TDT4265 Road Damage Detection

Group 17
Anders Thallaug Fagerli

Hyperparameters

Hyperparameters for RDD2020 dataset: `SSD/configs/train_rdd2020.yaml`.

Hyperparameters for TDT4265 dataset: `SSD/configs/train_tdt4265.yaml`.

Running the code

NB: Make sure you're in the `SSD/` directory.

Training on the RDD2020 dataset:

```
python3 train.py configs/train_rdd2020.yaml
```

Training on the TDT4245 dataset:

If you want to use a network pre-trained on the RDD2020 dataset, you must give a path to the `.pth` file that contains its parameters. In the config, set `PRETRAINED_PATH` to this path. If you want to train from scratch, just remove `PRETRAINED_PATH` from the config.

```
python3 train.py configs/train_tdt4265.yaml
```

Network layout

ResNet uses blocks to build up each layer. A block in ResNet18, $\begin{bmatrix} \texttt{filter size, channel} \\ \texttt{filter size, channel} \end{bmatrix}$, is made up of two convolutions with filters of size `filter size` and with output channels `channel`. There is batch normalization after each convolution, and the first convolution has a stride of 2, while the second has a stride of 1. There is a ReLU after the first convolution, and again after the skip connection $F(x) + x$. The network layout is shown in Table 1, where the last 6 layers are used as feature maps for SSD. In layer `Extra4`, the first convolution has a stride of 3, so that the size of the last feature map is 1×1.

Model development requirements

The fulfillment of the requirements listed in the project description should be clear from the code and config files.

- The models use image sizes greater than 300×300, as seen from both config files
- The model for RDD2020 is a pre-trained ResNet, as seen from `resnet_rdd.py` in the `backbone`.
- Only random horizontal mirror was used in the final models, but random sample crop was also tested. This can be seen from `__init__.py` in `transforms`, also shown in Figure 1.

Layer name	
Conv1	7×7 filter, 64 channels, stride 2 3×3 maxpool, stride 2
Conv2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
Conv3	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
Conv4	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
Conv5	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Extra1	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Extra2	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Extra3	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Extra4	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$

Table 1: *Network layout for both RDD2020 and TDT4265 datasets.*

```
def build_transforms(cfg, is_train=True):
    if is_train:
        transform = [
            ConvertFromInts(),
            RandomMirror(),
            #RandomSampleCrop(),
            ToPercentCoords(),
            Resize(cfg.INPUT.IMAGE_SIZE),
            SubtractMeans(cfg.INPUT.PIXEL_MEAN, cfg.INPUT.PIXEL_STD),
            ToTensor(),
        ]
```

Figure 1: *Transforms used in `__init__.py`.*