|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | pretrained | freeze | iters | mAP |
| Basic90 | no | No | 10k | 0.19 |
| Resnet | yes | no | 12.5k | 0.23 |
| ResneXt50 | yes | no | 10k | 0.29 |
| ResneXt50 | yes | Yes (last block trained) | 22.5k | 0.17 |
| ResneXt101 | yes | Yes (last 3 block trained) |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Pretrained | Batch size | Learning rate | Data aug. | Optimizer | mAP |
| Resnext50\_1 | Yes, no freeze | 10 | ReduceLROnPlateau:   * Init lr = 1e-4 * Factor = 0.7 * Patience = 3 | RandomMirror  RandomEffect | AdamW | 0.3067 – step 20k  0.3018 – step 10k |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Augmentation:

* Blur = ±0.4
* Brightness = ±0.4
* Contrast = ±0.2
* Saturation = ±0.3
* Sharpness = ±0.5

Ideer

* Sjekk hvor mye plass de forskjellige classes tar i bildet + hvilke calsses som forekommer oftest
* Konverter alle bilder til grayscale i trenings settet
* Start med større resolution og legg til polling midt i nettverket

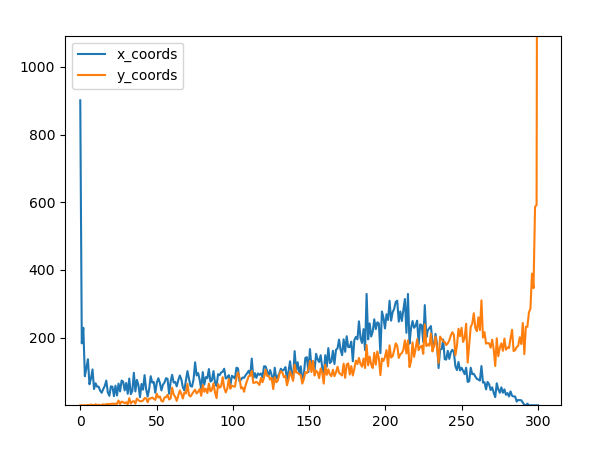
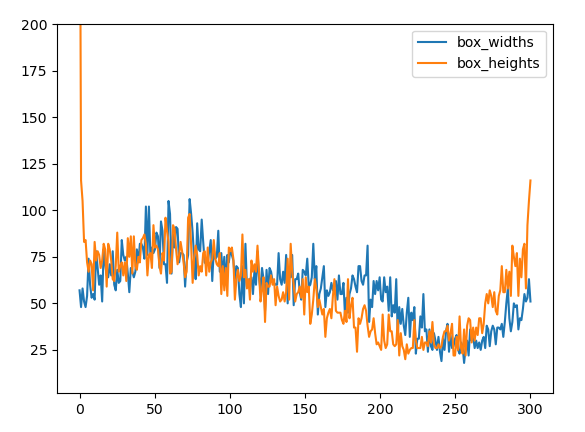
|  |  |
| --- | --- |
| class |  |
| D10 | 4692 |
| D20 | 2544 |
| D30 | 5463 |
| D40 | 4580 |

About dataset:

* Mean = [116.18, 121.74, 121.28]
* Std = [64.97, 67.24, 72.36]



Balanced weights: <https://arxiv.org/ftp/arxiv/papers/2006/2006.01413.pdf>

Box position distribution: (0 = up-left)

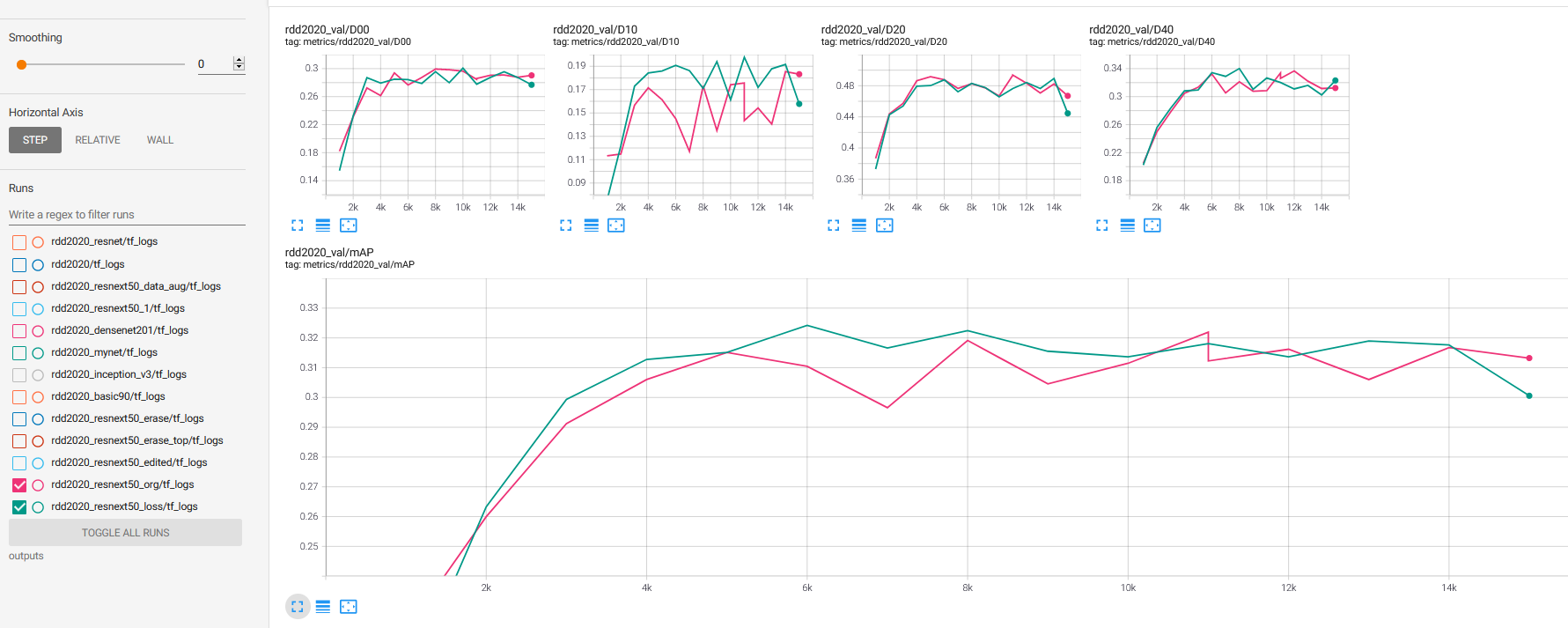
|  |  |  |
| --- | --- | --- |
| Output size | # block |  |
| 150x150x64 | 0 | Conv2d |
|  | 1 | BatchNorm2d |
|  | 2 | ReLU |
| 75x75x64 | 3 | MaxPool2d |
| 75x75x256 | 4 | Bottleneck0  Bottleneck1  Bottleneck2 |
| 38x38x512 | 5 | Bottleneck0  Bottleneck1  Bottleneck2  Bottleneck3 |
| 19x19x1024 | 6 | Bottleneck0  Bottleneck1  Bottleneck2  Bottleneck3  Bottleneck4  Bottleneck5 |
| 10x10x2048 | 7 | Bottleneck0  Bottleneck1  Bottleneck2 |

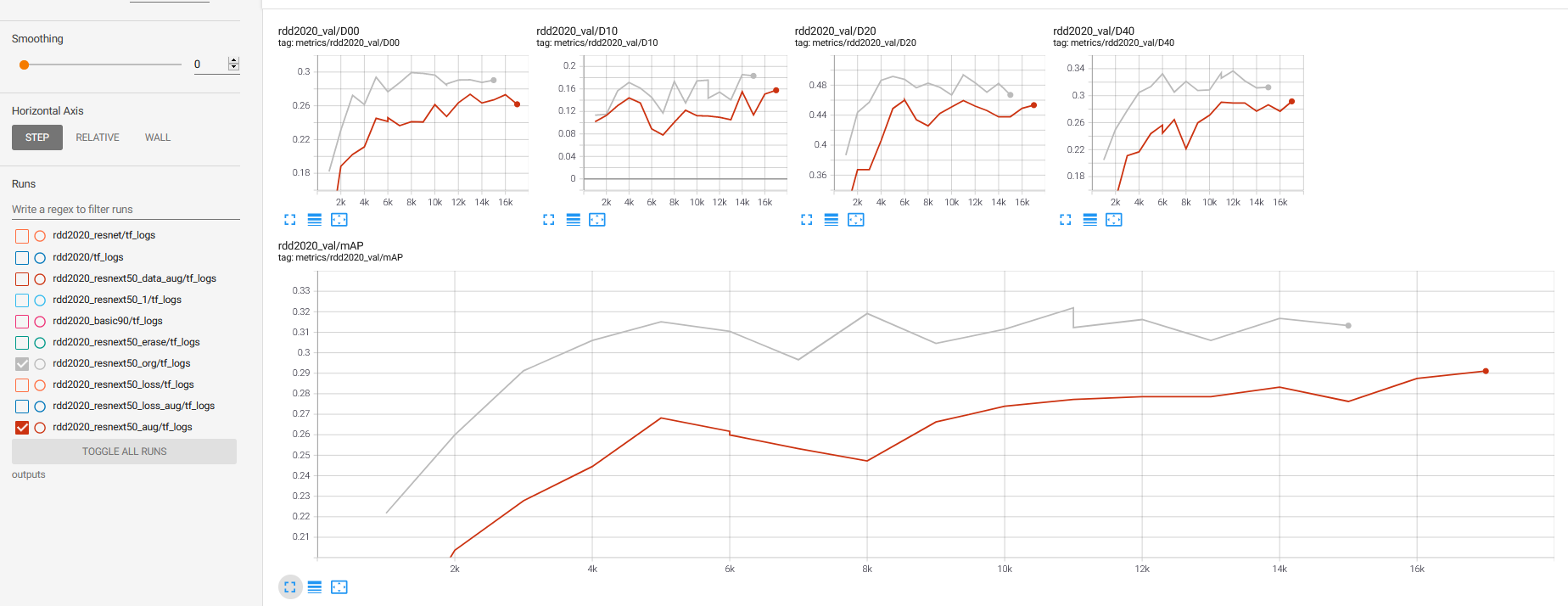
Priori boxes:

Min: [[0.1X, 0.1Y], [0.2X, 0.2Y], [0.37X, 0.37Y], [0.54X, 0.54Y], [0.71X, 0.71Y], [0.88X, 0.88Y]]

Max: [[0.2X, 0.2Y], [0.37X, 0.37Y], [0.54X, 0.54Y], [0.71X, 0.71Y], [0.88X, 0.88Y], [1.05X,1.05Y]]

Stride = [[X/x\_feature Y/y\_feature] … ]





Result on 500x500: mAP = 0.37

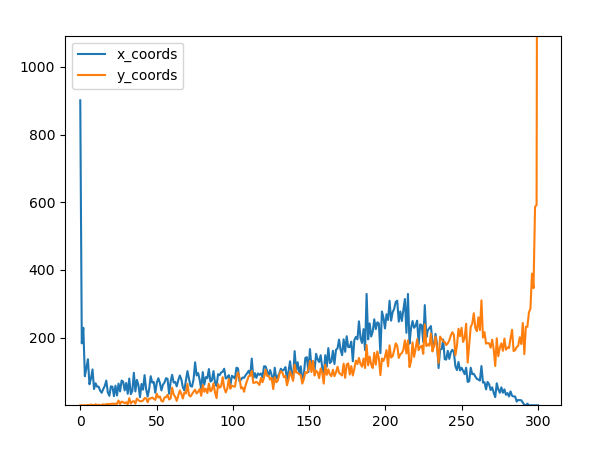
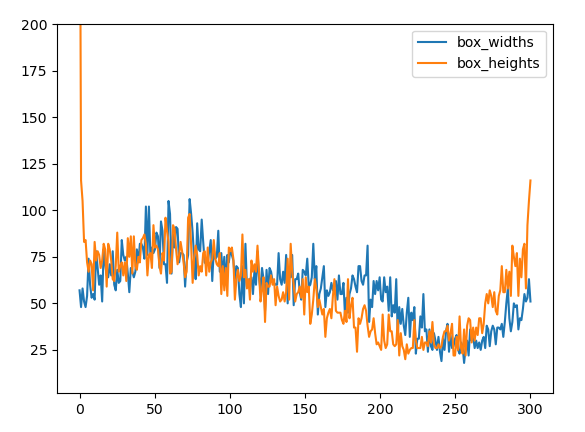
Result on 300x300 using same checkpoint = 0.25

# Dataset

|  |  |
| --- | --- |
| Damage type | Label |
| Longitudinal crack | D00 |
| Lateral crack | D10 |
| Complex crack | D20 |
| Pothole | D40 |

Understanding dataset is an important part of developing a successful deep learning network. Therefore in this chapter we will look at some characteristics of the RDD2020 and tdt4265 dataset, that are used for training road damage detection network. As mentioned in the introduction the is to find and recognize different road damage types. We are using four different labels to classify the damage type, that can be seen in the table below. As a class we had a task to annotate road damage for Norwegian roads. This gave me quite good understanding of challenges when it comes to finding and classifying cracks and holes. Long shadows and wet spots on the road and snow laying o the side of the road can make it often difficult to see the cracks. Other factors that might make the process challenging are reflections in the windshield and video quality. The reason for that is that the cracks in the road are often very thin and small disturbance make the mall object harder to recognize. I would say that the property of the road crack, it’s thickness and length is a trait that will make the object detection challenging. In addition, classifying the road damage itself can be sometime challenging. Longitudinal crack has some common features with a complex crack and a complex crack might resemble a pothole. Road damage detection is generally much easier with high resolution images, where all details are easily visible.

Since all images are taken from a car the environment of each image is bit similar. Some interesting observations are that upper of the image contains almost always buildings and sky. It means that the objects we want to detect will almost never appear in this part of the image. In the graph below we can see distribution of coordinates of bounding boxes in rdd2020 dataset. We can clearly see that most objects lay in the lower left part of the images. Big part of the images comes from india and japan where there there is left-hand traffic, which may explain why majority of boxes lay on the left side.

Therefore cropping the image such that the part that that will anyway don’t contain any objects, might be a valid strategy. This we will not waste computation on searching for road damage in the sky and could use higher resolution to represent road damage without sacrificing too much computational power.

Other interesting aspect of the datasets is number of different types of road damage. Below we can see how many of different classes we can find in each training dataset. In the RDD2020 dataset there are about half as few D20 labels compared to the others. This will clearly be reflected in final mAP where D20 will most likely get lowest accuracy.



# Backbone model

* Resnext50

# Data augmentation

* Horizontal Flip
* Other effects

# Other improvements

* Optimizer
* Loss weights
* Image size

# Future improvements

* Tiling
* More “nondestructive” augmentation
* Other loss weight starategies
* Editing resnext model to handle bigger images and pretrain on other dataset
* Diliation