this dataset.

4. Conclusion

Throughout the creation of this demonstration, there were parts of the creation that could have been a difficult obstacle to overcome but were greatly alleviated by the helpful assets that were discovered. As an example, the fact that the raccoon database contained annotated files that indicated the ground truth boxes helped in creating the second dataset and avoided having to go through all 200 images to draw a ground truth box to make the positive and negative results for the second dataset. Additionally, having OpenCV's selective search function produce 2000 region proposals automatically helped avoid making the proposals by hand for each image. Lastly, having a CNN model with a customized head helped with being able to use a model with already-created feature detectors and all that needed to be created was the classification portion.

Regarding the overall demonstration, there were plenty of details that felt too abstract to be able to figure out and implement initially, such as how non-maximum suppression works with bounding boxes and how to generate region proposals. After being able to work on this R-CNN, the concepts now feel easier to implement since I can see which parts of the source code does which task in the network. This allows me to have a better appreciation towards how object detection works in general and leads me to become more interested in how the optimized implementations were crafted.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," arXiv:1311.2524, 2014.

[2] R. Girshick, "Fast R-CNN," arXiv:1504.08083, 2015.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv:1506.01497, 2016.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv:1801.04381, 2019.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," In CVPR, 2005.

[6] D. Lowe, "Distinctive image features from scale-invariant keypoints," IJCV, 2004.