Cityscapes Semantic Segmentation challenge

T.S.W. Stevens N.A.A. Nijssen
Final Assignment - 5LSM0 - Convolutional Neural Networks for Computer Vision

I. Introduction

In the world of computer vision there are some existing challenges for classification and segmentation tasks. One can participate in such a challenge by using the dataset to develop a method with good performance on the task. The Cityscapes challenge [1] is one of those challenges. It is focused on semantic understanding of urban street scenes. The Cityscapes dataset consists of images of streets from different cities across Germany. Three different tasks are available for this challenge: pixel-level, instance-level and panoptic semantic labeling tasks.

The focus of the assignment is on the pixel-level semantic labeling task. This is a semantic segmentation problem, where the goal is to assign a class to each pixel in an image. So basically, classification on a pixel-level. For this, the output image size should be the same as the input image size. However, if there is no downsampling, the computations of the convolutions become very expensive. Therefore, semantic segmentation problems are generally tackled by networks with a downsampling and an upsampling part (encoder and decoder). State-of-the-art architectures for segmentation tasks are for example fully convolutional networks, U-Net and DeepLab [2].

In this assignment, a method for the pixel-level semantic labeling task of the Cityscapes challenge is developed. This is done by starting from a baseline implementation of U-Net, and making improvements on that baseline to get better performance on the challenge. The results of the baseline and experiments on improvement are shown. Lastly, some limitations on the method and options to improve further are discussed. We provide our PyTorch implementation and models for this segmentation task on GitHub.[1]

II. Baseline implementation

A simple U-net PyTorch implementation [3] has been chosen as basis framework during this research. The U-net architecture has been introduced as a solution for biomedical image segmentation [4]. U-net utilizes convolutional operations in an encoder-decoder fashion which is common in segmentation schemes. The main distinct feature of U-net is that is makes use of skip connections that concatenate the high-resolution feature maps of the encoder or contracting path to the decoder or expansive path. This allows the network to learn global image context as well as preserving spatial accuracy. Especially this last feature allows U-net to predict highly detailed semantic segmentations which is not possible with conventional fully convolutional networks that lack these skip connections.

[1]Source code https://github.com/tristan-deep/5LSM0_CityscapesDataset

TABLE I
Architecture settings for the baseline

Setting	Value
Input image size	1024×2048
Number of input channels	3
Number of output channels	34
Number of filters in the first layer	8
Depth	4
Batch normalization	`False`
Upsampling	`bilinear`

Each block in the encoder path of this particular U-net architecture consists of two 2D convolutional layers each followed by a ReLU activation. The output of this block is subsequently downsampled by a factor of two using maxpooling. Roughly speaking, the inverse is done in the decoder path.

Finally, a pixel-wise soft-max over the final feature map is taken and fed into a cross-entropy loss function defined by

$$\mathcal{L}_{CE} = -\sum_{i}^{C} y_i \log(\hat{y}), \tag{1}$$

where y is the ground truth and $\hat{y}$ the prediction.

A. U-Net network architecture

U-Net consists of a downstream and an upstream path, where the downstream convolution maps are appended to the upstream convolution maps of the same size. The downstream path (encoder) consists of convolution blocks with maxpooling layers in-between. The maxpooling layer halves the size of the feature map and the number of channels doubles per block. A convolution block has the following structure:

```
conv_block = (conv_layer > batch_norm > ReLU) x 2
```

The batch normalization layer can be enabled or disabled. The depth of the network can be chosen, where the depth is equal to the amount of downstream blocks:

```
down_block = convolution_block > max_pooling
```

A block in the upstream path (decoder) consists of some form of upsampling, followed by concatenation with the downstream convolution maps and a convolution block:

```
up_block = upsampling > concatenate > conv_block
```

Upsampling doubles the size of the feature maps and the number of channels halves per block. Upsampling can be in the form of bilinear upsampling combined with a convolution layer, or transposed convolutions (upconvolutions). The last layer of the network is a convolution layer with the number of output channels equal to the number of classes to predict.

The architecture settings for the U-Net network of the baseline are given in Table I. The hyperparameters for the baselin can be found in Table II.

TABLE II
HYPERPARAMETERS OF THE BASELINE

Parameter	Value
Number of epochs	3
Batch size	4
Optimizer	`Adam`
betas	$(0.9, 0.999)$
eps	$1e-8$
weight decay	$5e-4$
Learning rate	`StepLR`
init, step size, gamma	$1e-2$, 1, 0.1

TABLE III
HYPERPARAMETERS OF THE IMPROVED MODEL

Parameter	Value
Number of epochs	10
Batch size	3
Optimizer	`Adam`
betas	$(0.9, 0.999)$
eps	$1e-8$
weight decay	$5e-4$
Learning rate	`MultiStepLR`
init, gamma	$1e-2$, 0.1
milestones:	1, 4, 8 epochs

III. IMPROVED MODEL

Several adjustments can be made to the baseline model. We will discuss these changes that proved to increase performance and explain why they work. The used hyperparameters for the final model are given in Table III.

A. Weighted cross-entropy

As the Cityscapes dataset contains highly unbalanced distribution of the classes, the loss function is dominated by the outnumbering classes which results in poor predictions. The trained network simply does not "care" about classes such as humans or smaller objects in comparison with the road and therefore does not predict these classes. One way to deal with an unbalanced dataset is with the use of a weighted cross-entropy loss:

$$\mathcal{L}_{WCE} = -\sum_{i}^{C} w_i \left(y_i \log(\hat{y})\right), \tag{2}$$

where the weights w_i are determined by there relative frequency as follows:

$$f_i = \log\left(\mu \frac{\sum p}{\sum p_i}\right), \tag{3}$$

$$w_i = \begin{cases} f_i & f_i \geq 1 \\ 1 & \text{else.} \end{cases} \tag{4}$$

Here $\mu = 0.15$ is a tunable smoothing parameter that determines how sever the weights are related to the size of the classes. The logarithm is used to prevent weighing the loss function directly with the relative frequencies, which can differ in many orders of magnitude.

TABLE IV
ARCHITECTURE SETTINGS FOR THE IMPROVED MODEL THAT ARE DIFFERENT FROM THE BASELINE SETTINGS

Setting	Value
Depth	5
Batch normalization	`True`
Upsampling	`transposed convolutions`

B. Learning rate decay

Learning rate decay over time is used for better convergence and to avoid oscillations around the minimum. It takes smaller steps towards to minimum as it gets closer to this minimum. In the improved model, we use a multi step-wise decay, where the learning rate is decreased by a factor γ each time a defined number of epochs (milestones) is reached .

C. Improved network architecture

In the original U-Net the batch normalization layer was applied after the activation layer (ReLU). Batch normalization is used to achieve activations with a stable distribution, which is why it was originally added before the nonlinearity [5]. Therefore, we choose to apply the batch normalizations before the ReLu activations. Furthermore, transposed convolutions are used instead of bilinear upsampling, as this allows for the network to learn the upsampling by itself. Lastly, the depth of the network is increase to 5, which introduces higher level features. The trade-off for this decision is a lower batch size since increasing the depth requires more memory. The final network architecture settings of the improved model are given in Table IV.

IV. RESULTS

A. Training

The Cityscapes dataset consists of images of urban street scenes from 50 different cities. For more information on the dataset see the Cityscapes challenge website [1]. The dataset has 5000 images with fine annotations and 20000 images with coarse annotations. We only used the fine annotations. We therefore had 2975 training, 500 validation and 1525 testing images.

The images are labelled with 34 different classes, which can be subdivided into 8 categories. However, the performance of the submissions to the Cityscapes challenge are evaluated on only 19 classes, spread over 7 categories. We train our networks on all 34 classes, so with 34 output channels.

B. Evaluation

To evaluate the performance of our models on the validation set, the evaluation script for pixel-level semantic segmentation from Cityscapes [6] is adapted. The performance is assessed by the intersection-over-union (IoU) metric, or Jaccard index. We used this to get the IoU for the baseline and improved model on the validation set, and the same evaluation is used by Cityscapes on the test set.

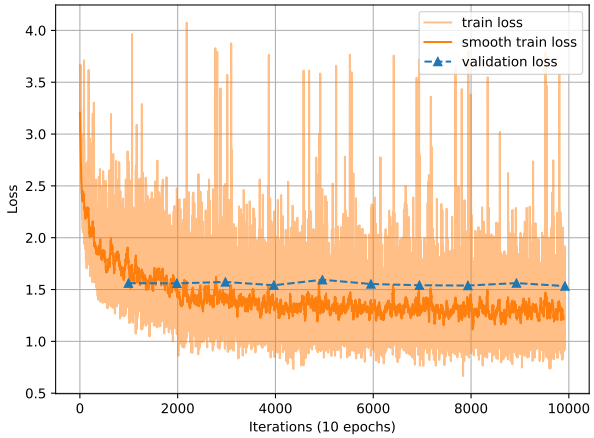

Fig. 1. Train (orange) and validation (blue) loss computed by a weighted cross-entropy (WCE) loss function. Due to the small batch-size, the loss is very noisy w.r.t. each iteration. Therefore a moving averaging filter is used on the train loss to be able to see the trend.

C. Comparison of the models

The results for the category scores for the baseline on the validation set and for the improved model on the validation and test set are given in Table V. The test results are obtained after submission to the Cityscapes challenge [1], as the labels for this part of the dataset are hidden. The improved model vastly outperforms the baseline. As a disclaimer, the baseline model is not optimized in any way and is only here to show what happens if you do not use most of the available deep learning tools. Only priors are a optimizer with suitable learning rate as well as the U-net architecture. As the loss for the baseline model is not weighted, the network does not predict any less dominant classes such as humans correctly (IoU$=0$). Lots of improvements have been made in this regard with the introduction of the weighted loss function (2).

The train and validation loss on the improved model is shown in Fig. 1. The loss function is very noisy, due to the small batch size. The size of the input images is rather large and the network is not able to overfit with this network. This is confirmed by the validation loss which is does not increase during training.

In Fig. 2 some examples of the validation set images with corresponding predictions and ground truth are shown. Larger continuous areas that belong to the same class are mostly identified correctly. Independent of the classifications, most details are preserved which is characteristic for the U-net. Still, humans and most objects are still miss-classified. In the third example, a person on a bike is incorrectly seen as a car.

Lastly, it can be seen that our predictions have many holes of a few pixels. This can be explained by a too small receptive field of the filters. In section V some techniques for improving the current results further are discussed.

V. Discussion

Due to limited hardware resources, we were not able to train a deeper network. A disadvantage of the U-net is that it is very memory expensive. Since lower level features are stored to concatenate later with the upstream path, the amount of parameters grows very quickly when increasing the depth of the network. Increasing the depth of the network will allow for higher level features which will help the network with more precise classification. The holes in the predictions can be explained by the lack of sufficient receptive field of the network. Since the network is not very deep, it might be possible that it mainly learned local features. By downsampling the images more, the receptive field is increases and the network is able to learn high-level features. This could give more smooth predictions. Downsampling can be achieved by making the network deeper or using dilated convolutions in some network layers.

Another possible improvement on the method can be to use a different loss function. The weighted cross-entropy loss function that we used is able to deal with some class imbalance, but measures of overlap are more robust against it [7]. Dice overlap, for example, can be used as a loss function. The dice loss is based on the dice coefficient which is similar to the intersection-over-union method. Since we use IoU as a metric for the performance, it would be a good idea to use a loss function that is closely related to the metric, as this is what we want to maximize. Using the dice loss instead of cross-entropy loss does not guarantees better performance of the method, however, it might be worth a try.

One method that is often used to increase the performance of a model is to use data augmentation. Data augmentation increases the diversity and size of the dataset. This can lead to increase in performance as the network is more robust to changes in the test set.

TABLE V
Benchmark category scores on the Cityscapes data set.

Categories (split, model)	IoU (test, improved)	(val, baseline)	(val, improved)
construction	66.7421	11.3660	63.9651
flat	89.7289	60.6645	89.4850
human	5.89255	0.0000	9.4731
nature	74.5018	44.8942	72.3765
object	3.63665	0.0000	3.6762
sky	78.8369	0.0000	65.7392
vehicle	48.5196	0.0000	42.6824
Score average	**52.5512**	**16.7035**	**49.6282**

VI. Conclusion

Designing a convolutional neural network is not trivial, and even applying the same proven architecture such as U-net to a different use case (Cityscapes) is hard. As the results show, just plugging in an existing network without thinking carefully about the data you are dealing with leads to poor predictions. This paper has shown some insights in the design choices that lead to an at least descent network given the hardware constraints. After applying some key concepts within deep learning, the overal mIoU score increased from 16.7% to 52.6%. Lastly some shortcomings of the current architecture and additional tips that will improve the quality of the predictions further are given.

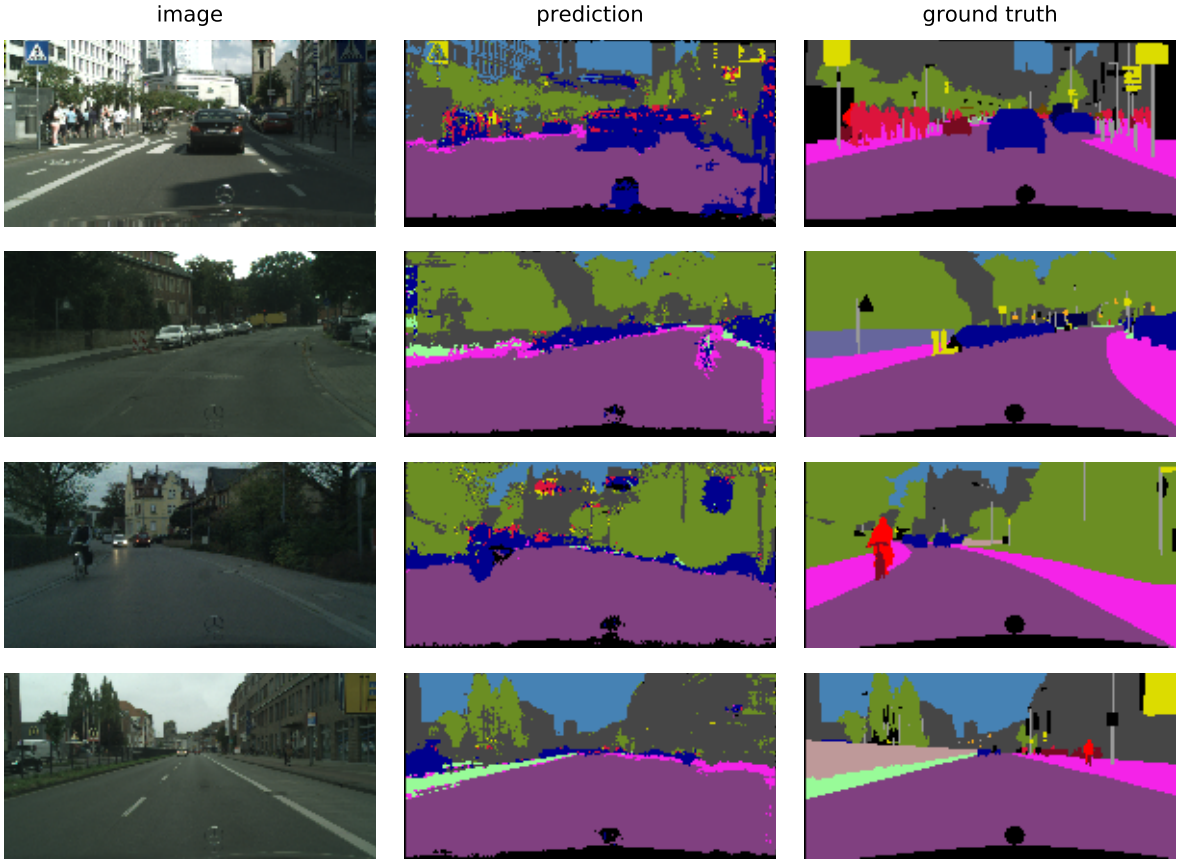

Fig. 2. Multiple examples out of the Cityscapes validation set with corresponding predictions from the improved model and ground truth labels.

REFERENCES

[1] Cityscapes Dataset, "The cityscapes dataset," 2020. [Online]. Available: https://www.cityscapes-dataset.com/

[2] D. Mwiti, "A 2019 guide to semantic segmentation," 2019. [Online]. Available: https://heartbeat.fritz.ai/a-2019-guide-to-semantic-segmentation-ca8242f5a7fc

[3] jvanvught, "Tunable u-net implementation in pytorch," 2019. [Online]. Available: https://github.com/jvanvugt/pytorch-unet

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[6] M. Cordts and M. Omran, "The cityscapes dataset," 2020. [Online]. Available: https://github.com/mcordts/cityscapesScripts

[7] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.