ASSIGNMENT 2
Weakly Supervised Object Localization

March 30, 2019

Name: Xinjia Yu
Andrew ID: xinjiay

Contents

0.1 Task 0: Visdom

Q 0.1: What classes does the image at index 2019 contain (index 2019 is the 2020-th image due to 0-based numbering)?

The class for index 2019 is 'motorbike'.

Q 0.2: What is the filename of the image at index 2019?

The name of the file for index 2019 is '004003.jpg'.

Q 0.3 Use Visdom+CV2 to visualize the top 10 bounding box proposals for image at index 2019. You would need to first plot the image and then plot the rectangles for each bounding box proposal.

Q 0.4 Use Visdom+CV2 to visualize the ground truth boxes for image at index 2019.

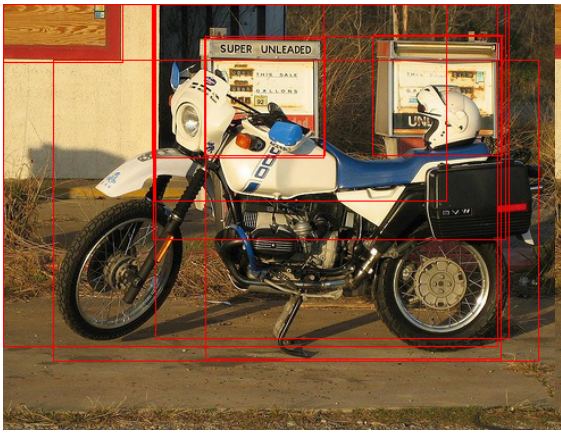


Figure 1: Top 10 bounding box proposals for index 2019

Figure 2: Ground for index 2019

0.2 Task 1: Is object localization free?

Q 1.1 In the report, for each of the TODO, describe the functionality of that part.

In **custom.py** file:

1. find_classes function is used to map classes to its index, the input is the database. returns are classes and their index.
2. make_dataset function is used to return a list of tuples, which includes the image path and the list of classes.
3. LocalizerAlexNet function is used to construct the defined model
4. LocalizerAlexNetRobust function is the refined version of LocalizerAlexNet function, which adds the dropout layers between the last two layers.
5. localizer_alexnet function is used to initialize the model defined by LocalizerAlexNet.
6. localizer_alexnet_robust function is used to initialize the model defined by LocalizerAlexNetRobust.
7. IMDBDataset.__getitem__ is the function which could return the tuple of image and labels if the image index is given. It is worth to mention that the class is in the vector version.

In **main.py** file:

1. Criterion was set to be BCELoss and optimizer was selected as SGD.
2. Seed was set to be 1 to avoid randomness and generate same pictures.
3. Metric 1 is the mAP criterion.
4. Metric 2 was selected as top 5 precision, which improve the robust of the evaluation.

Q 1.2 What is the output resolution of the model?

The output resolution of the model is 29 * 29.

Q 1.3 INITIALIZE THE MODEL FROM IMAGENET (TILL THE CONV5 LAYER), INITIALIZE THE REST OF LAYERS WITH XAVIER INITIALIZATION AND TRAIN THE MODEL USING BATCHSIZE=32, LEARNING RATE=0.01, EPOCHS=2 (YES, ONLY 2 EPOCHS FOR NOW).

See the loss curve below.

Q 1.4 IN THE FIRST FEW ITERATIONS, YOU SHOULD OBSERVE A STEEP DROP IN THE LOSS VALUE. WHY DOES THIS HAPPEN? (HINT: THINK ABOUT THE LABELS ASSOCIATED WITH EACH IMAGE).

At the beginning, the loss is a pretty high value because most of the predictions are wrong. Since we are using BCELoss in this task and the target labels are pretty sparse, the model tends to predict all labels to be zero at the beginning which may output could fastly converge near to the target value, and thus the loss would show a steep drop.

Q 1.5 Metric 1 and Metric 2

For metric 1, we are using simple mAP to evaluate the network performance, which is not quite robust. Therefore, for metric 2, we are going to use Top 5, which evaluates the probability of top 5 classes contains ground truth. Top 5 metric could increase the robustness of the evaluation. Details for implementation please see code.

Q 1.6 INITIALIZE THE MODEL FROM IMAGENET (TILL THE CONV5 LAYER), INITIALIZE THE REST OF LAYERS WITH XAVIER INITIALIZATION AND TRAIN THE MODEL USING BATCHSIZE=32, LEARNING RATE=0.01, EPOCHS=30. EVALUATE EVERY 2 EPOCHS.

In 1.6, I have added the visualization module including using visdom and tensoboardx. That is all I added in to the train.py file.

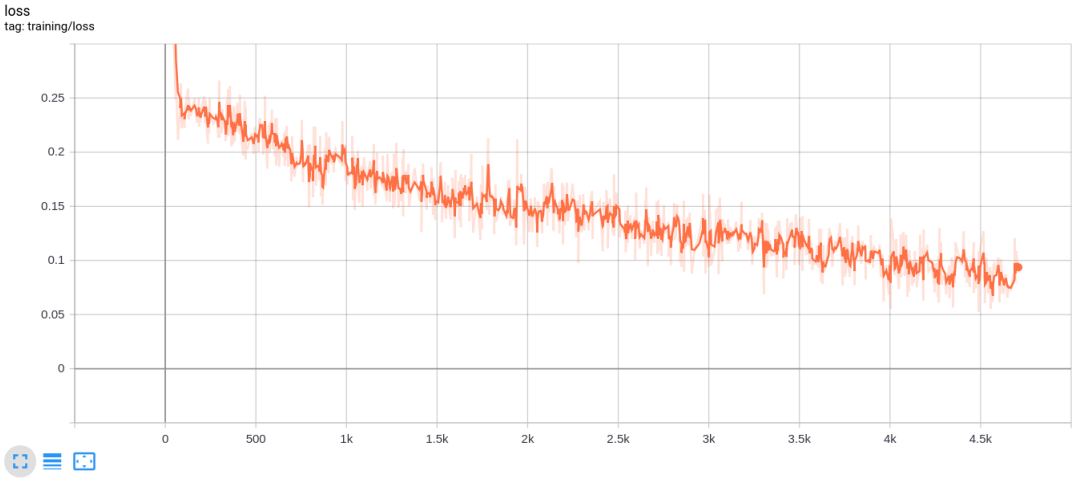


Figure 3: Loss for alexnet

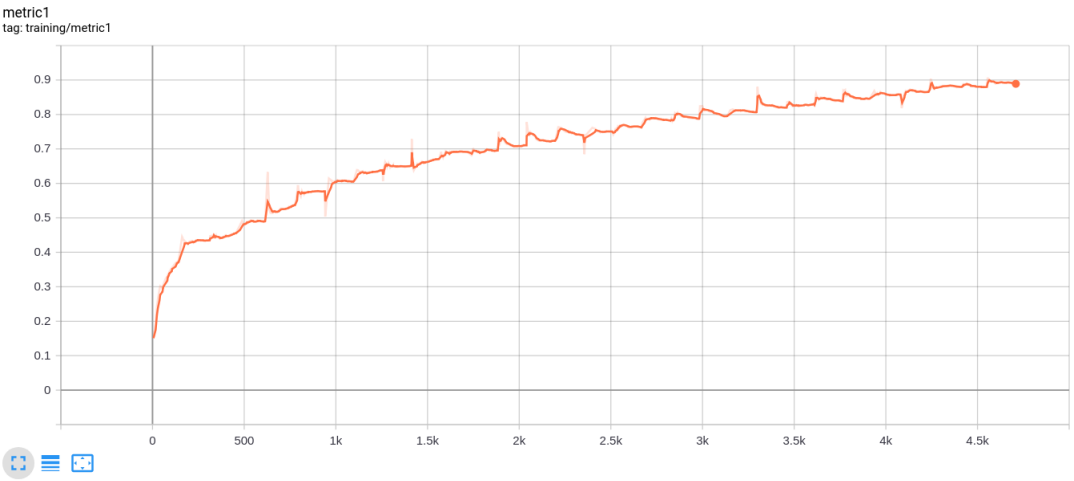


Figure 4: Metric 1 for alexnet in training set

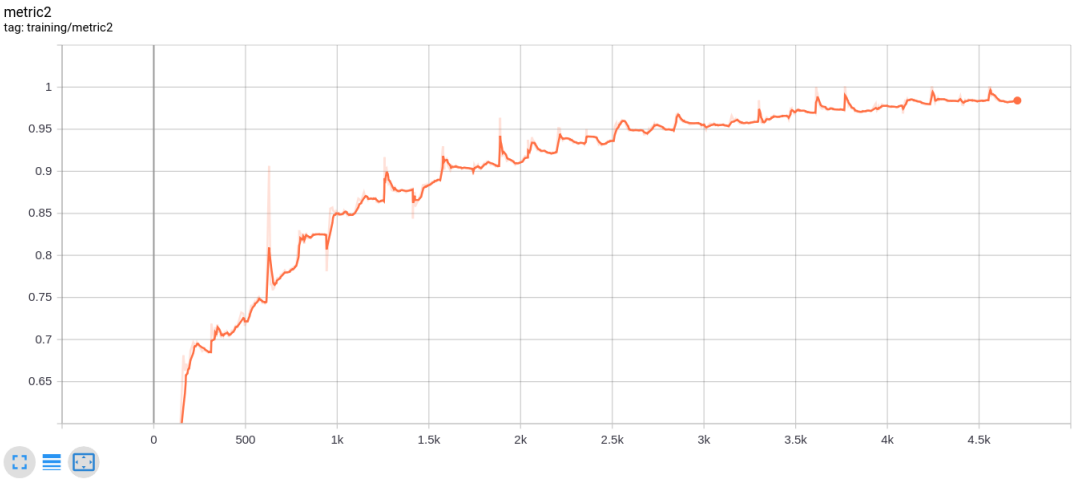


Figure 5: Metric 2 for alexnet in training set

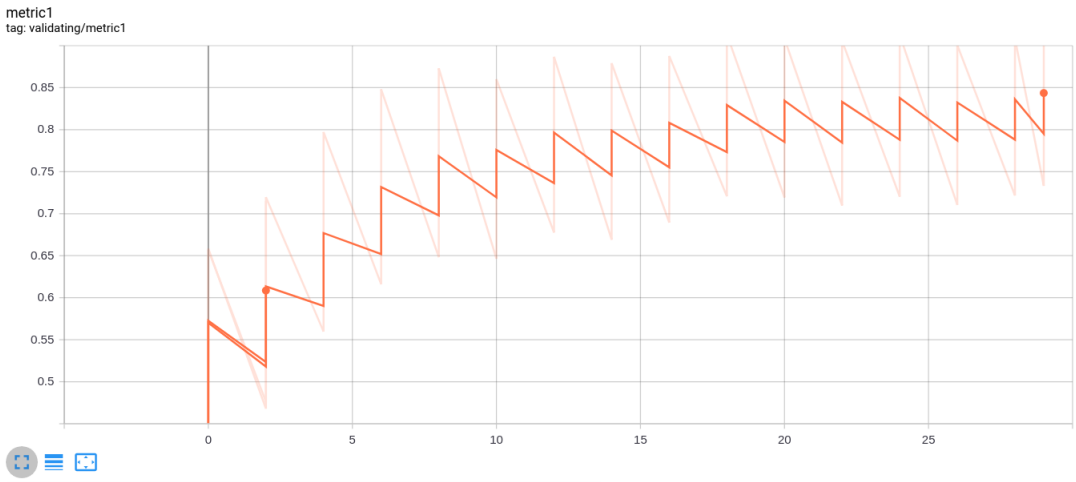


Figure 6: Metric 1 for alexnet in validating set

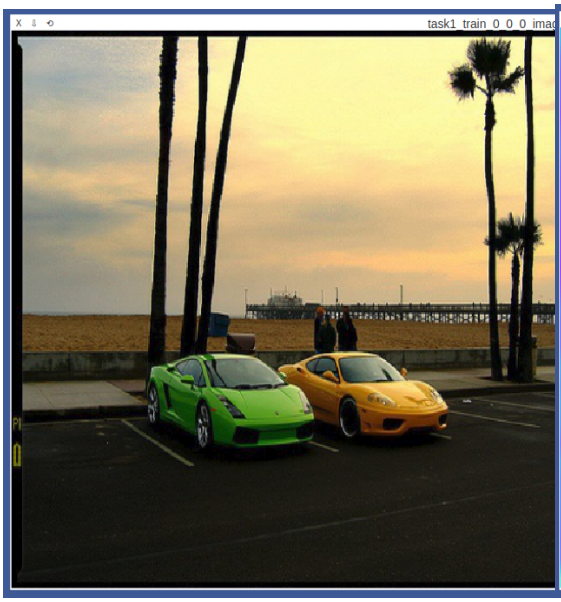


Figure 7: image

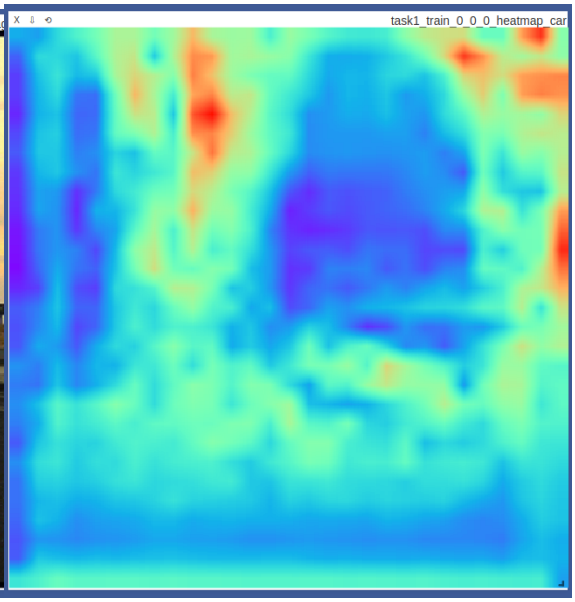
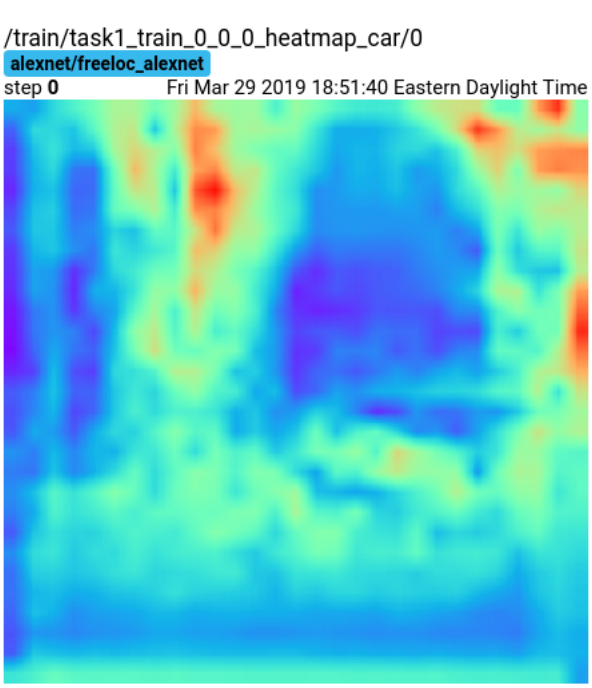


Figure 8: Ground truth



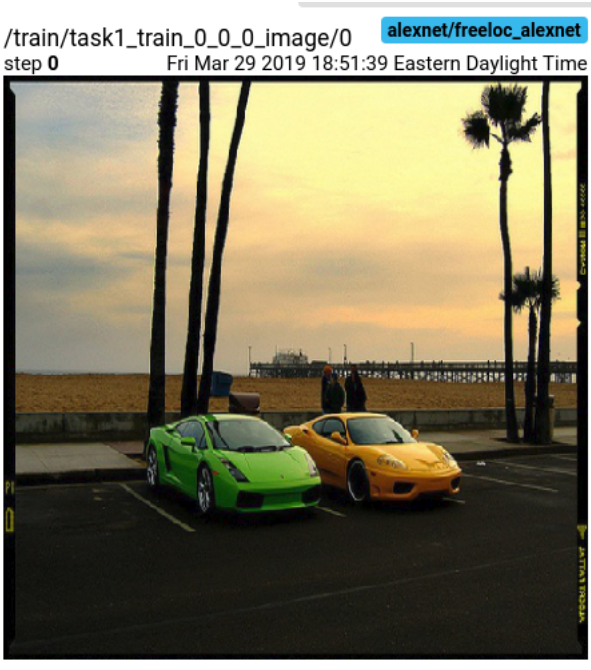


Figure 9: Image and Heatmap

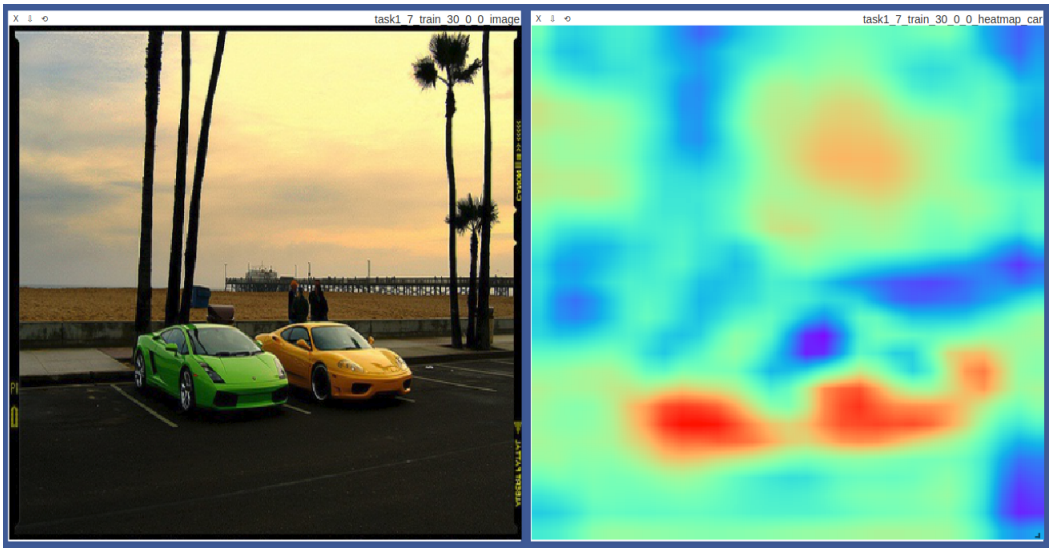


Figure 10: Image and Heatmap (30 epochs)

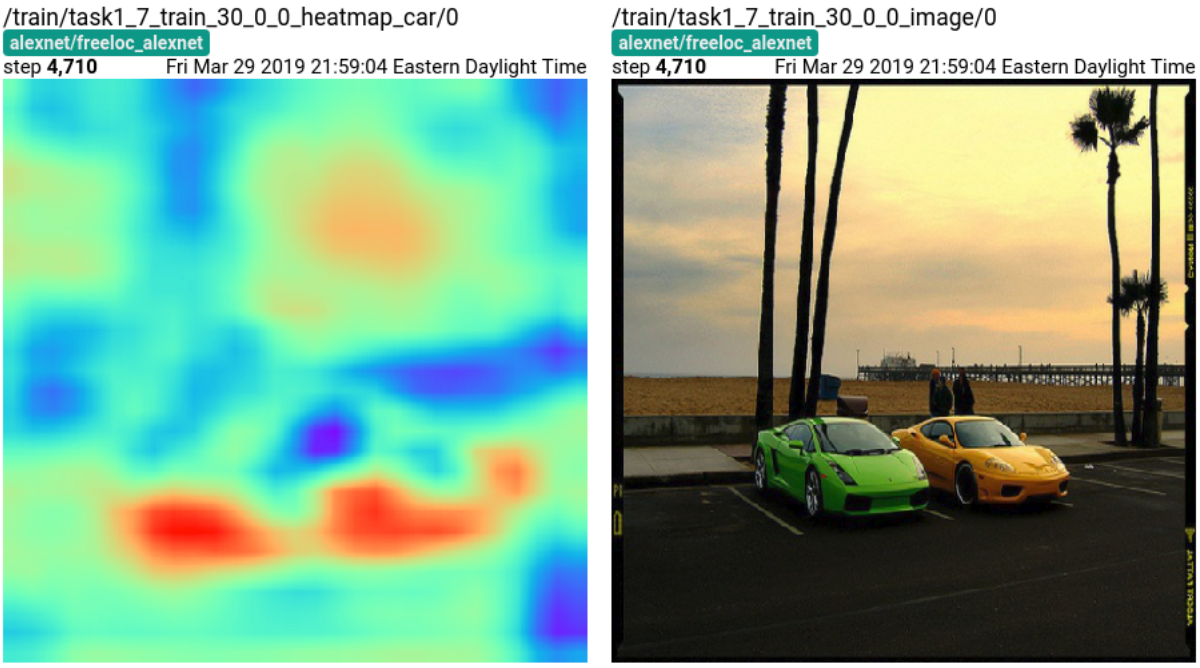


Figure 11: Image and Heatmap (30 epochs)

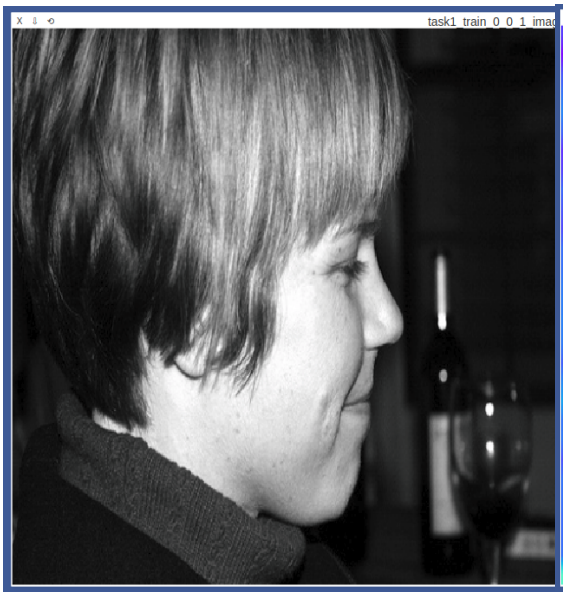


Figure 12: image

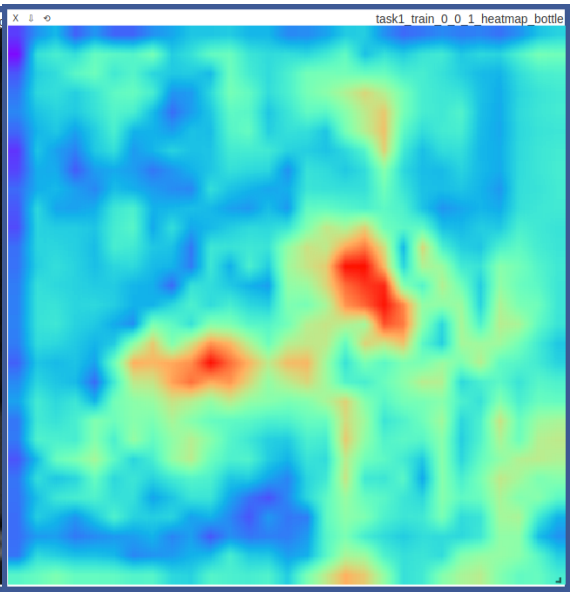


Figure 13: Ground truth

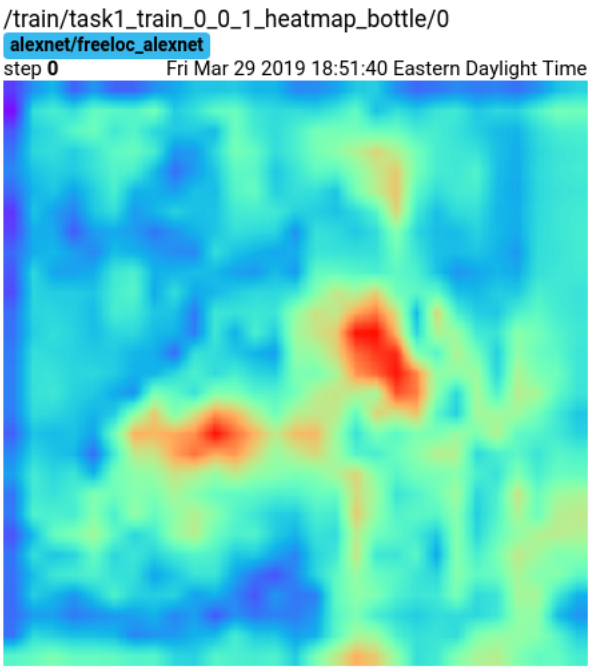


Figure 14: Image and Heatmap

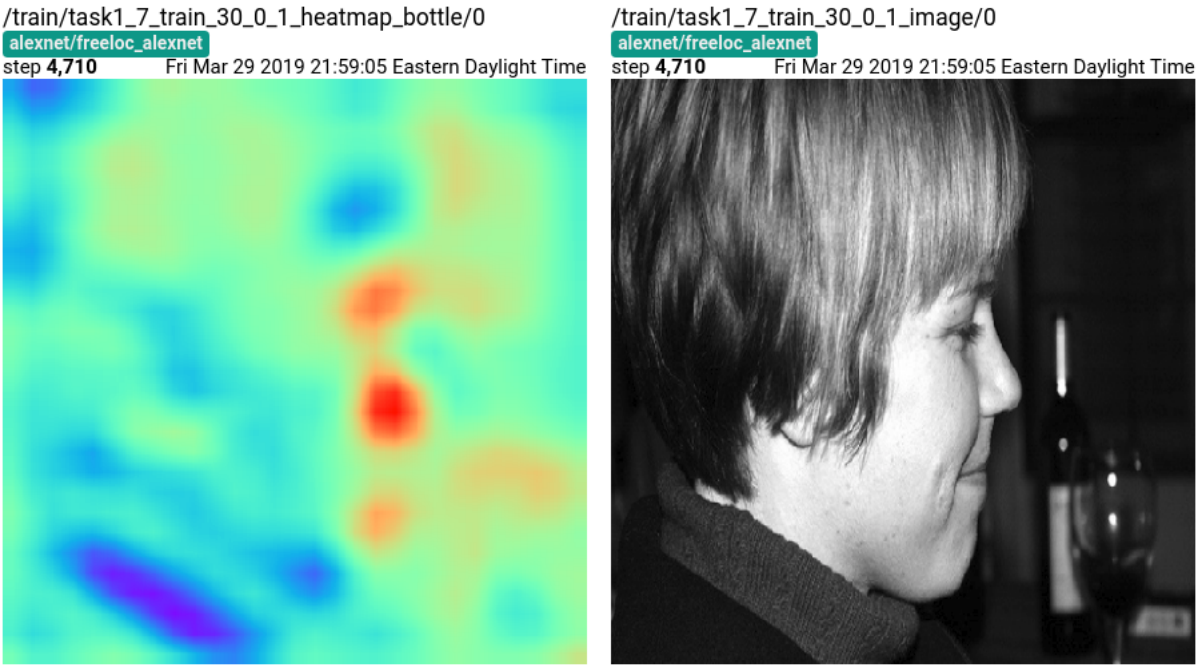


Figure 15: Image and Heatmap(30 epochs)

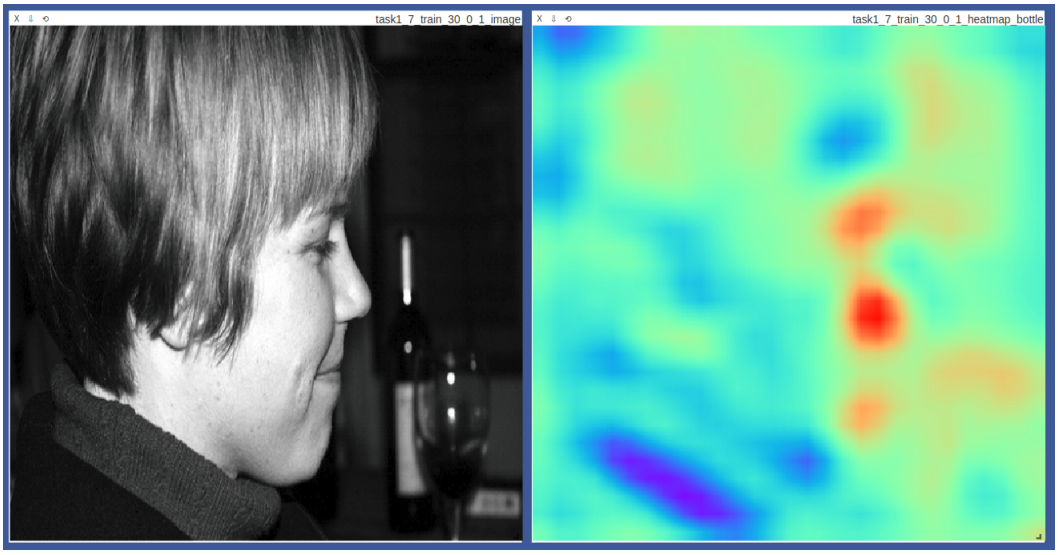


Figure 16: Image and Heatmap (30 epochs)

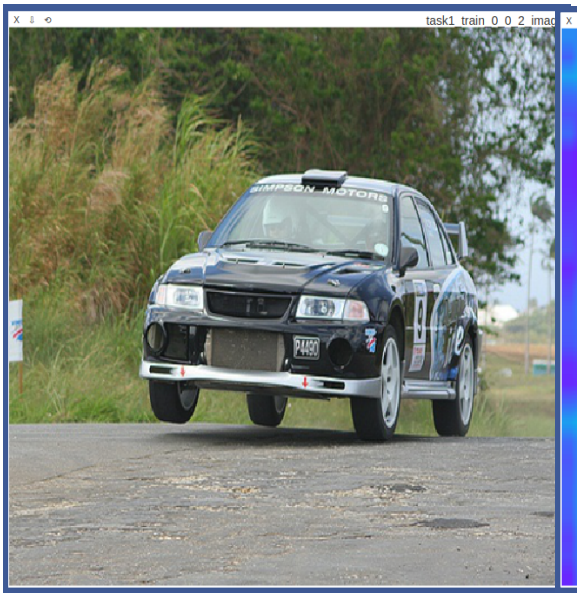


Figure 17: image

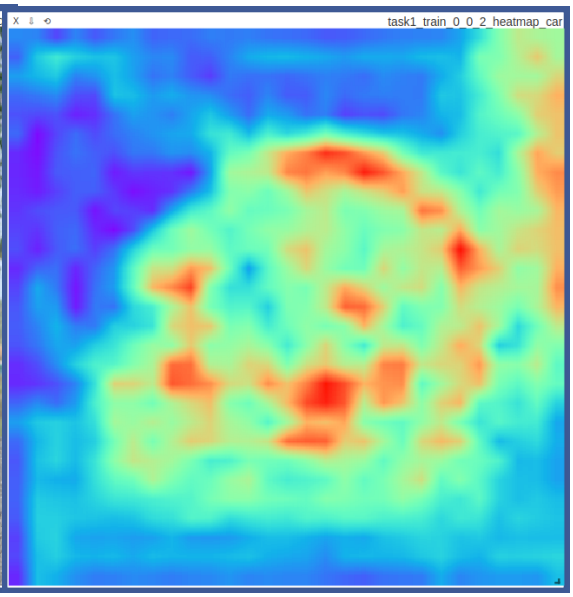


Figure 18: Ground truth

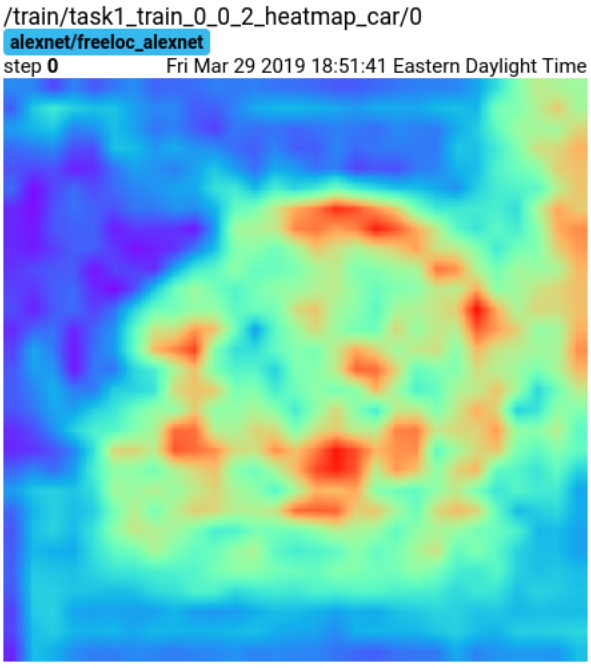


Figure 19: Image and Heatmap

Figure 20: Image and Heatmap (30 epochs)

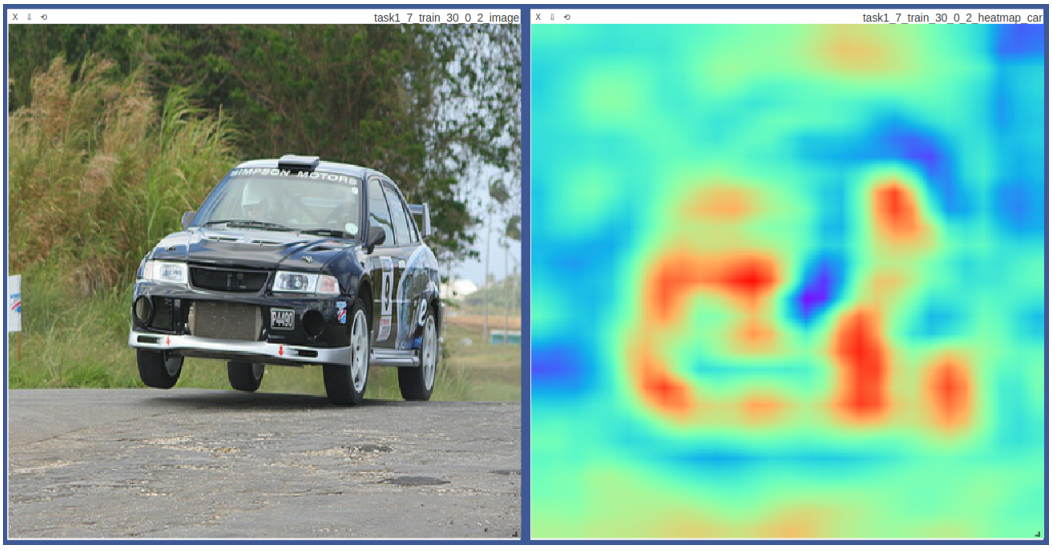


Figure 21: Image and Heatmap (30 epochs)

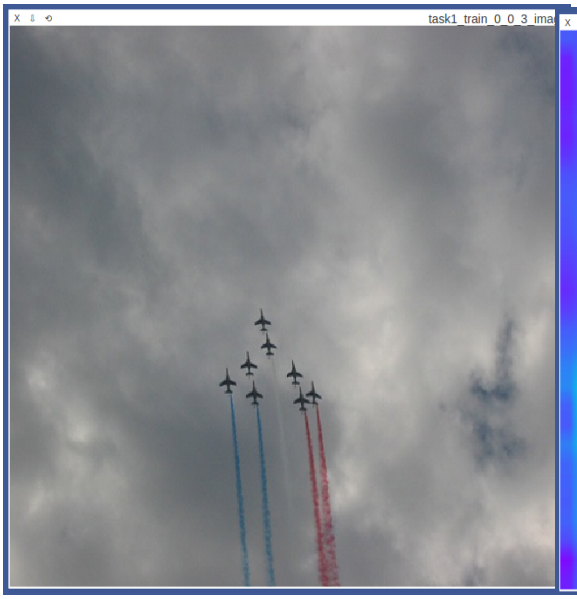


Figure 22: image

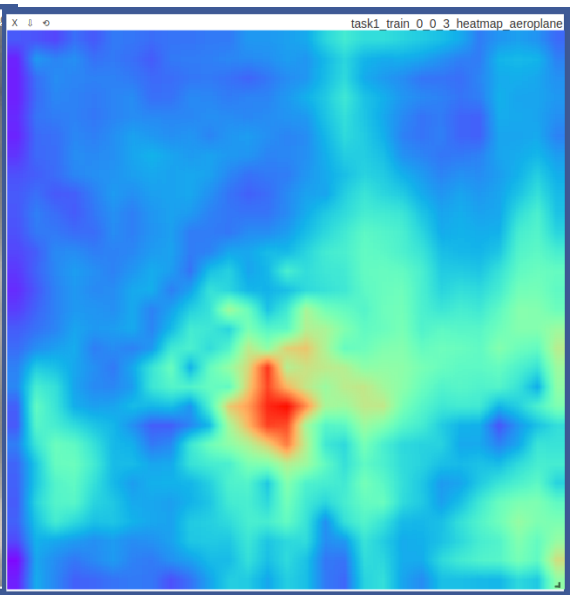
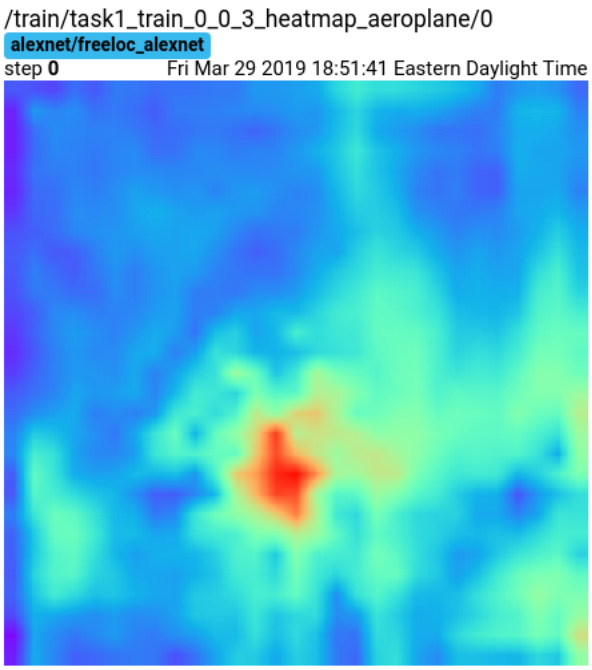


Figure 23: Ground truth



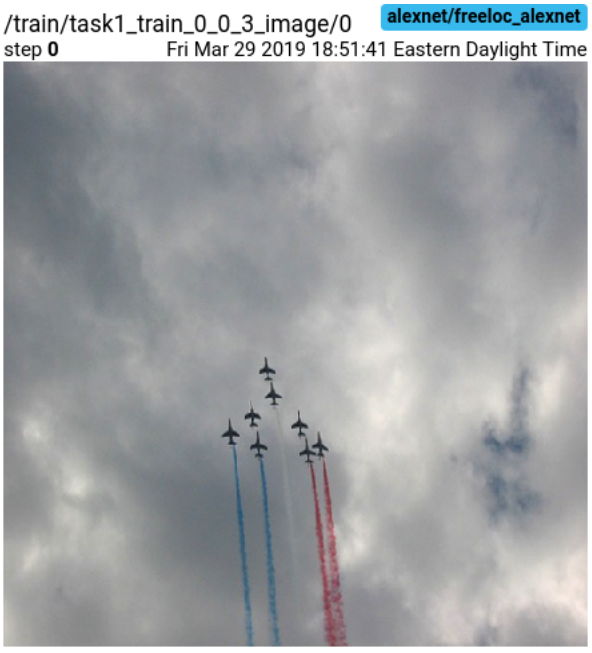


Figure 24: Image and Heatmap

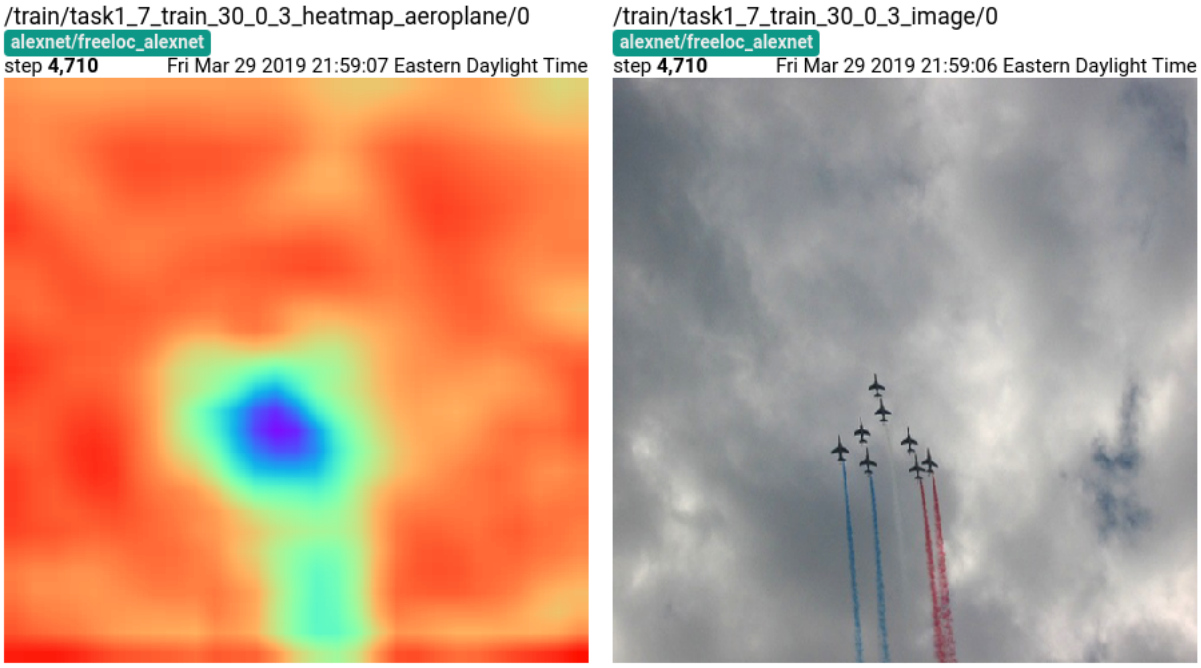


Figure 25: Image and Heatmap (30 epochs)

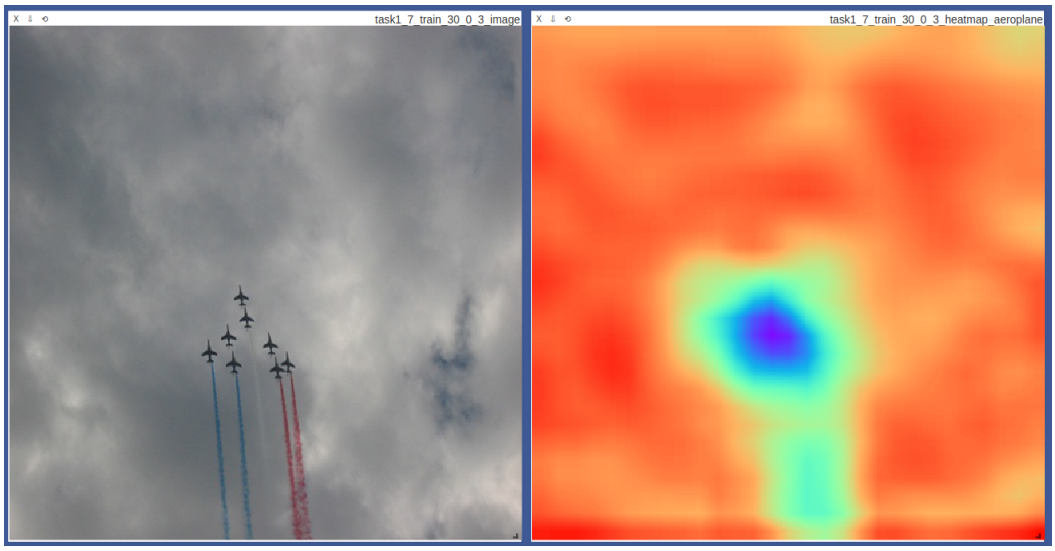


Figure 26: Image and Heatmap (30 epochs)

Loss: 0.0565, Metric 1 for test set: 0.732, Metric 2 for test set: 0.910.

```
* Metric1 0.732 Metric2 0.910
```

Q 1.7 IN THE HEATMAP VISUALIZATIONS YOU OBSERVE THAT THERE ARE USUALLY PEAKS ON SALIENT FEATURES OF THE OBJECTS BUT NOT ON THE ENTIRE OBJECTS. HOW CAN YOU FIX THIS IN THE ARCHITECTURE OF THE MODEL?

In 1.7, I changed nothing of the code, except the agreement for architecture. I changed it from localizer_alexnet to localizer_alexnet_robust. That was all what I did for 1.7.

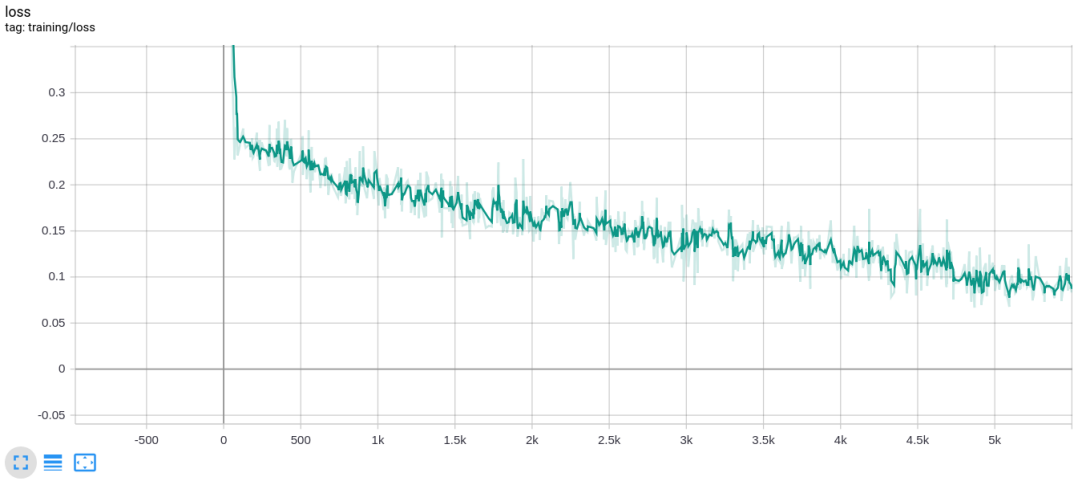


Figure 27: Loss for alexnetrobust

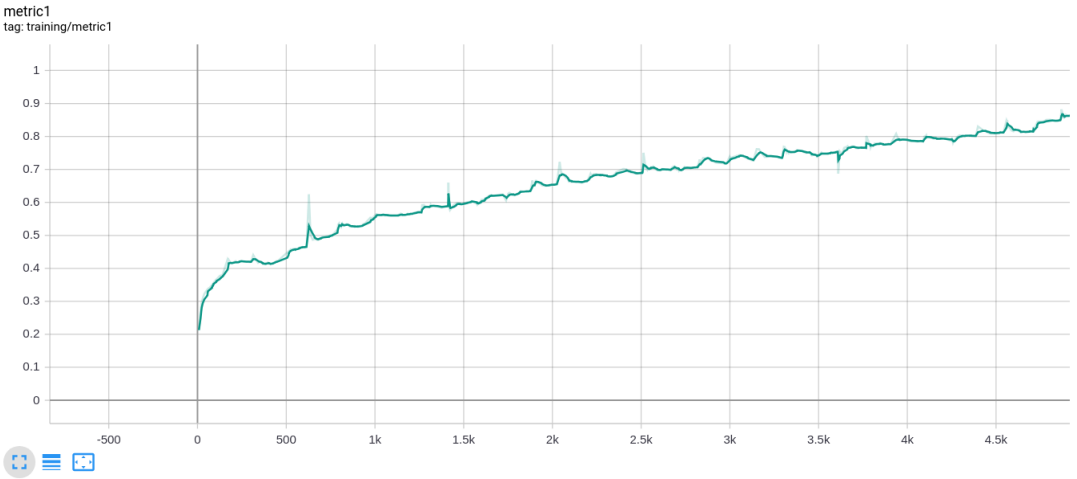


Figure 28: Metric 1 for alexnetrobust in training set

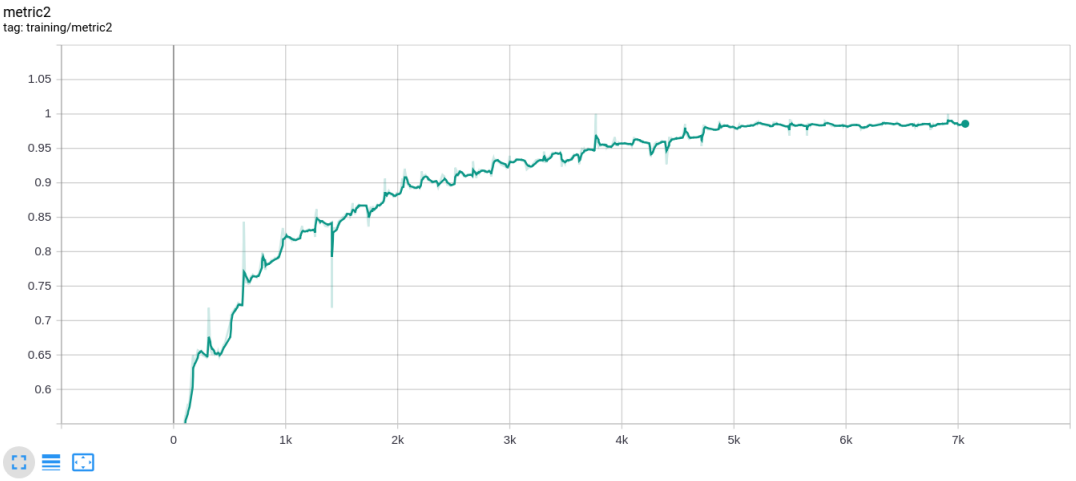


Figure 29: Metric 2 for alexnetrobust in training set

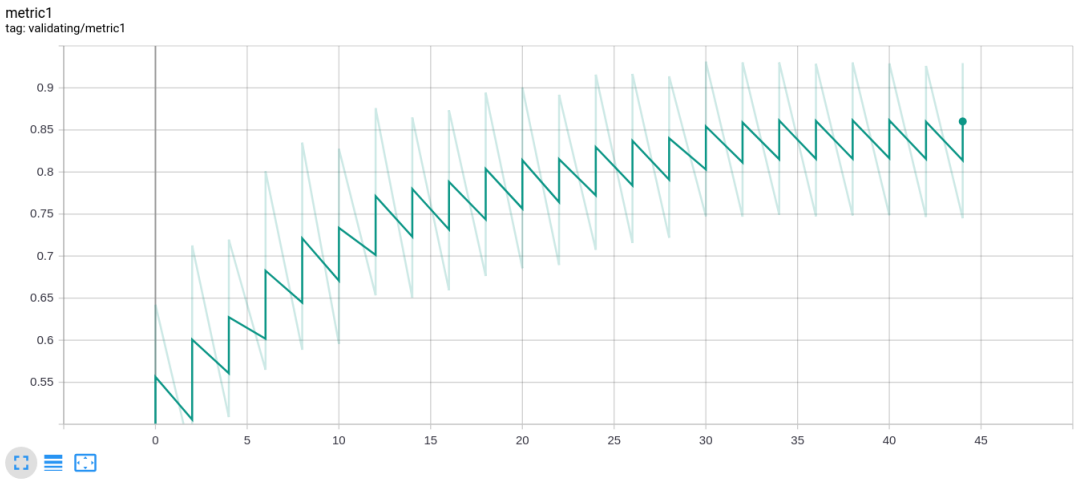


Figure 30: Metric 1 for alexnetrobust in validating set

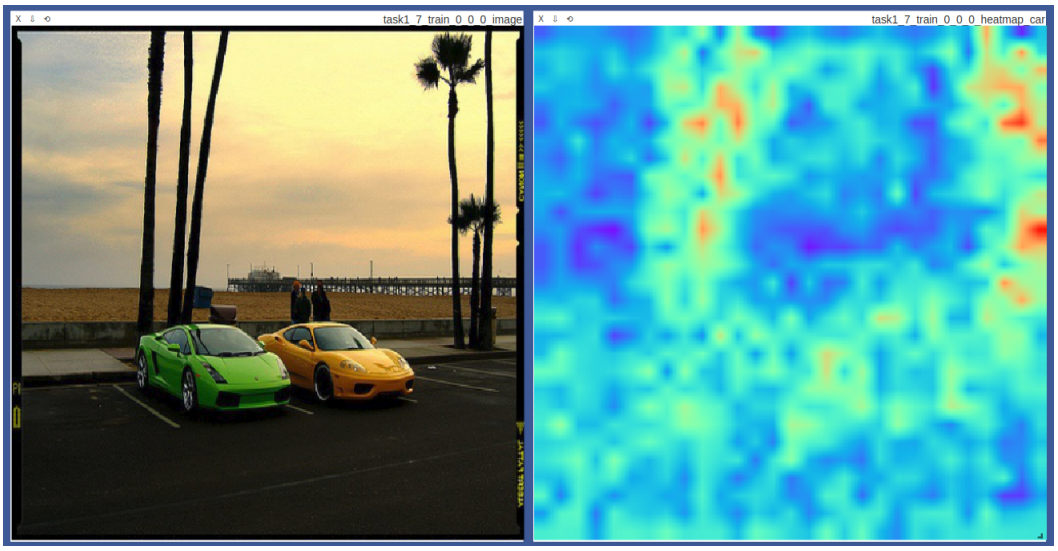


Figure 31: Image and Heatmap

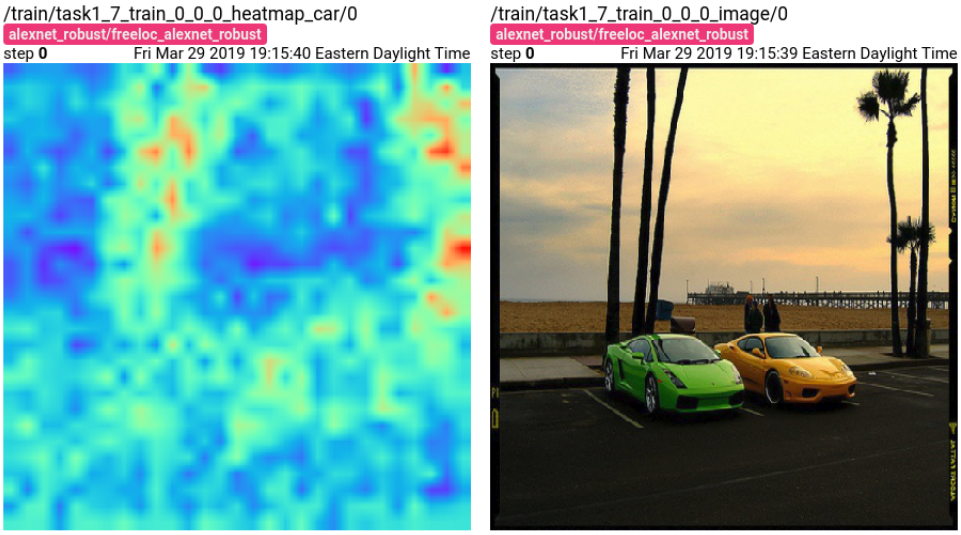


Figure 32: Image and Heatmap

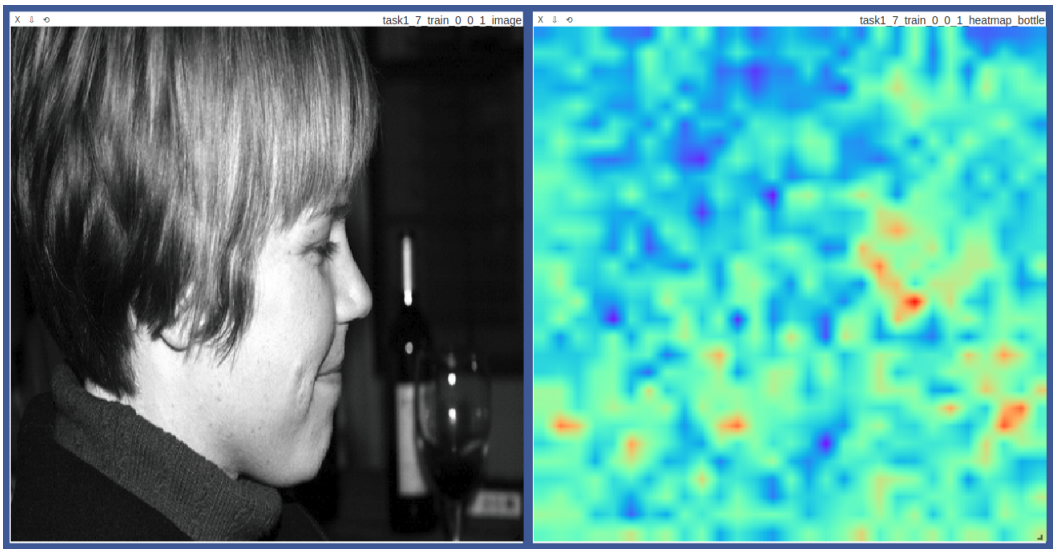


Figure 33: Image and Heatmap

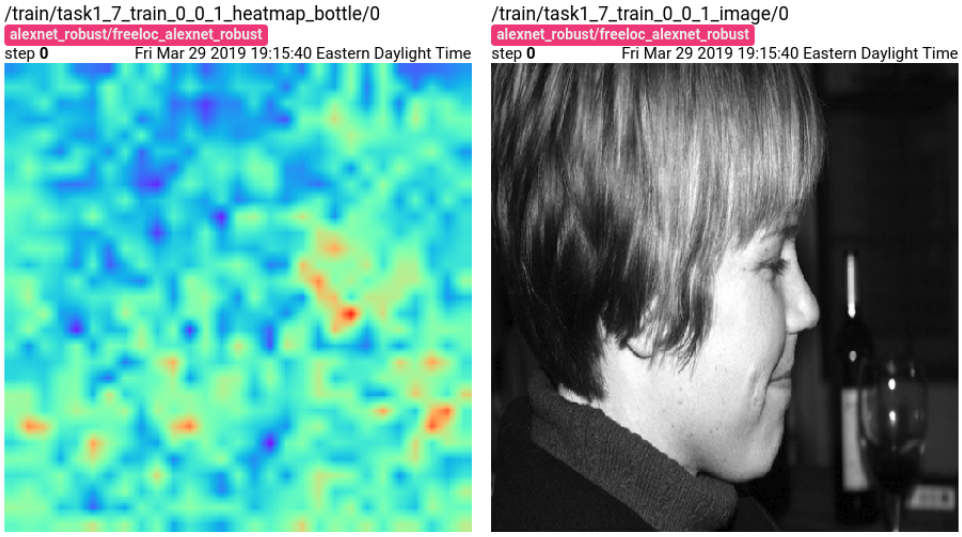


Figure 34: Image and Heatmap

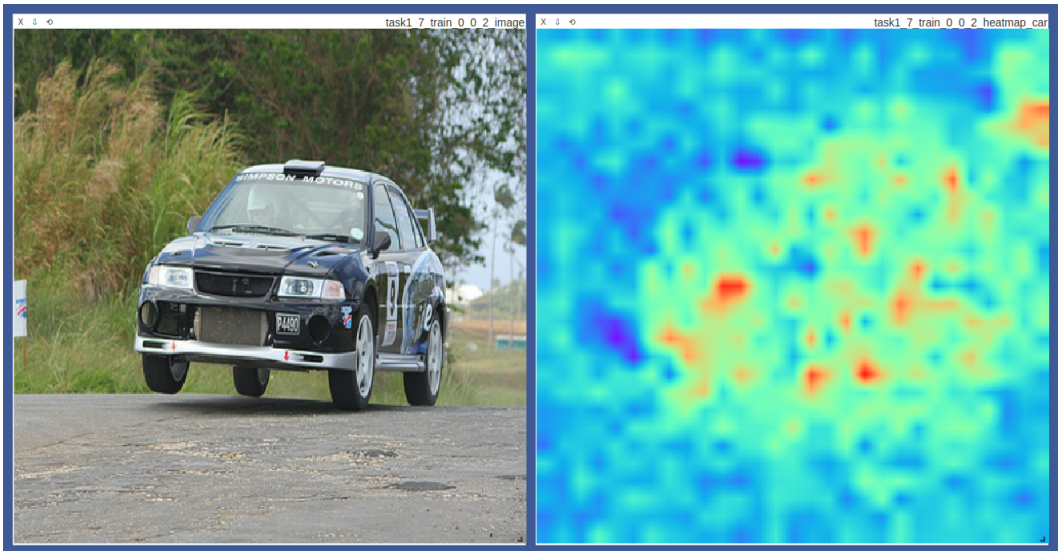


Figure 35: Image and Heatmap

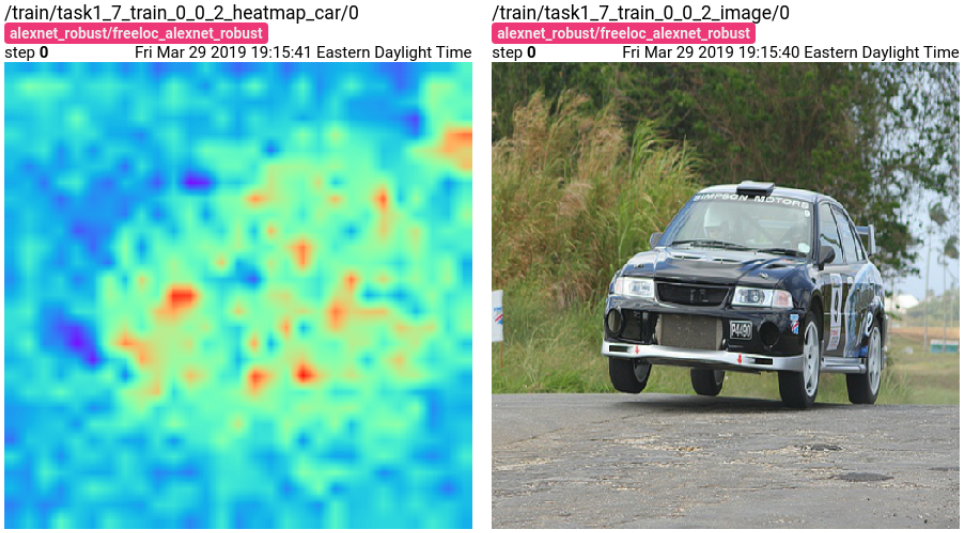


Figure 36: Image and Heatmap

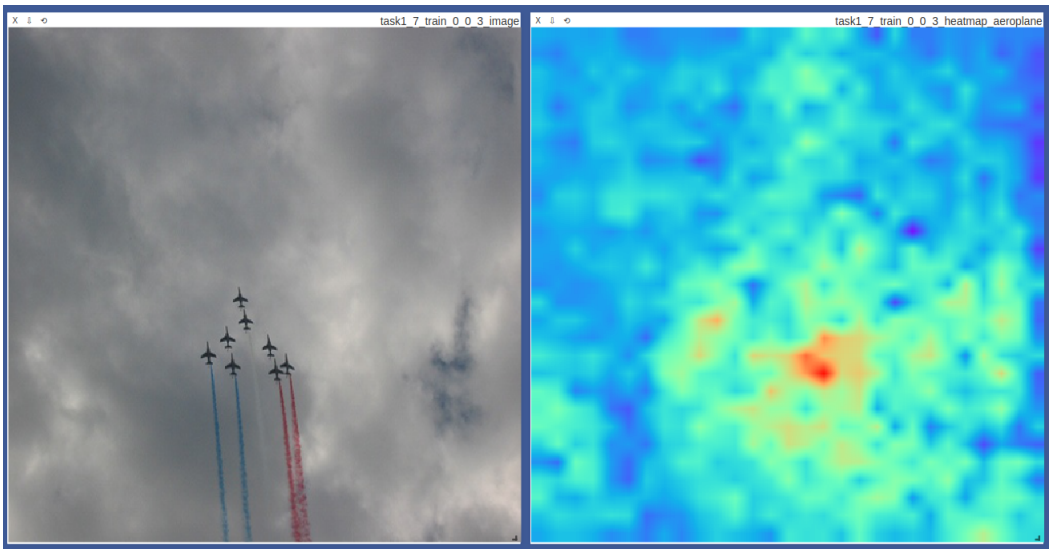


Figure 37: Image and Heatmap

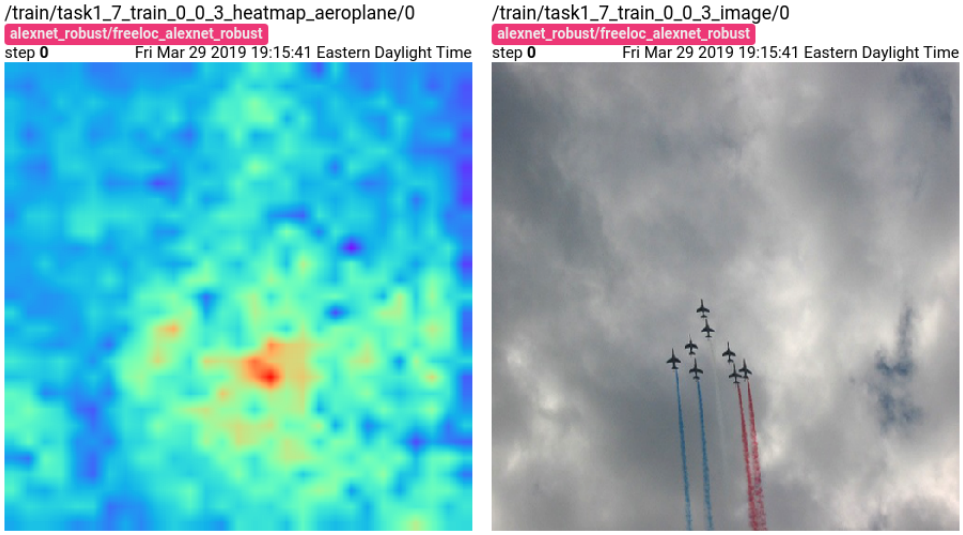


Figure 38: Image and Heatmap

Loss: 0.0752, Metric 1 for test set: 0.755, Metric 2 for test set: 0.936.

```
* Metric1 0.755 Metric2 0.936
```

0.3 Task 2: Weakly Supervised Deep Detection Networks

Q2.4 train the model using the settings provided in experiments/cfgs/wsddn.yml for 30000 iterations.

I made a stupid mistake that I forgot to record mAP in my first running. Therefore I only recorded the loss for 30000 steps. And for the mAPs, I could only record for 20000 steps because the code is too slow and I do not have enough time to have a full training. But I am confident that even for 20000 steps, you could still notice that my result is reasonable.

Loss curve from visdom:

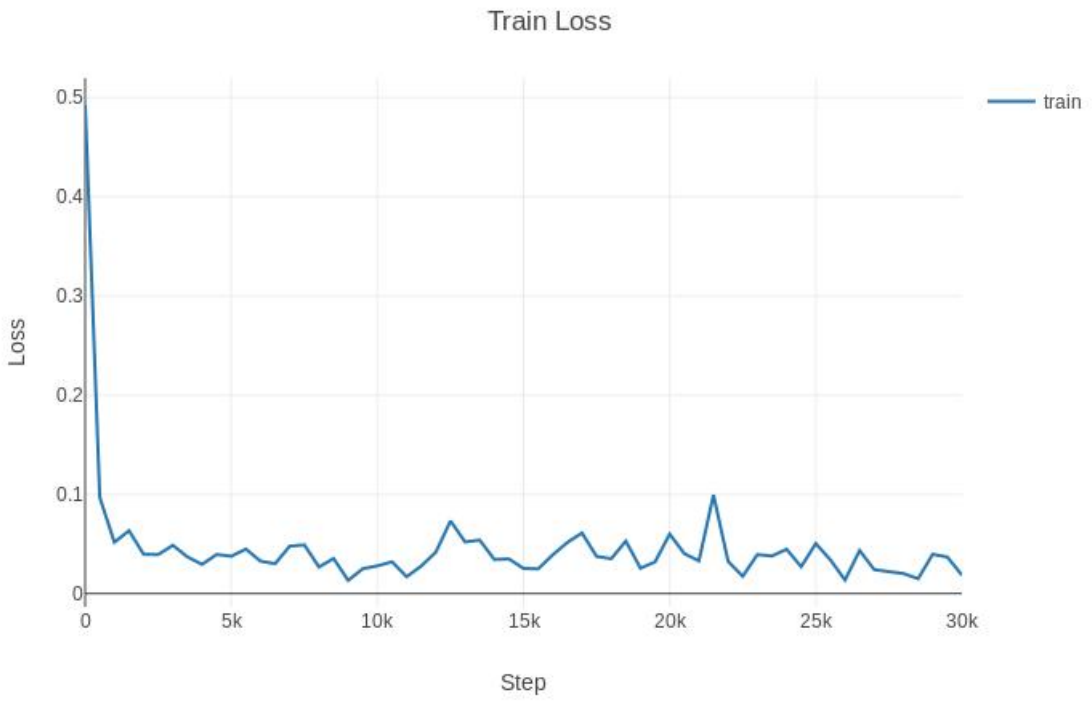


Figure 39: Loss curve for WSDDN

mAP curve from visdom:

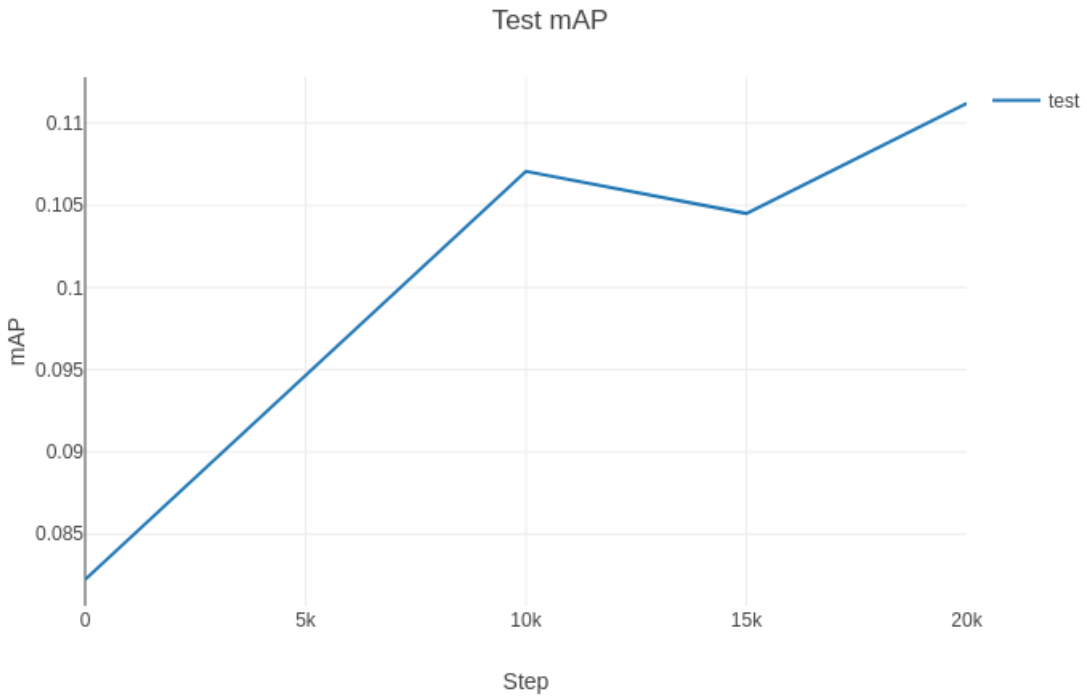


Figure 40: Test mAP curve for WSDDN in 20000 training steps

Loss curve from tensorboard:

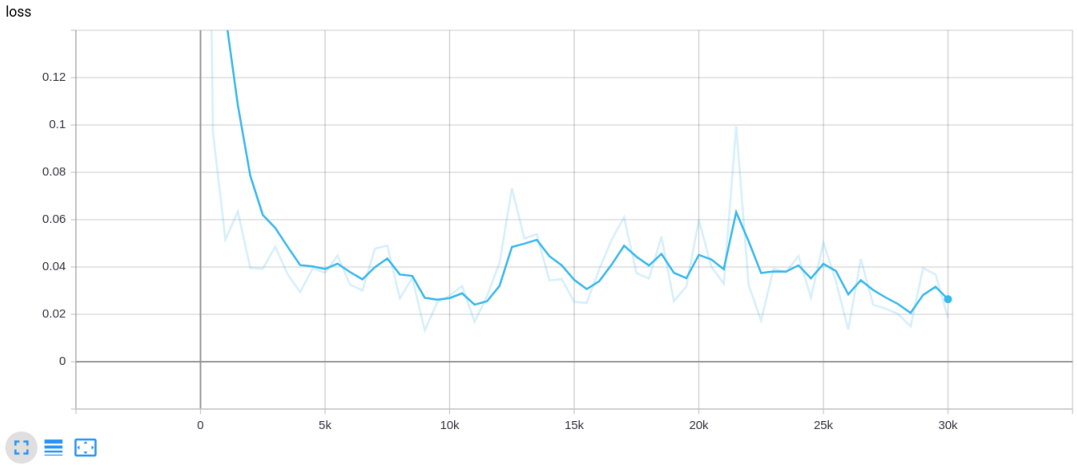


Figure 41: Loss curve for WSDDN

mAP curve from tensorboard:

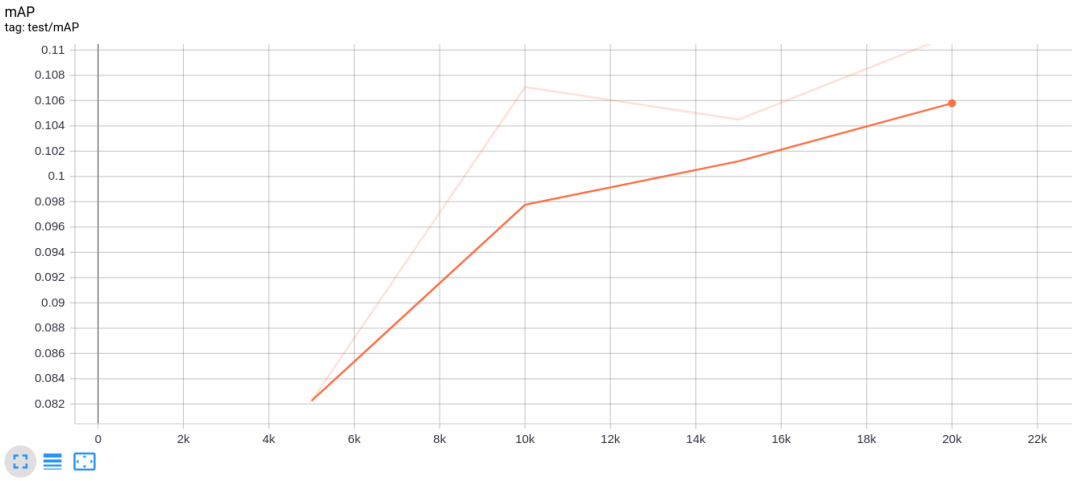


Figure 42: Test mAP curve for WSDDN in 20000 training steps

Histogram of gradients of weights for conv1, conv2 and fc7:

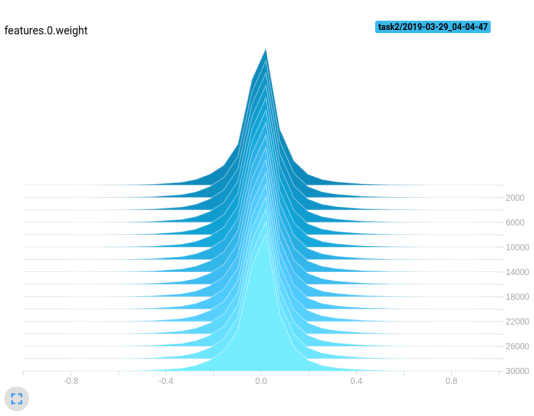


Figure 43: Histogram for conv1

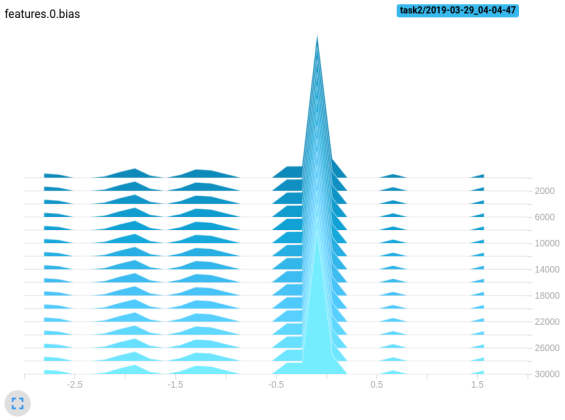


Figure 44: Histogram for conv1

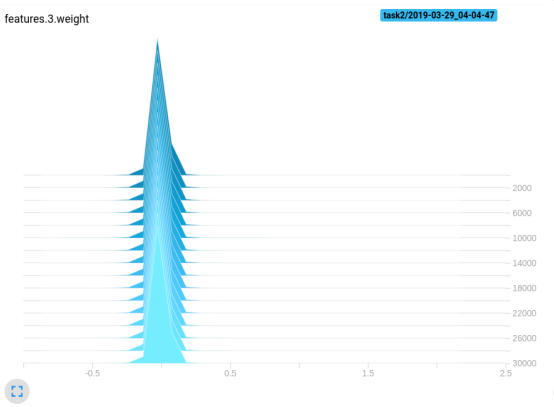


Figure 45: Histogram for conv2

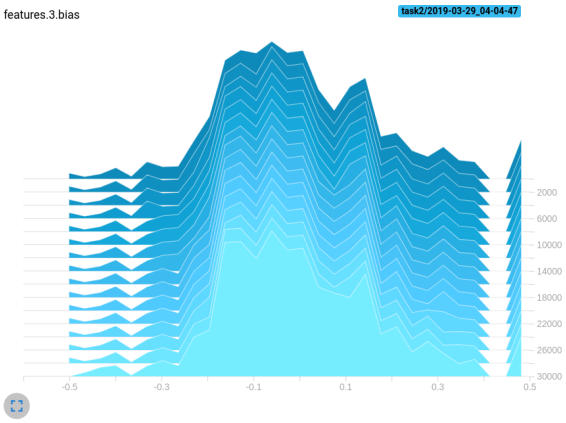


Figure 46: Histogram for conv2

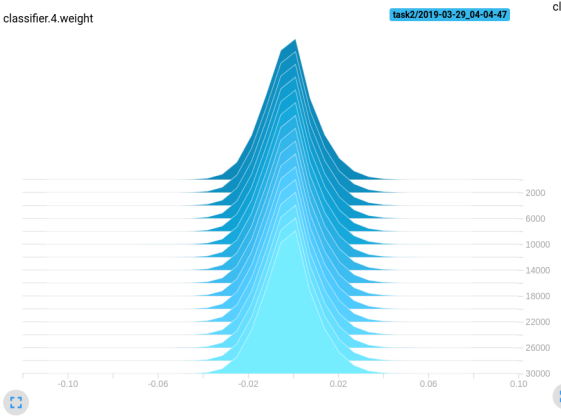


Figure 47: Histogram for fc7

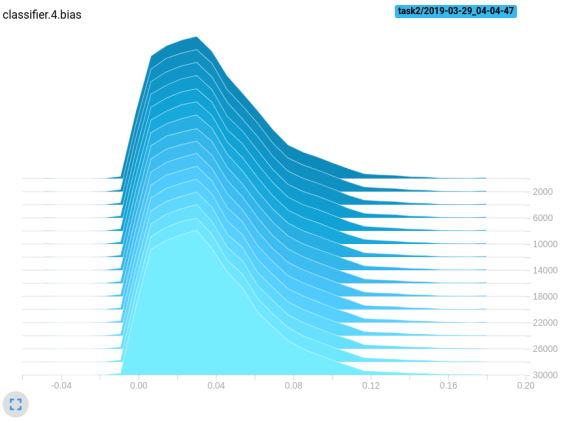


Figure 48: Histogram for fc7

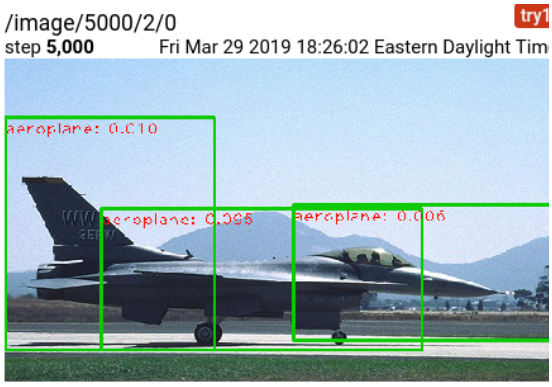


Figure 49: 5000 steps

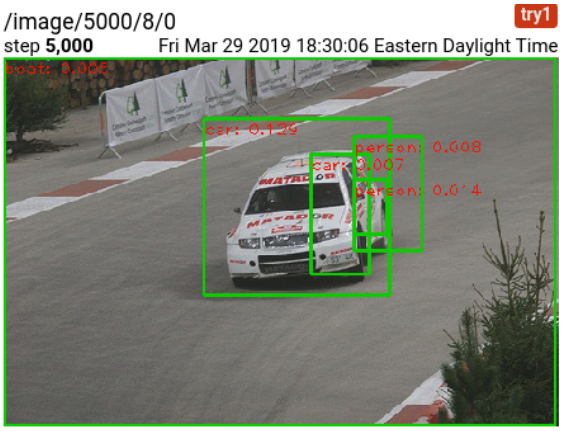


Figure 50: 5000 steps

Figure 51: 5000 steps

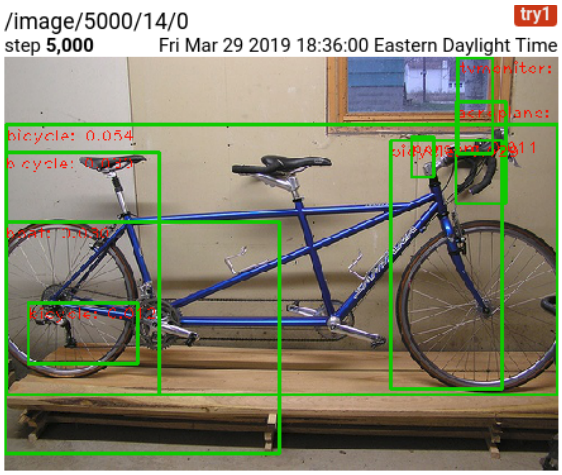


Figure 52: 5000 steps

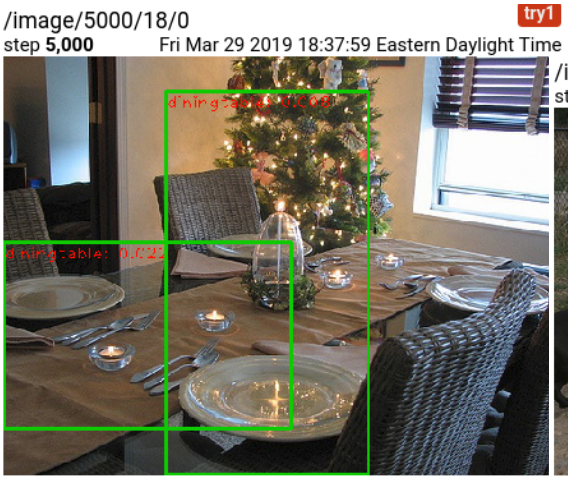


Figure 53: 5000 steps

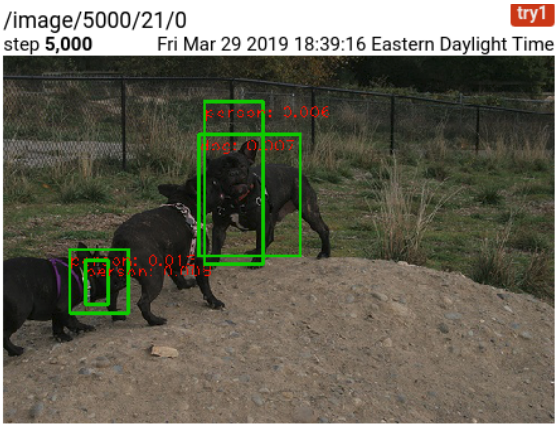


Figure 54: 5000 steps

For 20000 steps, I made a small mistake that the step number should be 20000 instead of 5000. But as you can notice that for 20000steps, the prediction bounding box is much more accurate than 5000 steps.

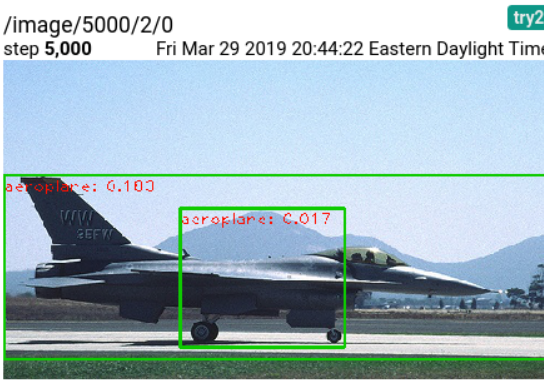


Figure 55: 20000 steps

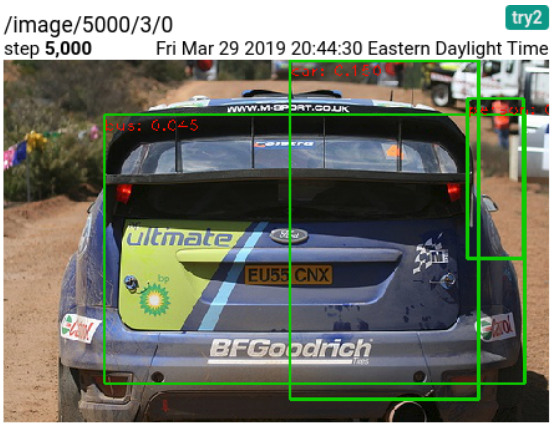


Figure 56: 20000 steps

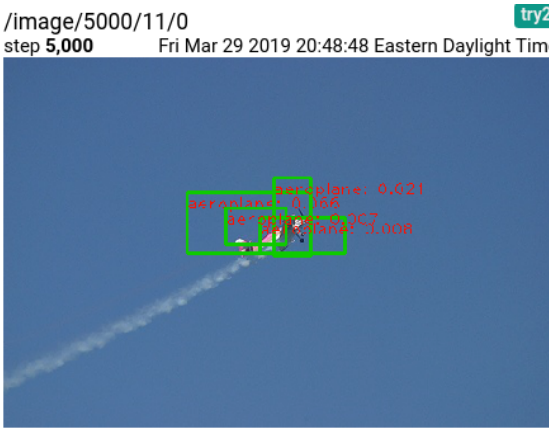


Figure 57: 20000 steps

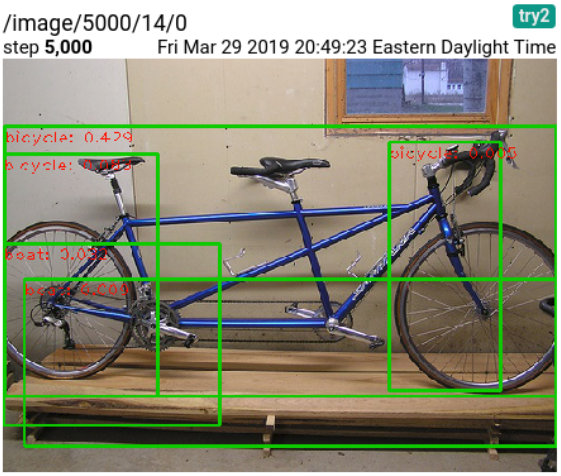


Figure 58: 20000 steps

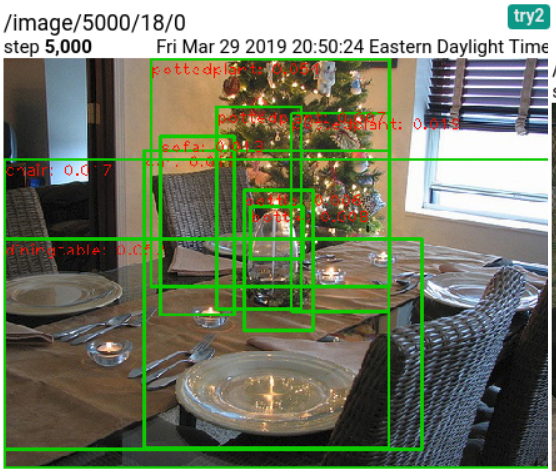


Figure 59: 20000 steps

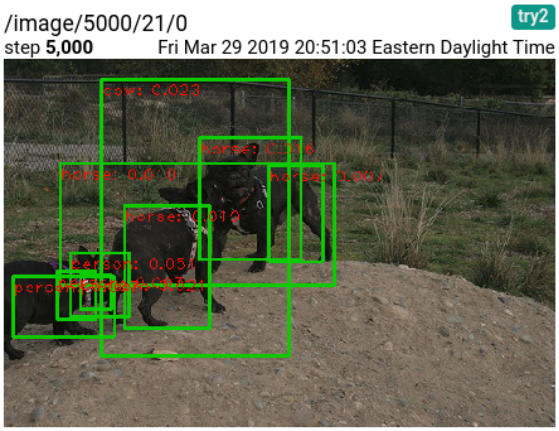


Figure 60: 20000 steps

mAPs for certain class:

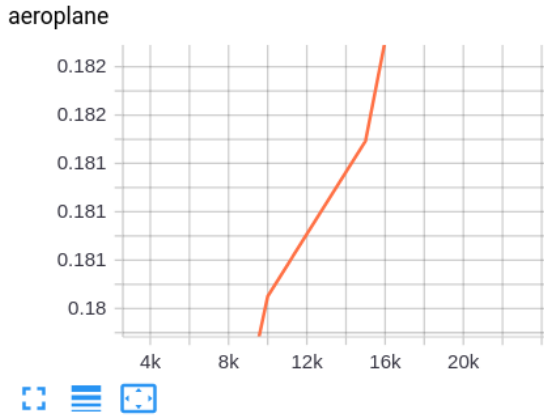


Figure 61: aeroplane

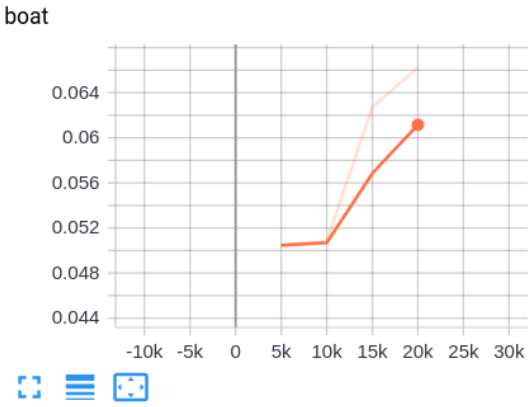


Figure 62: boat

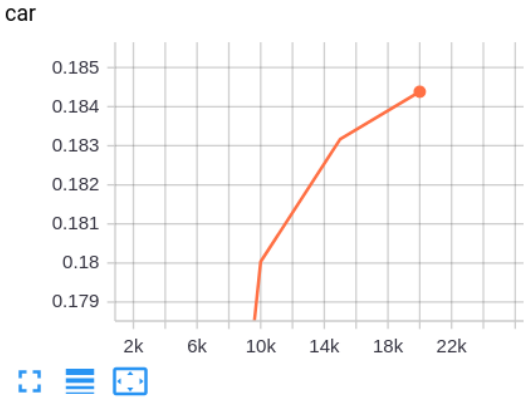


Figure 63: car

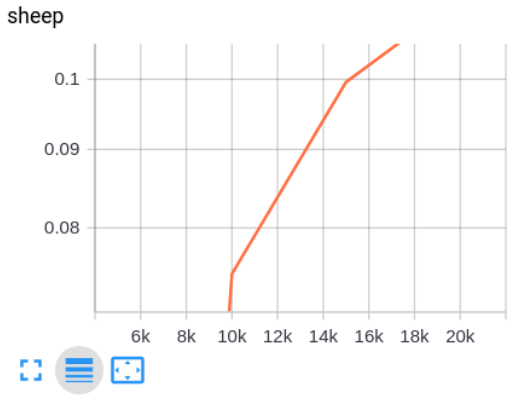


Figure 64: sheep

The final results for APs for 20000 steps:

classes	**mAP**
aeroplane	0.191
bicycle	0.138
bird	0.131
boat	0.066
bottle	0.067
bus	0.159
car	0.186
cat	0.050
chair	0.030
cow	0.085
dining table	0.033
dog	0.062
horse	0.143
motorbike	0.216
person	0.040
potted plant	0.076
sheep	0.129
sofa	0.119
train	0.179
tv monitor	0.124
mAP	0.112