Fine-tuning Pre-trained Large Scaled ImageNet model on smaller dataset for Detection task

Kyunghee Kim
Stanford University
353 Serra Mall Stanford, CA 94305
kyunghee.kim@stanford.edu

Abstract

We use a pre-trained model on ILSVRC dataset for classification to train a model for a smaller dataset for different task i.e., detection. The result shows promising performance of this approach and we furthur think about the difference between training the model for classification and detection and the method that can improve the detection where classification works fairly good meanwhile the detection fails.

1. Introduction and related work

In RCNN[1] Girshick et al. shows that supervised pre-trained model on large dataset(ILSVRC) to a smaller dataset(PASCAL VOC). They extract region proposals using selective search[2] to extract candidate regions for bounding boxes and distinguish between negative and positive bounding boxes using the overlap with the ground truth bounding box. Using pre-trained model on large dataset for classification gives very good result also on classification on smaller dataset for example for pre-trained VGG 16 layer model performance on PASCAL VOC 2007 dataset is 89.3%(mean AP)[4] while fine-tuning with Krizhevsky et al.'s model[5] on PASCAL VOC 2007 dataset gives 54.2% (mean AP) for detection task [6].
We follow this approach and use VGG 16 layer model [3] for pre-trained large scale ImageNet model.

2. Approach

We use Caffe[7] to fine-tune VGG 16 layer model on PASCAL VOC 2011[8] dataset. VGG 16 layer model architecture is like the following in Table 1 as described in [3]. The last Fully-Connected layer of this model has been changed to FC-21 for our fine-tuning on PASCAL VOC 2011 dataset since there are 20 object classes in PASCAL VOC dataset and one more class is for the background. The 20 object classes in VOC dataset is like the following;

aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair , cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor

We use 224 x224 as input image size instead of 227 x 227 in RCNN[1] and batch size 10 to fit the parameters in VGG model.

Input(224 x 224 RGB image)
conv3-64 conv3-64
maxpool
Conv3-128 Conv3-128
maxpool
Conv3-256 Conv3-256 Conv3-256
maxpool
Conv3-512 Conv3-512 Conv3-512
maxpool
Conv3-512 Conv3-512 Conv3-512
maxpool
FC-4096
FC-4096
FC-1000
Soft-max

Table 1. Architecture of VGG 16 layer model

3. Experiment

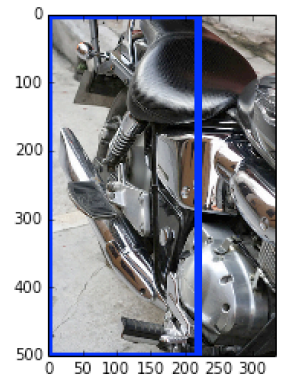
Fine-tuning We use Caffe[7] and did 100000 iterations which took about 10 hours for training. For training and validation data we used the code for RCNN[1] and extracted region proposals in PASCAL VOC 2011 training and validation datasets. We add one more layer at the last layer of VGG model which is the accuracy layer to see the performance during training and it was about 0.8~1 accuracy from 10000 iterations. In the experiment below we are using the model that was saved in 90000 iterations. However by the observation of the accuracy and loss during training the model might have converged already at 10000 iterations but we do not test the performance of this model here.

Detection We use python wrapper for Caffe[7] for

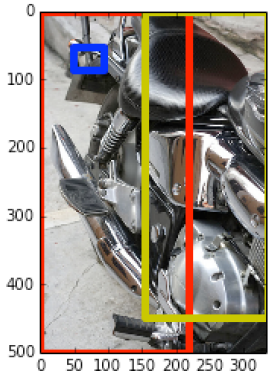
detection. It was not easy to understand and figure out how to use 'Deployment Input'[9] in Caffe so this step took a while to setup prototxt file as a caffe model definition. In the results below we show the detection results using this caffe python wrapper.

Results Table 2 below shows the detection results. For example for the first example motor bike the left column show the region proposal that has the highest detection score for motorbike class. And the below 5 scores mean for this given detection box the top 5 prediction class for this box. We can see that for all example we can find the gound truth label of the object. But the thing is that the location of the bounding box is not always centered on the object. For example for the aeoplane example on the 3rd and train on the 7th, the bounding box is drawn on the nearby context cloud and grains near the ground of the railroad. From this we can conclude that for classification this model works good because the nearby context also helps the scene to be correctly classified. For example if there is cloud that it is likely the scene is in the sky so it is likely that we will find an aeroplane or a bird. The original purpose of this project was to see these features from filters by visualizing them so that whether we can distinguish filters that learn context or the object so probably we can figure out which filters are helpful for detection task. But we have not made a progress on that point in this quarter. But that could be a good future work relating to this project. And on the right column it shows the top 3 detection bounding box after applying Felzenszwalb et al.'s non-maxima suppression[10]. Underneath there is a yellow/red box indicating whether human would judge this detection result succeeded or not. Yellow means success and red means a failure case. For example for the 5th example in 'dog' example, its classification will be correct because we are getting the top 1st score as a dog but its detection is a failure because even though the detected bounding box says it is a dog it does not overlap with a ground truth bounding box of a dog. Meanwhile interestingly the detection result is a nearby context which looks like a context for garden dining table and pottedplant. A dog is likely to appear in that kind of environment. Therefore it looks like the filters that extract features for context is helpful for classification task but they are not helpful to get the exact location of the object for detection task so it will be helpful if we can investigate among the learned features which filters contribute more to the object rather than the object. The pre-trained model is trained for classification task so this performance gap between classification and detection seems to be reasonable. Therefore in 13 detection results in Table 2 all of them are classified correctly but 8 of them are detected correctly and 5 of them failed in detection.

motorbike



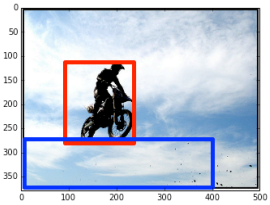
aeroplane 5.209
person 4.671
potted plant 2.94
cow 2.37
motorbike 2.32



r : 2.329
b: 1.338
y : 1.26

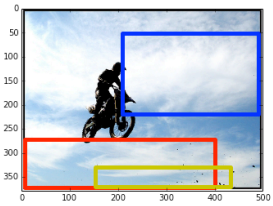
Detection: o

Person and bicycle



For person:
person 6.25
pottedplant 5.985
bird 5.22
aeroplane 4.027
motorbike 3.07

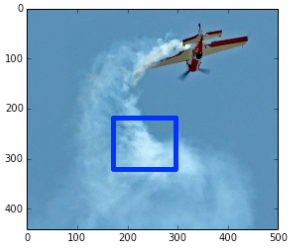
For bicyle
aeroplane 12.06
bicycle 5.746
boat 2.35
bird 1.806
cat



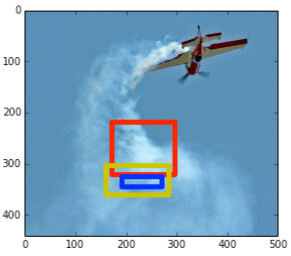
bicycle:
r : 5.74
b: 5.67
y : 5.49

Detection: x

aeroplane



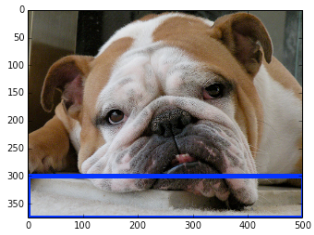
aeroplane 12.034
bicycle 3.0557
bottle 1.94
cow 0.88
pottedplant 0.614



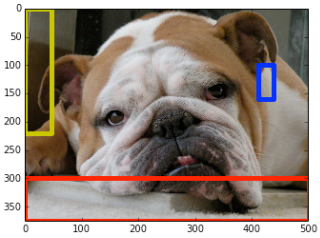
r : 12.0346
b: 11.922
y : 11.375

Detection: x

dog



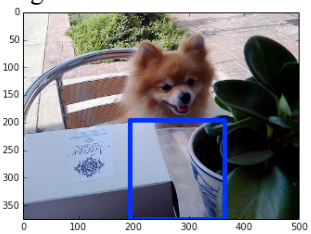
aeroplane 5.494
dog 2.16
train 2.03
chair 1.04
sofa 0.96



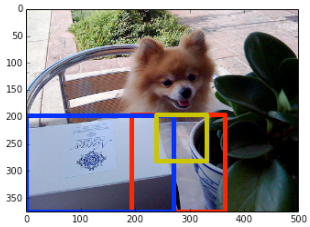
r : 2.16
b: 1.9547
y : 1.85

Detection: o

dog



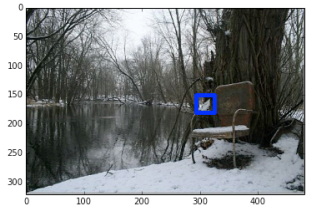
dog 5.46
aeroplane 5.42
cow 4.429
sheep 3.205
train 1.75



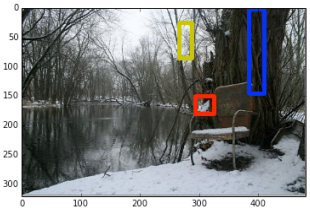
r : 5.465
b: 3.299
y : 3.266

Detection: x

chair



aeroplane 6.784
chair 2.68
sofa 1.8027
diningtable 1.486
sheep 1.2448

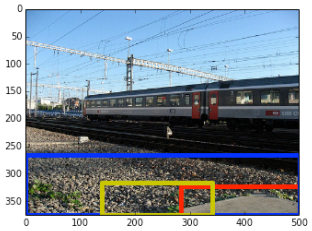


r : 2.680
b: 2.312
y : 2.30

Detection: o

train

aeroplane 10.008
train 2.66
sheep 2.37
cow 1.411
dog 1.159



r : 2.66
b: 2.2965
y : 2.03

Detection: x

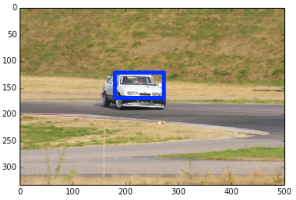
bottle

aeroplane 9.577
bottle 6.930
tvmonitor 3.09
bicycle2.87
bus 1.639

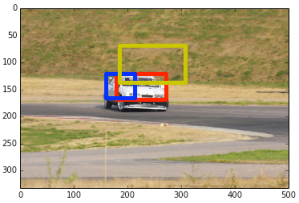
r : 6.930
b: 5.944
y : 5.62

Detection: o

car



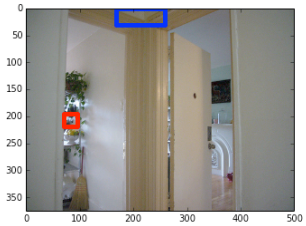
cat 8.36
car 7.24
aeroplane 6.80
tvmonitor 2.04
bicycle 0.92



r : 7.2476
b: 5.874
y : 5.522

Detection: o

pottedplant and bottle



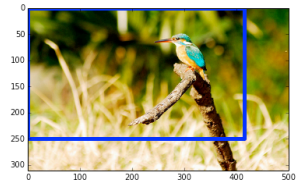
For pottedplant
aeroplane 4.833
pottedplant 4.24
sheep 1.85
boat 1.12
background 0.18

For bottle
aeroplane 11.47
cow 2.40
bottle 2.02
background 1.60
train 0.89

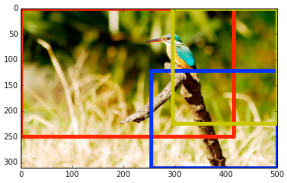
For potted plant
r: 4.24
b: 3.996
y : 3.75

Detection: o

bird



aeroplane 8.332
boat 7.48
bird 3.42
bicycle 2.16
pottedplant 0.38



r: 3.4275
b: 2.015
y : 1.74

Detection: o

person and bus

For person:
pottedplant 5.105
aeroplane 4.90
cat 2.674
bottle 1.48
bird 0.64

For person

r : 0.62
b: 0.0387
y : -0.025

for bus

For bus:
aeroplane 6.706
bus 3.17
pottedplant 2.43
cow 1.48

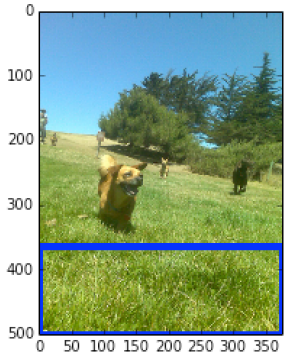
r : 3.17

background 0.83

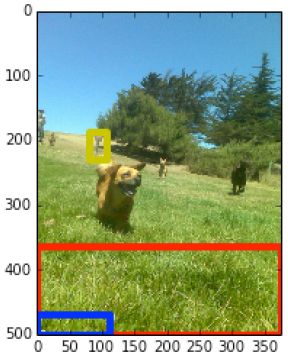
b: 2.937
y : 2.00

Detection: o

dog



aeroplane 10.07
sheep 1.82
cow 1.53
train 1.29
dog 1.27



r : 1.27
b: 1.24
y : 0.789

Detection: x

Table 2. Detection results on PASCAL VOC 2011 dataset

And as a second part of the experiment we found that for example the 10th example seems interesting because the door looks like a big part of the picture even though the PASCAL VOC object class does not have the 'door' label. So we did a classification with vgg 16 layer model but not fine-tuned with PASCAL VOC object using MatConvNet[11] matlab library. Figure 1 below shows a very interesting result that this image classified as a 'sliding door' which has a lot of correlated to 'room door'.

Figure 1. The 10th figure in Table 2 classified as a sliding door in vgg 16 layer model

And this classification is different from the classification

task in [4] since in [4] the classification is to figure out the ground truth label in PASCAL VOC class labels. In Figure 1 result here though is more general classification that can be confirmed with human observation that cannot be measured in computer since the ground truth for door is not annotated and therefore not provided. We thought this is interesting so we also listed the top 10 prediction result for this image and those are with score; sliding door(0.4436), wardrobe, closer, press(0.1836), medicine chest, medicine cabinet(0.1428), window shade(0.0585), shower curtain(0.0275), refrigerator, icebox(0.0180), safe(0.0079), washbasin, washbowl(0.0072), doormat(0.0069), radiator(0.0062).

References

[1] R. B. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in:CVPR, 2014, pp. 580–587.

[2] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[3] Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman, ICLR 2015

[4] http://www.robots.ox.ac.uk/~vgg/research/very_deep/

[5] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet clas- sification with deep convolutional neural networks. In *NIPS*, 2012.

[6] https://github.com/rbgirshick/rcnn

[7] http://caffe.berkeleyvision.org/

[8] Visual Object Classes Challenge 2011. http:// pascallin.ecs.soton.ac.uk/challenges/VOC/voc2011/#data

[9] http://caffe.berkeleyvision.org/tutorial/data.html.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ra- manan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

[11] MatConvNet: CNNs for MATLAB http://www.vlfeat.org/ matconvnet/