Documentation of Details

The report should be short and concise. What we expect you to include in this is anything ”boring” and ***note that we will not read this document except*** *if we are interested in technical details of your model, or we want to re-run/validate your experiments. Note that anything included in the presentation should not be included in the report. What we expect is that the report includes anything required to re-produce your results. Such as:*

- ***Hyperparameters. This can be referred to as ”We used the config file our_amazing_model.yml and all hyperparameters are there”. Nothing else is required. 7***
- ***How to train your model. Assume that we want to re-run all your experiments. Document clearly how we should be able to do this. An example of this could be***
    - ***To setup your environment, install the additional packages ”some-package” (Not required if you used the default environment used in the assignments). Then, you can train the model on cityscapes by running the file ”some train.py”. Furthermore, fine-tune the model on TDT4265 dataset by running ”some train2.py. Finally, run the evaluation script.***
- *Specific details of your model architecture. Examples of this can be the tables with models given in previous assignments.*
- *Any additional results that you did not have place for in the report. However, we do not want any discussion of this result in the report.*

The reason we want such a short report is that we do not have enough staff resources to read through everything. Even though I truly enjoy reading some of your assignments and reports, it would take me way too much time getting through all of your reports!

Report

Hyperparameters

How to train

Architecture					
Is output	Layer name	Layer Type	Kernel size	Number of Filters	Stride
		Conv2d	7x7	64	2
		Bn	-	-	-
		ReLU	-	-	-
		MaxPool2D	3x3	-	2
	Layer1	3x			
		Conv2d	3x3	64	1
		Bn	-	-	-
		ReLU	-	-	-
		Conv2d	3x3	64	1
		Bn	-	-	-
	Layer2				
		Conv2d	3x3	128	2
		Bn	-	-	-
		ReLU	-	-	-
		Conv2d	3x3	128	1
		Bn	-	-	-
		Conv2d	1x1	128	2
		Bn	-	-	-
		3x			
		Conv2d	3x3	128	1
		Bn	-	-	-
		ReLU	-	-	-
		Conv2d	3x3	128	1
		Bn	-	-	-
Yes – Res: 102 x 135					
	Layer3	Conv2d	3x3	256	2
		Bn	-	-	-
		ReLU	-	-	-
		Conv2d	3x3	256	1
		Bn	-	-	-
		Conv2d	1x1	256	2
		Bn	-	-	-
		5x			
		Conv2d	3x3	256	1
		Bn	-	-	-
		ReLU	-	-	-
		Conv2d	3x3	256	1
		Bn	-	-	-
Yes – Res: 51 x 68					
	Layer4	Conv2d	3x3	512	2
		Bn	-	-	-
		ReLU	-	-	-
		Conv2d	3x3	512	1
		Bn	-	-	-
		Conv2d	1x1	512	2
		Bn	-	-	-
		2x			
		Conv2d	3x3	512	1

Yes – Res: 26 x 34		Bn ReLU Conv2d Bn	- - 3x3 -	- - 512 -	- - 1 -
Yes – Res: 14 x 17		Conv2d ReLU	2x4	256	2
Yes – Res: 7 x 9		Conv2d ReLu	3x4	256	1
Yes – Res: 5 x 6		Conv2d ReLU	3x3	256	2
Yes – Res: 3 x 3		Conv2d ReLU	2x2	256	1
Yes – Res: 1 x 1		Conv2d ReLU	3x3	256	2

```
2020-04-25 08:31:52,330 SSD.trainer INFO: Start training ...
Input Shape:torch.Size([16, 3, 810, 1080])
Shape after: conv1torch.Size([16, 64, 405, 540])
Shape after: bn1torch.Size([16, 64, 405, 540])
Shape after: relutorch.Size([16, 64, 405, 540])
Shape after: maxpooltorch.Size([16, 64, 203, 270])
Shape after: layer1torch.Size([16, 64, 203, 270])
Appending, Shape after: layer2torch.Size([16, 128, 102, 135])
Appending, Shape after: layer3torch.Size([16, 256, 51, 68])
Appending, Shape after: layer4torch.Size([16, 512, 26, 34])
Shape after: extra k:0 torch.Size([16, 256, 14, 17])
appending
Shape after: extra k:1 torch.Size([16, 256, 7, 9])
appending
Shape after: extra k:2 torch.Size([16, 256, 5, 6])
appending
Shape after: extra k:3 torch.Size([16, 256, 3, 3])
appending
Shape after: extra k:4 torch.Size([16, 256, 1, 1])
appending
```