Single Shot Multibox Detection

Bahdah Shin
Computer Science Department
Rochester Institute of Technology
Rochester, NY, USA
bcs3904@rit.edu

Matthew Witman
Computer Science Department
Rochester Institute of Technology
Rochester, NY, USA
mnw4380@rit.edu

Daniel Moore
Computer Science Department
Rochester Institute of Technology
Rochester, NY, USA
dxm9604@rit.edu

***Abstract*—Image object detection is one of the most interesting problems in computer vision, determining what is in an image, and where in the image those things are. We implemented the Single Shot Detection (SSD) algorithm, to detect multiple classes of items in our images, and tested it on the VOC dataset. This document is an analysis and practical application of the SSD algorithm to perform image analysis.**

I. Introduction

The purpose of object detection is to locate meaningful objects within an image and draw a box around it. It does two things: object classification and localization. Object classification makes scores for various object types, which makes it a classification task. Object localization makes coordinates of a box that may or may not contain an object, which makes it a regression task. Single Shot Detection (SSD) is one of many convolutional neural networks (CNN) that can be organized into three parts.

1) Base convolution
2) Auxiliary convolution
3) Prediction convolution

II. Base Convolution

Base convolution can be derived from existing image classification. The paper explores VGG-16 architecture, but it is not limited to this architecture. It uses existing network architecture because it is proven to work well with image classification and is excellent at capturing the features of an image. In addition, the layers can be pre-trained so that it can transfer the learning from the VGG network over to SSD. Like the paper, we used VGG-16 that is pre-trained on the ImageNet Large Scale Visual Recognition Competition (ILSVRC) dataset. The VGG-16 has a fully connected convolutional layer at the end of the network. A fully connected convolutional layer is a layer converted from an N-dimensional convolutional layer into a one-dimensional layer. To add more layers at the end, the fully connected convolutional layer needs to be converted into a convolutional layer.

III. Auxiliary Convolution

The auxiliary convolution is added on top of the base convolution and provides the higher-level feature maps. Each layer gets progressively smaller than the last.

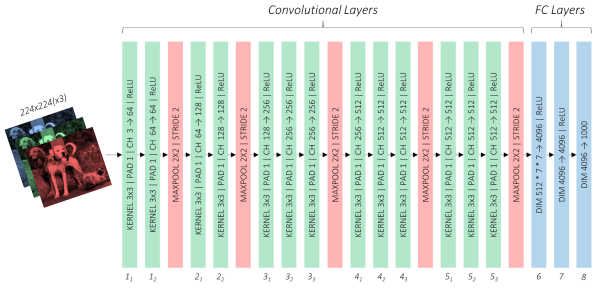

Fig. 1. Original VGG-16

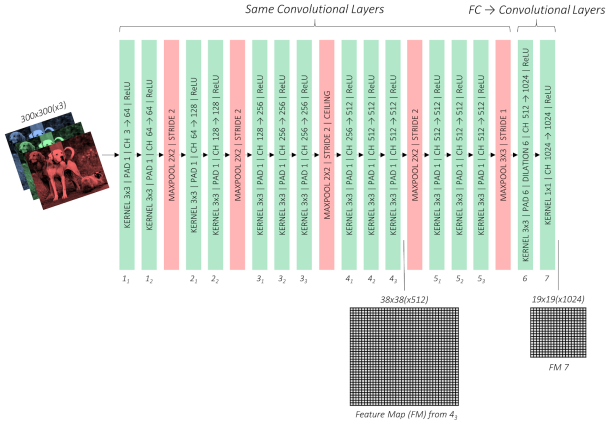

Fig. 2. Modified VGG-16

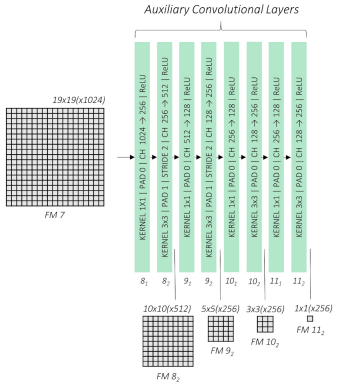

Fig. 3. Auxiliary Convolution

IV. Prediction Convolution

There are six feature maps of various scales and granularity located in layers: conv4-3, conv7, conv8-2, conv9-2, conv10-2, and conv11-2. The sizes are 38x38, 19x19, 10x10, 5x5, 3x3, and 1x1 in that order. For each prior(anchor box) at each location on each feature map, we want to predict the offsets of the bounding box. The values include the x and y coordinates for the top-left corner of the box. It also includes the width and height. Two convolutional layers are created from each feature map. They are the localization prediction and class prediction. For localization prediction, there are 24 channels: 6 predicted boxes with 4 offsets(x,y, w, h) derived from 6 priors. For class prediction, there are 6 * c channels where c refers to a number of classes and backgrounds. The 6 refers to 6 sets derived from 6 priors. In total, there are 8732 predicted boxes.

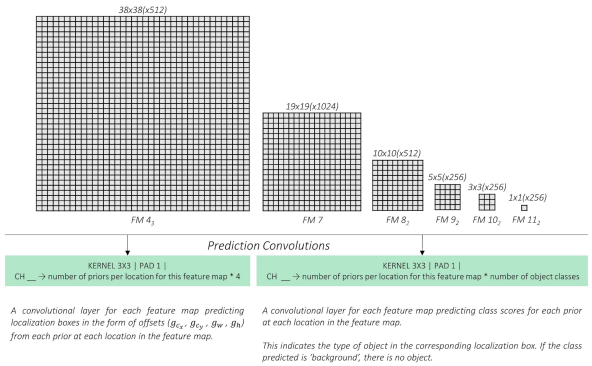

Fig. 4. Prediction Convolution

V. Prior (anchor boxes/ default boxes)

Objects can occur at any position with any size and shape, but many positions are improbable or not interesting. In addition, an approximation of location with a slight deviation is good enough. Priors can be used to shrink the mathematical space of infinite possibilities to thousands. Priors are precalculated, fixed boxes that contain predictions of classes. Collectively, they can show where it's more probable. Other papers refer to this term as anchor boxes or default boxes. Priors are created based on the shapes and sizes of ground truth objects in the dataset. They are applied to various low-level and high-level feature maps. For each pixel in the feature maps, there are various aspect ratios. In total, we get 8732 priors.

Feature Map	Feature Map Size	Prior Scale	Number of Priors per Position	Total Number of Priors on this Feature Map
conv4-3	38x38	0.1	4	5776
conv7	19x19	0.2	6	2166
conv8-2	10x10	0.375	6	600
conv9-2	5x5	0.55	6	150
conv10-2	3x3	0.725	4	36
conv11-2	1x1	0.9	4	4
Grand Total	-	-	-	8732 priors

VI. Training

During training, the predicted box can be compared with the ground truth by using the Jaccard Index, otherwise known as Jaccard Overlap or Intersection over Union(IoU). It measures the degree or extent to which two boxes overlap. If the value is greater than 0.5, it considers it as an object.

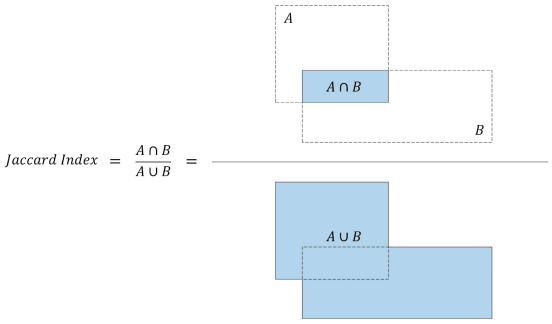

Fig. 5. Jaccard Index

VII. Inference

During inference, the final image will contain all of the 8732 prediction boxes. Within those boxes, there could be redundant boxes indicating the same object. To reduce the redundant boxes, we use non-maximum suppression, which suppresses all but one box with the maximum score indicating one object. The non-maximum suppression uses the IoU equation on boxes indicating the same class. If it is greater than 0.5, it has a high chance that it is the same object.

VIII. Dataset

Our dataset comes from the PASCAL Visual Object Classes (VOC). PASCAL, or Pattern Analysis, Statistical Modelling, and Computational learning, was a long running object class recognition challenge that ran between 2005 and 2012. Their Visual Object classes are sets of images that are labeled with 20 classifiers (such as planes, persons, and dogs). We trained out SSD implementation on the 2007 and 2012 training sets (460 MB and 2GB in size respectively) and the 2007 testing set (451 MB in size).

IX. Practical Application: Training

Training our network took around 3 days, on a CUDA accelerated consumer-grade Nvidia RTX 2070. Training on our training dataset averaged 5 GB of VRAM usage, however there was little CPU usage as our training was primarily GPU accelerated.. The completed training weights were 200MB in size, and these served as our Anchor Boxes. Once completed, this allowed us to test out the inference of our SSD algorithm on hardware that was not CUDA accelerated.

The algorithm ran the training for 232 epoch (cycles).

X. Results

Categories	AP
aeroplane	0.793396
bicycle	0.84419
bird	0.780478
boat	0.702716
bottle	0.501022
bus	0.872393
car	0.865685
cat	0.88398
chair	0.588105
cow	0.838266
dining table	0.773233
dog	0.85965
horse	0.879353
motorbike	0.849623
person	0.787463
potted plant	0.541377
sheep	0.795752
sofa	0.801068
train	0.872066
tv monitor	0.758006

Mean Average Precision (mAP): 0.779

Fig. 6. Correctly found dog, identified plane shaped cart correctly as a motorcycle

Fig. 7. Correctly finds both chairs, but thinks there is a table in the middle of them.

XI. Discussion of results

As can be seen from our evaluation, the SSD Multibox detection algorithm is in general good at detecting objects. This however depends on many factors. For instance while the AP peaks at 0.88398 for a cat, it goes as low as 0.501022 for a bottle. Overall our model had a mAP of 0.779. That means that on average across all the objects we attempted to detect, we got 77.9% of the correct. While this is good enough for an example project it isn't nearly sufficient enough for a vital application.

In the image with the dog and the plane, the model detected the dog with ease since it's very distinctive. However when it had to predict the plane shaped cart, it chose motorcycle. This is debatably correct since it isn't truly a plane but it also looks more like a plane than a motorcycle. In that case the development team would have to decide what answer they consider to be more correct and train the model to that specification. The image with the chairs is able to correctly identify both chairs. However it also identifies the middle as a table, where instead it's just the two seats of the chairs.

XII. Conclusion

In conclusion, we decided to implement our own variation of the popular Single shot multibox detection algorithm. This algorithm is a fast and efficient object detection algorithm that can be run in near real time on relatively light hardware. Because of it's multibox capabilities we are able to detect multiple objects in one image. Our code will take in a list of files and save the resultant detection for each image.

For our model, we had average precision ranging from 50% to 88% with the mAP being 77.9%. That's decent precision for an example project like this but for a real world application you would want that number to be higher.

The two images shown demonstrate the common successes and pitfalls of out algorithm. For things that are very clear an distinctive such as the dog and the chair, it does quite well. However for things that are a bit more ambiguous such as the plane shaped cart, it doesn't do quite so well. It also by it's nature will sometimes find objects where they don't exists such as finding a table where it's really just the two chair seats close to each other.

Overall, single shot multibox detection is an incredibly useful algorithm that has wide ranging applications such as with self-driving cars and environmental surveys. As our data shows, it is still very much a developing field with some work left to do, but it still fairly accurate and highly useful.

REFERENCES

[1] Sgrvinod, "sgrvinod/a-PyTorch-Tutorial-to-Object-Detection," GitHub, 15-Feb-2020. [Online]. Available: https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection. [Accessed: 20-Apr-2020].

[2] E. Forson, "Understanding SSD MultiBox - Real-Time Object Detection In Deep Learning," Medium, 09-Jun-2019. [Online]. Available: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab. [Accessed: 20-Apr-2020].

[3] "13.7. Single Shot Multibox Detection (SSD)¶ Colab Open the notebook in Colab," 13.7. Single Shot Multibox Detection (SSD) - Dive into Deep Learning 0.7.1 documentation. [Online]. Available: https://d2l.ai/chapter_computer-vision/ssd.html. [Accessed: 20-Apr-2020].

[4] J. Rieke, "Object detection with neural networks," Medium, 29-Oct-2018. [Online]. Available: https://towardsdatascience.com/object-detection-with-neural-networks-a4e2c46b4491. [Accessed: 20-Apr-2020].

[5] D. Dmitriev, "VGG16 Neural Network Visualization," YouTube, 29-Oct-2019. [Online]. Available: https://www.youtube.com/watch?v=RNnKtNrsrmg. [Accessed: 18-Apr-2020].

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," Computer Vision – ECCV 2016 Lecture Notes in Computer Science, Dec. 2016.

[7] Simonyan, Karen, Zisserman, and Andrew, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv.org, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 20-Apr-2020].