Weakly Supervised Object Detection
16-824 Assignment 2

Salvador Medina
Andrew ID: salvadom

March 2017

1 Questions

1.1 Explain the functionality of SecretAssignmentLayer

The functionality is to swap dimensions of the classes and the regions from the input in the forward and restore the original order in the backwards pass.

1.2 There are a few similar layers implemented in Caffe. Name them and explain why they cannot be used in our case

This cannot be achieved through the repertoire of layers that Caffe offers as those do not have the swap dimension functionality. Layers such as Reshape, the main difference resides in that the latter does not change the data and only changes its indices.

1.3 Defining the loss function in the network definition prototxt is not as straightforward as adding one more layer. Explain why.

In our case there was no layer that optimized the sum of C binary-log-loss terms in Caffe, therefore we had to program our own function and link it through the prototxt into our codebase. Although this could have also been solved by concatenating more predefined layers as explained in the following question.

1.4 Describe a combination of layers that would express our target loss for WSDNN.

Given that we require to compute the loss through:

$$L = -\sum_{i=1}^{n}\sum_{k=1}^{C} log((y_{ki}(\phi_k^y(x_i|w) - \frac{1}{2}) + \frac{1}{2}) \quad (1)$$

We only need to preserver the order of the operations by connecting the next four layers in the following order:

1.4.1 1. Bias

$\phi_k^y(x_i|w) - \frac{1}{2}$

In this layer we would have as input the classification result tensor and the set the bias value to -0.5.

1.4.2 2. Eltwise

$y_{ki}(\phi_k^y(x_i|w) - \frac{1}{2})$

In this layer we would have as an input the output of the previous proposed bias layer and the ground truth label, setting the operation to multiplication.

1.4.3 3. Log

$log((y_{ki}(\phi_k^y(x_i|w) - \frac{1}{2}) + \frac{1}{2})$

This layer would have as an input the output of the previous proposed Eltwise layer and set the shift to 0.5.

1.4.4 4. Reduction

$-1.0 \times \sum_{i=1}^{n}\sum_{k=1}^{C} log((y_{ki}(\phi_k^y(x_i|w) - \frac{1}{2}) + \frac{1}{2})$

Finally this layer would have as an input the output of the Log layer, set the sum for both dimensions and set the scale factor to -1.0.

1.5 Explain why multi-scale training works and why it is useful.

Multi-scale training is useful as it aggregates data into the training by keeping the original aspect ratio of the input images. As in training, in the multi-scale approach the images are resized to five different scales, added random horizontal flips and select one of the scales at random as a form of jittering in the forward pass.

2 Code

The submitted package which can be found in my Google Drive here, contains the following files as indicated in the assignment handout:

1. solver.prototxt

2. train.prototxt
3. test.prototxt
4. layer.prototxt
5. minibatch.py
6. test.py

3 Network Visualization

In Figure 1 we can look at a visualization of the built networks through Netscope. As we can see the test network is based on the training network but withouth the dropout layers and the final learning layers as those are not required during test time.

4 Training Error

In Figure 2 we can see that in both trainings (single-scale and multi-scale WSDNN) the error tends to be reduced in both cases. In each plot we can look at the loss after every 20 iterations through 50,000 iterations total.

5 Test Results

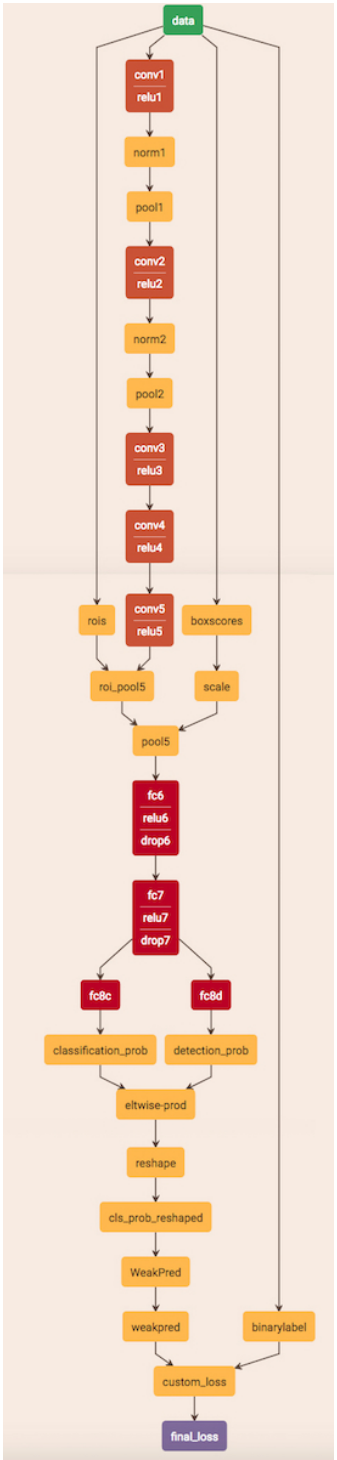
The results shown in this section are for networks trained through 50,000 iterations and after defining the test network in its respective prototxt and setting up correctly the networks in tools
test_net.py. As we can see Table 1 the multi-scale network has an overall improvement over the single-scale WSDNN, although the single-scale performed better over 6 out of 20 classes. Nonetheless, in most of the classes that the multi-scale network performed better than the single-scale it was by a large margin.

6 Test Visualization

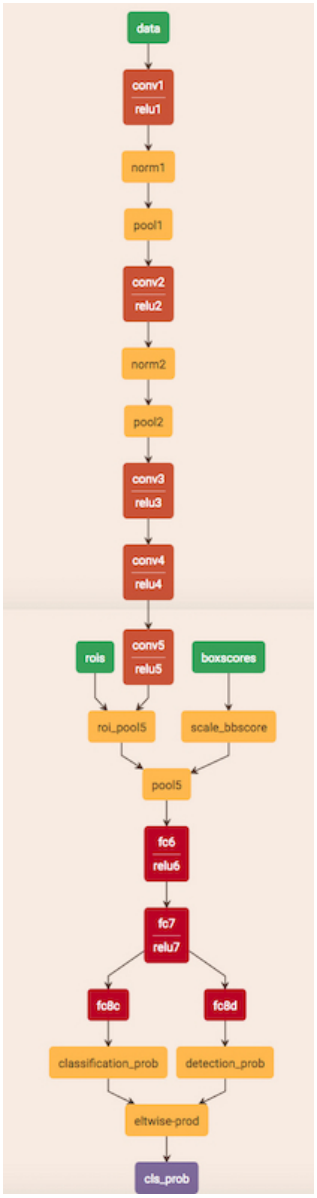
In Figure 3 we can look a sample of the output result of our test with single scale while in Figure 4 we show the results of the trained network with multi-scale jittering.

7 Discussion

During an analysis of the visualizations I realized that most of the bounding boxes for certain animals such as sheep, cat, dog and bird, focus on the head/face

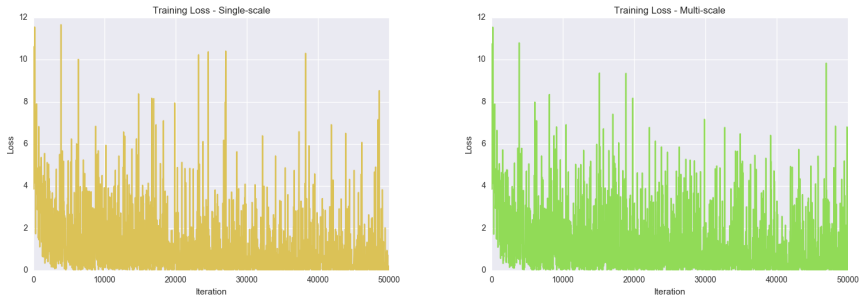


(a) Training



(b) Testing

Figure 1: Network visualization



(a) Single-scale 600 (b) Multi-scale [480, 576, 688, 864, 1200]

Figure 2: Training Loss

Table 1: WSDDN on PASCAL VOC 2007

Class	Single-scale AP	Multi-scale AP
aeroplane	**0.3089**	0.3038
bicycle	0.3733	**0.4471**
bird	0.1203	**0.1992**
boat	0.1842	**0.2105**
bottle	0.0514	**0.0828**
bus	**0.5085**	0.4835
car	0.3525	**0.3675**
cat	0.0775	**0.1175**
chair	0.0499	**0.0664**
cow	0.2036	**0.2251**
diningtable	0.0713	**0.1036**
dog	**0.0974**	0.0645
horse	**0.2786**	0.2725
motorbike	0.3878	**0.4848**
person	0.0187	**0.0285**
pottedplant	0.1174	**0.1217**
sheep	**0.2454**	0.2350
sofa	0.2448	**0.3012**
train	0.4782	**0.5115**
tvmonitor	**0.3778**	0.3585
mAP	0.2274	**0.2493**

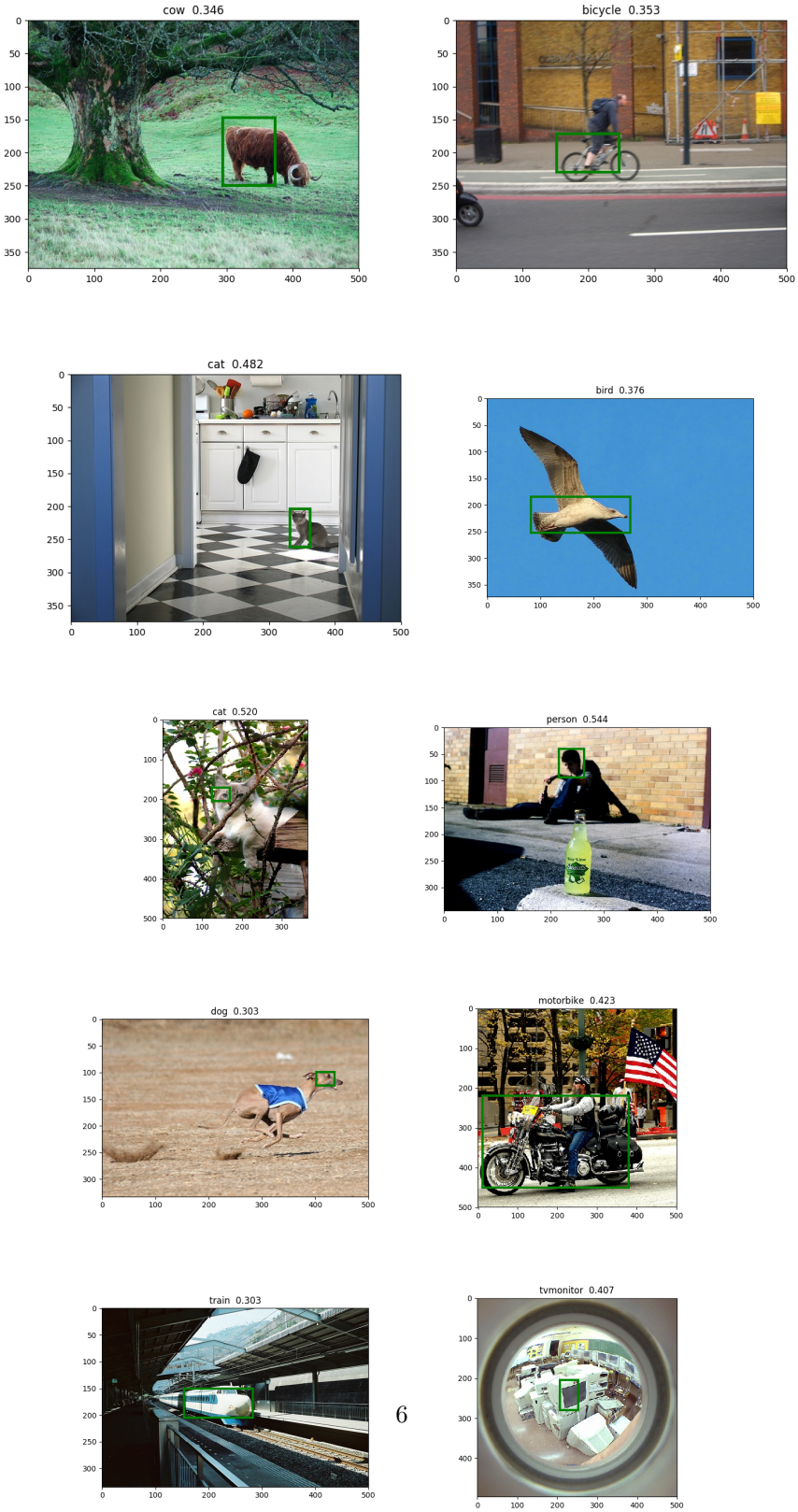


Figure 3: Output visualization of Single-scale WSDNN

Figure 4: Output visualization of Multi-scale WSDNN

of the subject. Since this area is small compared to the full target then the score would lower.

Overall, I was surprised about the results for a pre-trained weakly supervised model. The results show a step forward in the direction of training weakly supervised models by leveraging the learning task into smalled sub-tasks towards the same objective.