Geospatial Big Data Analysis - Lab 5

Konstantinos Papadakis
National Technical University of Athens
k.i.papadakis@gmail.com

Abstract

In this document we train and evaluate a single-stage and a two-stage object detector on the Batteries, Dice, and Toy Cars dataset, and answer several questions regarding object detection deep learning models.

1 The Dataset

The dataset contains images which are annotated by rectangular regions containing a battery, dice, toy-car, candle, highlighter, or spoon. It is split into train, validation and test subsets. In Table 1 the total number of objects in each split is displayed.

	Battery	Dice	Toy-car	Candle	Highlighter	Spoon
Train	830	768	647	101	90	46
Validation	42	65	59	0	0	0
Test	52	62	49	0	0	0

Table 1: Class representation across the images of the dataset

2 Object Detection

We train, evaluate and compare a two stage detector in the form of Faster R-CNN and a single stage detector in the form of RetinaNet. Both models use a ResNet50 pre-trained on ImageNet to create their feature pyramid network. Also, in both models, the trainable layers of the ResNet50 are the final 3 out of the 5 total.

During training, we load images in batches of size 8, we augment them by sequentially applying each one of the following transforms with probability 0.5:

- A horizontal flip.
- A random rotation by a right angle, zero or more times.
- A random change in brightness and in contrast by a factor in $[-0.2, 0.2]$.

2.1 One Stage Detector (RetinaNet)

We train a RetinaNet [1] with a setting of 110 epochs maximum and early stopping. More specifically, during training, every half epoch we evaluate the mean average precision (mAP) on the validation set. If the mAP didn't improve for 30 consecutive half-epochs, we stop the training early. Early stopping did happen on epoch 103, meaning that the best weights were achieved on epoch 89. A learning rate of 0.0001 was used.

2.2 Single Stage Detector (Faster R-CNN)

We train a Faster R-CNN using the 4-Step Alternating Training described in the original paper [2].

1. In step-1 we initialize and train a model that is composed of just the backbone and the RPN. When step-1 is finished, we call the model to create and cache proposals which we will use in step-2.
2. In step-2 we initialize a model composed of a *new* backbone and the ROI Heads. The proposals that we use are the ones cached from step-1.
3. In step-3 the model is composed of the step-2 backbone and the step-1 RPN. We update only the RPN weights and keep the backbone frozen.
4. In step-4 the model is composed of all the Faster R-CNN parts (Backbone, RPN, ROI Heads). We update only the ROI Heads weights and keep the backbone and the RPN frozen.

Steps 1, 2 and 3 were trained for 20 epochs. Step-4 was trained with a setting of 50 epochs maximum, and with the same early stopping strategy that was used for RetinaNet. Early stopping did happen on epoch 32, meaning that the best weights were achieved on epoch 17. In all steps, a learning rate of 0.0001 was used.

3 Results

For a summary of the results, see Table 2. More detailed results can be found in the images in Appendix A. We observe that the models perform equally well overall. One noteable difference is that RetinaNet seems to perform significantly better on medium sized objects, compared to Faster R-CNN (0.931 vs 0.899).

TODO: COMMENT RESULTS.

Metric	Faster R-CNN	RetinaNet
mAP	0.892	0.897
mAP@50	0.998	0.983
mAP@75	0.973	0.983
mAP battery	0.861	0.873
mAP dice	0.914	0.929
mAP toycar	0.900	0.890
mAP large	0.905	0.903
mAP medium	0.877	0.915
mAP small	-1.000	-1.000
mAR@1	0.672	0.675
mAR@10	0.913	0.922
mAR@100	0.913	0.922
mAR@100 battery	0.887	0.904
mAR@100 dice	0.932	0.945
mAR@100 toycar	0.920	0.918
mAR large	0.922	0.924
mAR medium	0.899	0.931
mAR small	-1.000	-1.000

Table 2: Evaluation results on the test set. For the mean average recall see [3].

4 Questions

4.1 Question 1

> Is the mean average precision a good metric for early stopping during RPN training?

It is indeed. We calculate the mAP as per usual with the number of classes being equal to 1 (the "object" class) and on intersection over union (IoU) thresholds, e.g. [0.5 0.55 .. 0.95]. More explicitly we do the following

---

for each threshold in IoU thresholds **do**
    Sort the proposals descendingly by their objectness score
    Initialize a list containing the ground truths
    Initialize a precision-recall curve
    for each proposed box **do**
        if the proposed box has IoU greater than the threshold with at least one ground truth box **then**
            Consider the proposal as a true positive
            Delete the ground truth box with the highest IoU from the current list of ground truths
        else
            Consider the proposal as a false positive
        end if
        Calculate the current precision and recall, and plot a point on the precision-recall curve
    end for
    Calculate the average precision, that is the area under the precision-recall curve
end for
Calculate the mAP, which is the mean of the average precisions over all IoU thresholds.

4.2 Question 2

> Say an object detection model predicts the classes with high accuracy but the proposed boxes have relatively low IoU with the ground truth ones. How could this issue be ameliorated?

We can give more importance to the box regression loss by changing the loss function from $\text{Loss}_{\text{box regressor}} + \text{Loss}_{\text{classifier}}$ to $a\,\text{Loss}_{\text{box regressor}} + \text{Loss}_{\text{classifier}}$ where $a > 1$.

4.3 Question 3

> Say we have images of a 100m part of a highway, with the images being taken from a UAV in 2840 by 2160 resolution. What issues would be encountered in attempting to detect cars with a model like Faster R-CNN or RetinaNet?

Typical object detector architectures use backbones that work with small images. If we feed a 4K image to a Resnet50 we would end up with a large number of features which would mean a very large number of proposals, which would be hard to handle. A naive approach to solving this issue is to resize the image down to a lower resolution, but this would mean that information about small objects would be lost and it would make it harder or even impossible to detect them. Another approach would be to use an algorithm like Slicing-Aided Hyper Inference (SAHI) [4] where we perform the detection on crops of the original image and then combine the results.

4.4 Question 3

> Say we want to use a model like Faster R-CNN or RetinaNet to perform detection of all people in a dense concert crowd. What issue would be encountered?

The model would probably not perform very well due to the nature of greedy algorithms like Non Max Suppression (NMS) which are typically used to prune "duplicate" proposals from the RPN. The NMS algorithm would repeatedly find the proposal with the highest objectness score and remove all other proposals with IoU higher than some predefined threshold. The issue arises from the fact that in a crowd, the IoU between ground truth boxes is high, which means that the NMS algorithm would suppress "good" proposals. Setting the IoU threshold high would result in many "duplicate" proposals, while setting the IoU threshold low would result in removing many "good" proposals.

References

[1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017. [Online]. Available: https://arxiv.org/abs/1708.02002

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015. [Online]. Available: https://arxiv.org/abs/1506.01497

[3] J. Hosang, R. Benenson, P. Dollar, and B. Schiele, "What makes for effective detection proposals?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 814–830, apr 2016. [Online]. Available: https://doi.org/10.1109%2Ftpami.2015.2465908

[4] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," 2022. [Online]. Available: https://arxiv.org/abs/2202.06934

A Detailed Results

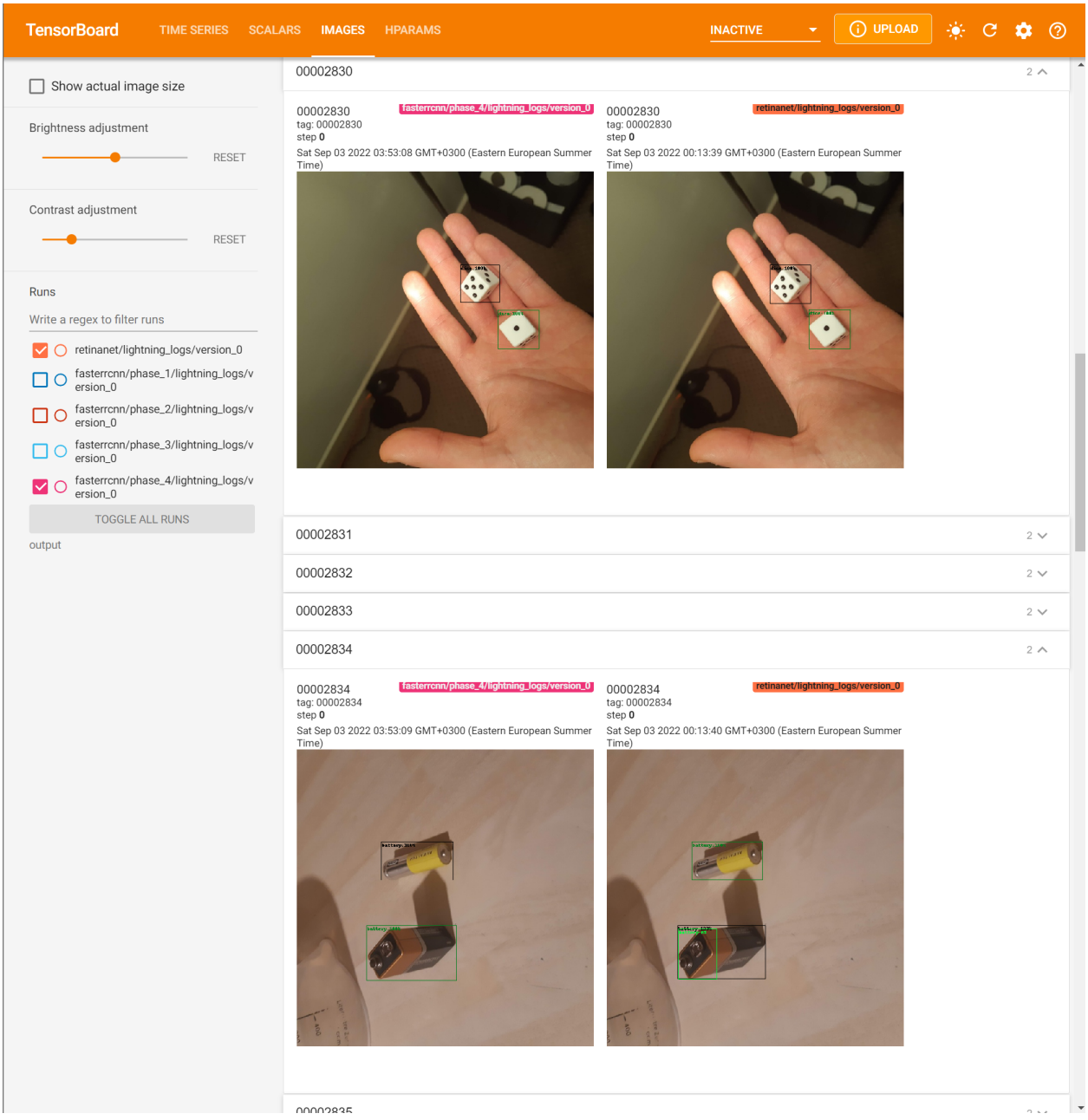


Figure 1: Predictions of Faster R-CNN phase 4, and RetinaNet.

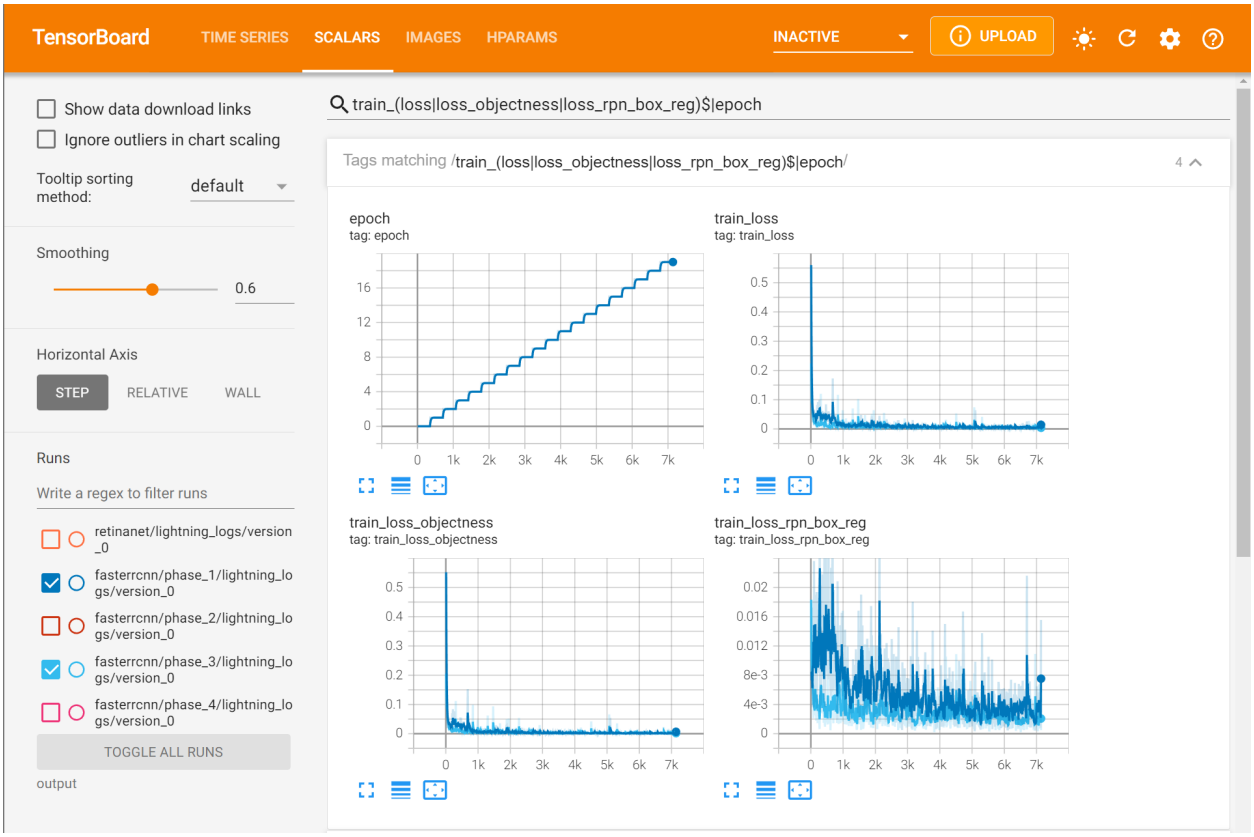


Figure 2: Training loss curves of Faster R-CNN phases 1 and 3.

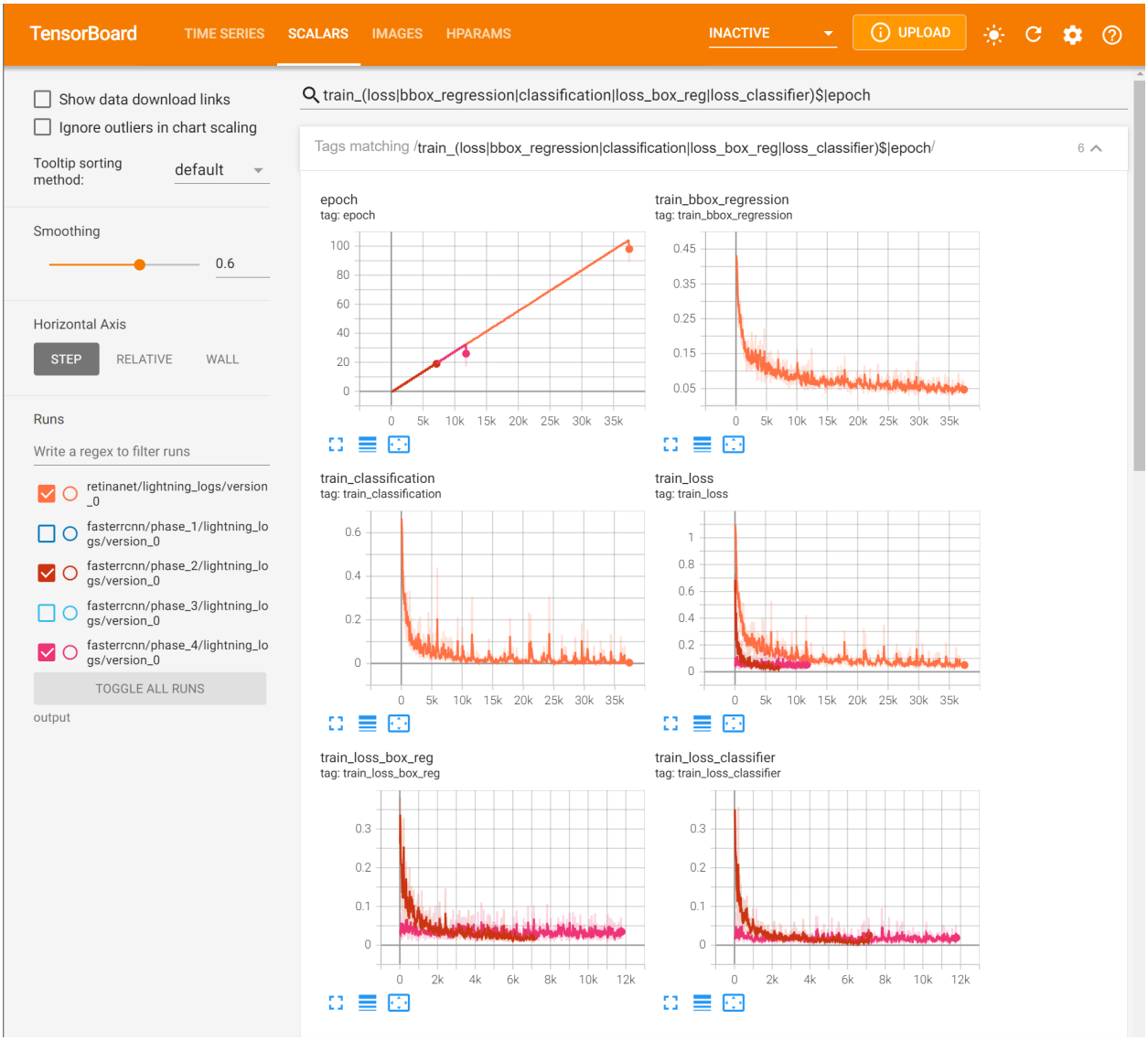


Figure 3: Training loss curves of Faster R-CNN phases 2 and 4, and RetinaNet

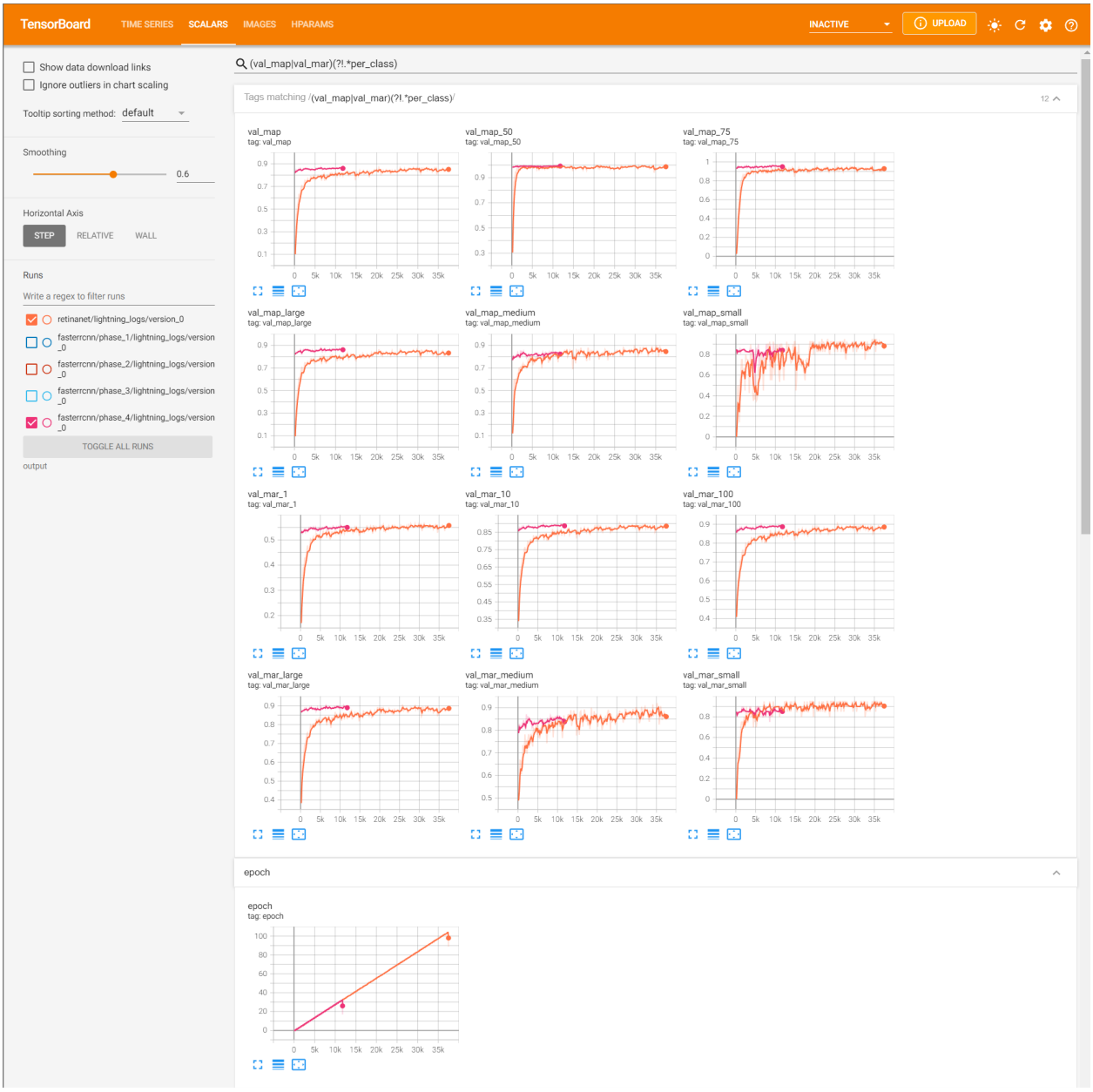


Figure 4: Validation mAP curves of Faster R-CNN phase 4 and RetinaNet

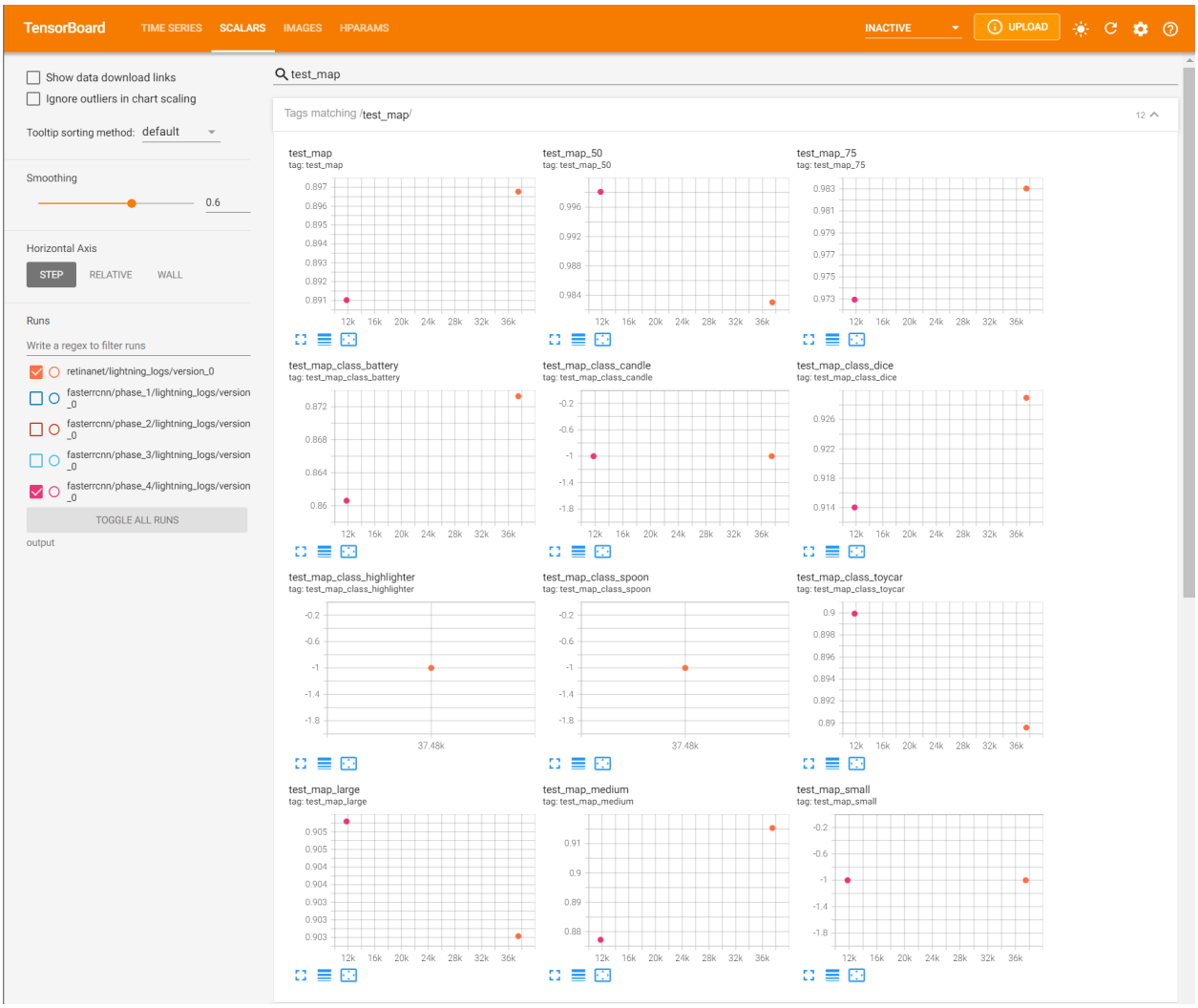
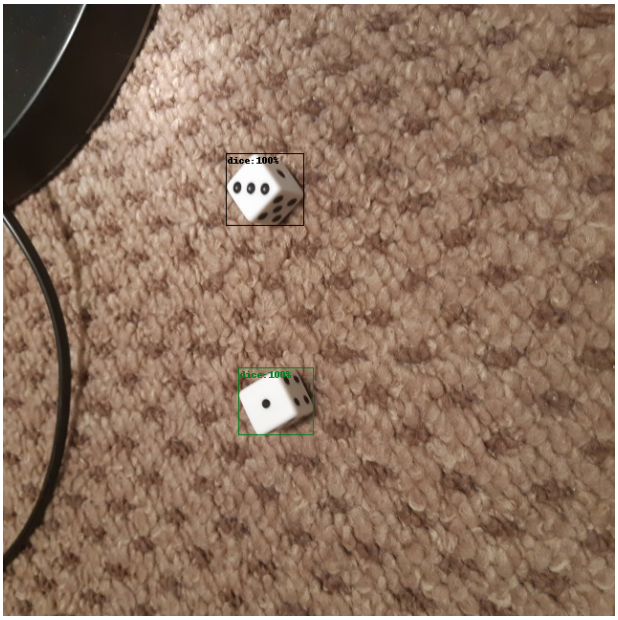
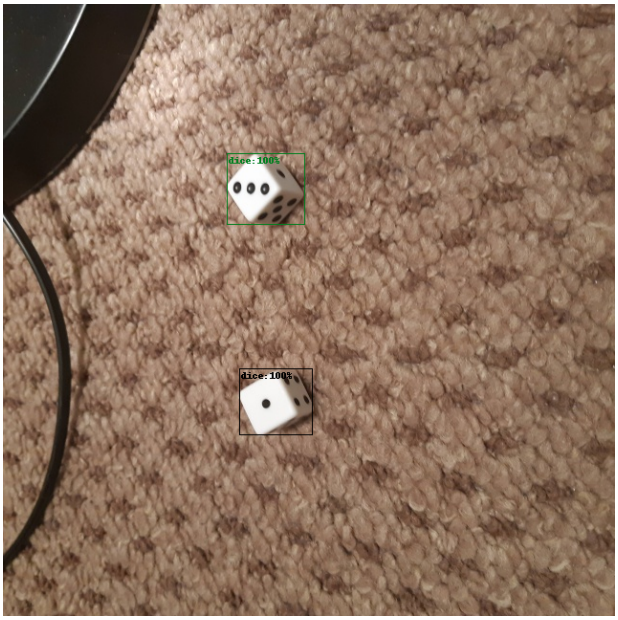


Figure 5: Test mAP of Faster R-CNN phase 4 and RetinaNet

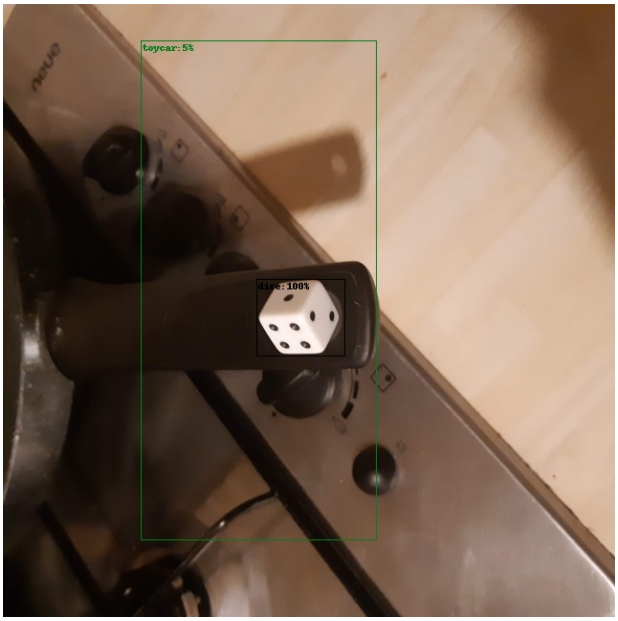


(a) Faster R-CNN

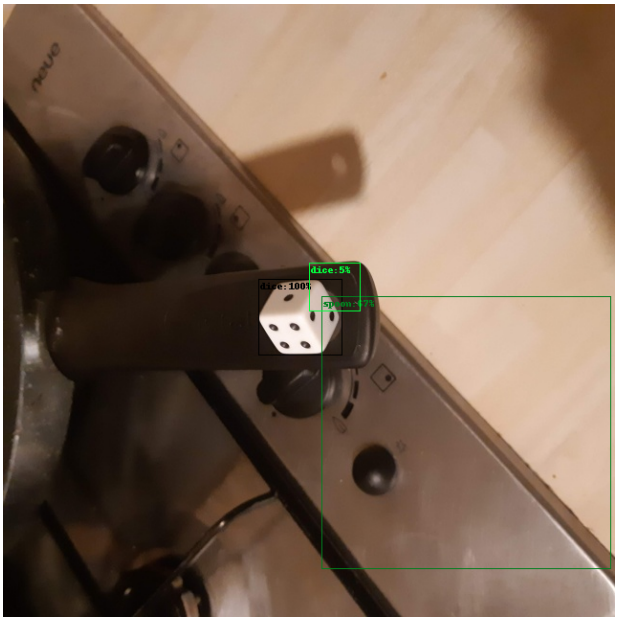


(b) RetinaNet

Figure 6: 00002806.png

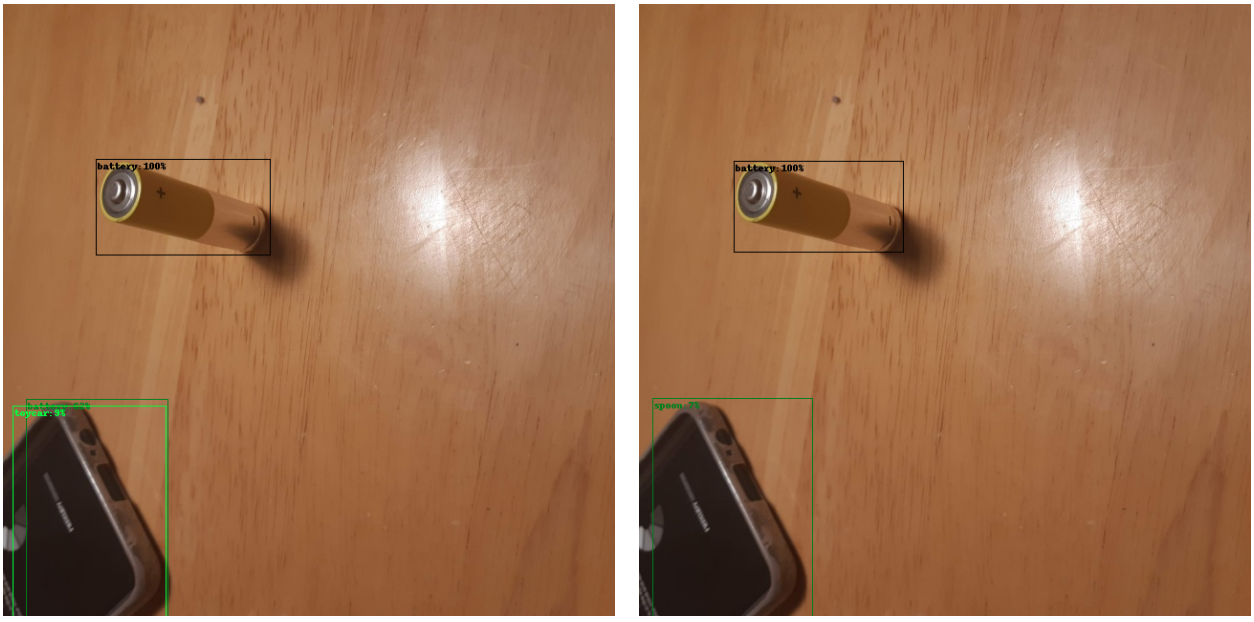


(a) Faster R-CNN



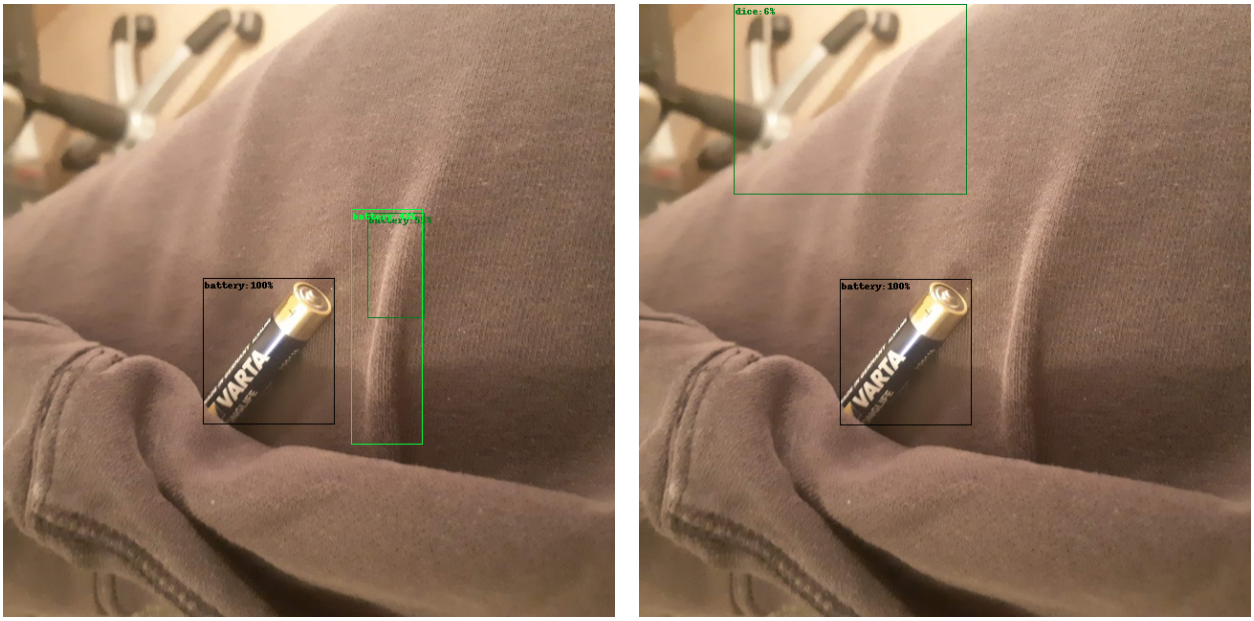
(b) RetinaNet

Figure 7: 00002812.png



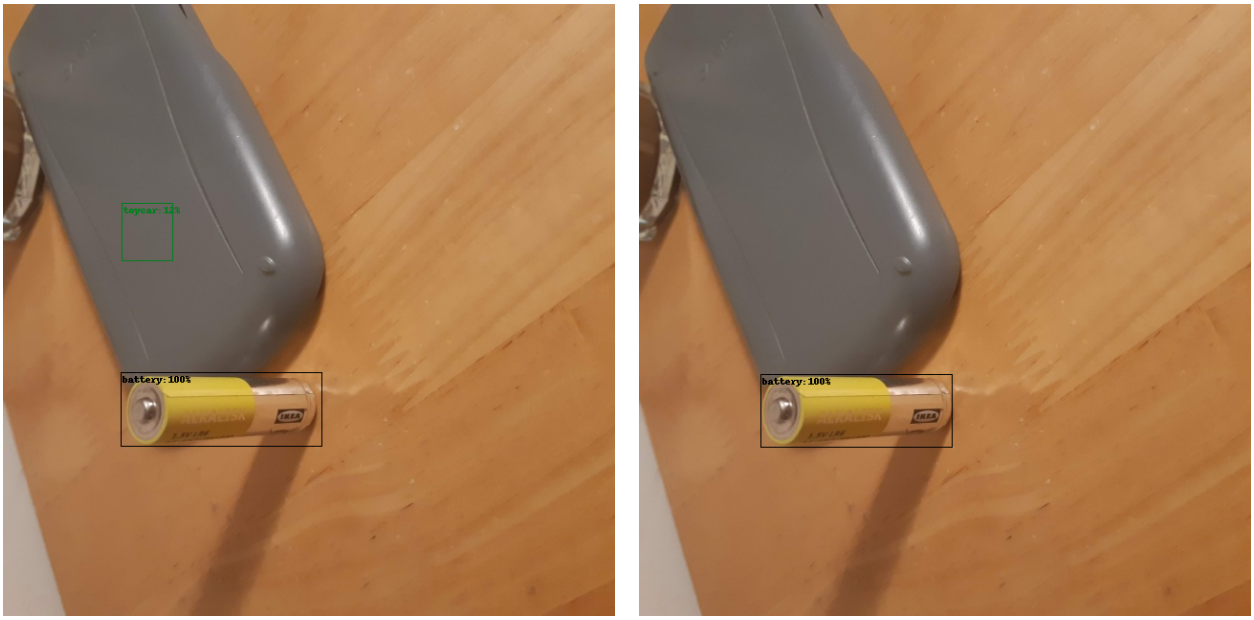
(a) Faster R-CNN (b) RetinaNet

Figure 8: 00002843.png



(a) Faster R-CNN (b) RetinaNet

Figure 9: 00002852.png



(a) Faster R-CNN (b) RetinaNet

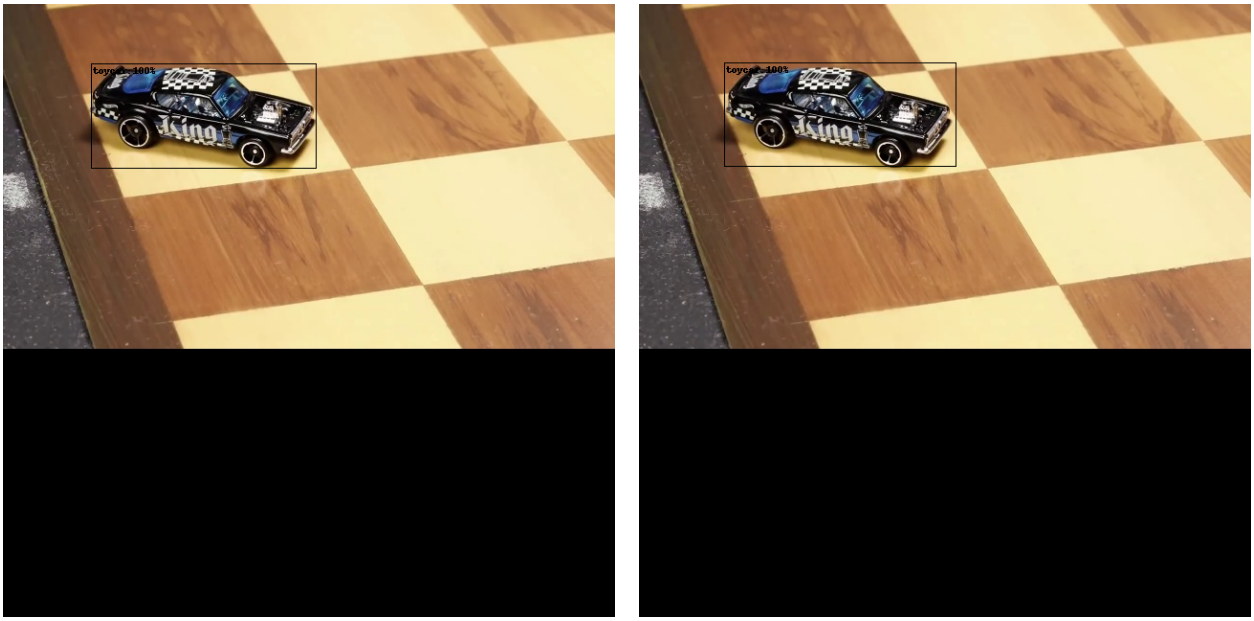
Figure 10: 00002863.png

(a) Faster R-CNN (b) RetinaNet

Figure 11: 00002884.png

(a) Faster R-CNN (b) RetinaNet

Figure 12: 00002894.png



(a) Faster R-CNN (b) RetinaNet

Figure 13: 00002898.png