IMPROVING DEEP LEARNING BASED SEMANTIC SEGMENTATION WITH MULTI VIEW OUTLIER CORRECTION

T. Peters[1,*], C. Brenner[1], M. Song[2]

[1] Institute of Cartography and Geoinformatics, Leibniz Universität Hannover, Germany - {peters,brenner}@ikg.uni-hannover.de,
[2] Institute of Cartography and Geoinformatics, Leibniz Universität Hannover, Germany - ming.song.cn@outlook.com

Commission II, WG II/6

KEY WORDS: Deep Learning, Transfer Learning, MMS, Point Cloud, Multi-view

ABSTRACT:

The goal of this paper is to use transfer learning for semi supervised semantic segmentation in 2D images: given a pretrained deep convolutional network (DCNN), our aim is to adapt it to a new camera-sensor system by enforcing predictions to be consistent for the same object in space. This is enabled by projecting 3D object points into multi-view 2D images. Since every 3D object point is usually mapped to a number of 2D images, each of which undergoes a pixelwise classification using the pretrained DCNN, we obtain a number of predictions (labels) for the same object point. This makes it possible to detect and correct outlier predictions. Ultimately, we retrain the DCNN on the corrected dataset in order to adapt the network to the new input data. We demonstrate the effectiveness of our approach on a mobile mapping dataset containing over 10'000 images and more than 1 billion 3D points. Moreover, we manually annotated a subset of the mobile mapping images and show that we were able to rise the mean intersection over union (mIoU) by approximately 10% with Deeplabv3+, using our approach.

1. INTRODUCTION

The problem of overfitting in deep neural networks is the norm rather than the exception when they are trained on small datasets. Even with large annotated datasets, such networks often do not generalize well without loss to unseen data. Therefore, much effort is put into reusing knowledge or adapting pretrained networks to new problems in order to avoid high costs for labeling data and increase the performance of the models. In computer vision, this is often accomplished by training a DCNN on a publicly available dataset and fine-tuning it to the target dataset to solve a similar task. It is desirable to use only few or even no annotations in the target dataset. In those semi- or unsupervised cases one often has to make assumptions about the nature of the data in order to solve these problems.

The approach of this work is to use a pre-trained DCNN for semantic segmentation in 2D images and apply this network to a new dataset with different camera sensors and different viewing angles in a new environment. The reduced performance of the DCNN on this new dataset is sometimes referred to as *domain gap*. The question we ask in this paper is: **Is it possible to correct false predictions, from multi-view images, and retrain the DCNN in order to close the domain gap?** To answer this question, we collected an image and point cloud dataset with a mobile mapping system (MMS), which is equipped with two laserscanners, two cameras, and an GNSS/ IMU system, all fully calibrated. We used the publicly available pre-trained networks Deeplabv3+ by (Chen et al., 2018b) and HRNetV2 by (Yuan et al., 2019) in order to semantically segment all images recorded by the MMS. Both networks were trained on the Cityscapes dataset (Cordts et al., 2016). Finally, we projected the 3D point cloud to all involved 2D images. This leads to different predictions for every single 3D point due to multi-view observations, which needs to be resolved. In order to measure the impact of our process we manually annotated a subset of the mobile mapping dataset, as shown in figure 1.

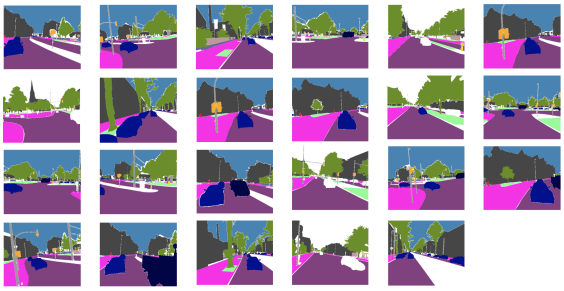

Figure 1. Manually annotated images for a subset of the MMS dataset, according to the official Cityscapes policy.

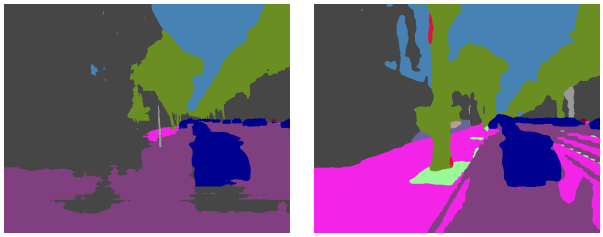

Figure 2. Predictions before (left) and after correction (right). We recovered the sidewalk (pink), terrain (light green), vegetation (green) and improved the overall mIoU by 10%.

Our key contributions in this work are:

- Creation of a MMS data set containing $\sim$ 10'000 partly annotated multi-view images and aligned 3D point clouds.
- Introduction of a neural network that resolves outliers of multi-view object points in any order of input and with variable observation length.
- Retraining of a DCNN in the new domain with the corrected data set, resulting in a $\sim$ 10% increase in mIoU.

* Corresponding author

2. RELATED WORK

Semantic segmentation attempts to segment and classify parts of a scene by doing pixel- or pointwise classification. Since the rise of deep learning, popular approaches are using fully convolutional neural networks for the semantic segmentation of images (Long et al., 2015). More advanced approaches still rely on neural networks (Badrinarayanan et al., 2017, Zhao et al., 2017, Chen et al., 2018a), but improved performance by modifying their architecture and components. The Network performance is measured using publicly available annotated data sets such as Cityscapes (Cordts et al., 2016) or the PASCAL VOC challenge (Everingham et al., 2010).

Semi-supervised learning is a task that lies between a supervised (completely annotated dataset) and unsupervised setting (no annotations). The aim is almost always to reduce expensive labelling costs. In most cases, a learner is trained by using a large number of unlabelled instances in combination with a limited number of labelled instances (Van Engelen, Hoos, 2020). Some popular key assumptions for semi-supervised learning are the smoothness and manifold assumption, which states that two closely located samples in the input space or on the same low-dimensional manifold should have the same label (Van Engelen, Hoos, 2020).

Transfer learning for semantic segmentation aims to improve the classifier in the target domain by transferring knowledge from a source domain. Like semi-supervised learning, it is often used in order to reduce labeling costs. Many current approaches in deep learning attempt to achieve this by training a discriminator that learns to distinguish between the extracted features of the source and target domains. The classifier, on the other hand, learns to deceive the discriminator, thus closing the domain gap (Tzeng et al., 2017, Hoffman et al., 2017, Liu et al., 2019).

Some approaches to domain adaptation or general improvement of predictions with **multi view consistency** have been presented in the past. For example, (Zhou et al., 2018) improves 3D keypoint predictions in an unsupervised manner by forcing them to be consistent in space. In the work of (Floros, Leibe, 2012), a conditional random field (CRF) is used for semantic segmentation by enforcing temporal consistency between video images. (Hermans et al., 2014) proposed a method for semantic segmentation in RGB-D images. This involves projecting the individual images to 3D and smoothing them in the point cloud using a CRF. (Ma et al., 2017) presented an approach for consistent depth learning with multiple views. They warped feature maps of RGB-D images from multiple views into a common reference frame to make the predictions consistent at different viewing angles.

Lastly, **transferring labels** between 3D and 2D space has been done before, e.g., (Xie et al., 2016) transferred human annotated point clouds into images in order to create arbitrary amounts of training images and corresponding labels. On the other hand, (Zhang et al., 2018) and (Peters, Brenner, 2019) used semantic segmented images and projected them to 3D point clouds in order to create annotated point clouds.

3. PROBLEM STATEMENT

In order to eliminate wrong predictions of the pre-trained DCNN, we are introducing The following strategy. As depicted in figure 3, box 2.), the general problem is that any 3D point X is possibly mapped to a number of images, and in each of these images, the DCNN will predict a class distribution. Then, given all these class distributions, the task is to predict the correct class label Y for the 3D point X. More precisely, if a point X is mapped to n images, the pixel coordinates x_i, $1 \leq i \leq n$ in each image are known, so that the corresponding predicted class distributions h_i can simply be looked up. Then, the task is to predict the correct class label Y from all class distributions $\{h_1, \ldots, h_n\}$. Depending on the number of outliers, this can be approached in different ways. In cases where only a few false predictions occur, this can be solved by a simple majority vote (unsupervised). The major label can then be propagated back to all images. Other cases, which suffer from strong noise, can be tackled by using prior information gathered by a small annotated subset. We can, for example, take into account image regions which regularly suffer from false predictions. Based on the region, we can calculate the weighted sum over all predictions. We show that we are able to correct outliers based on 3D point features and the list of class predicitons by training a neural network on the annotated subset. All those cases are semi-supervised, since we are only using a small subset in order to correct our large database of over 10'000 images. Finally, the corrected predictions are then used as coarse annotations in order to retrain the DCNN, as shown in figure 3, box 5.).

4. CORRECTION STRATEGY

In contrast to most classification problems that require a fixed length input or some sort of ordered data, the list of 2D predictions and features assigned to a single 3D point can be of arbitrarily length and order. Therefore there is no straightforward solution for the problem of predicting the correct class label Y from all class distributions $\{h_1, \ldots, h_n\}$.

In order to introduce our solution we would first specify the data and input features we calculated, figure 3, box 2.), and later explain how we solved the classification problem, figure 3, box 3.).

A 3D point X_j will be projected to the pixel coordinates $x_{i,j}$, $1 \leq i \leq n_j$, in case of n_j multi-view images, in which the point is visible. That means that the number of multi-view images n_j differs for every 3D point X_j. Therefore, if we group all 2D pixel coordinates $x_{i,j}$ we receive a list z_j of length n_j. Additionally, since the order of the 2D pixel coordinates does not necessarily have a meaning related to the classification problem, we assume that the list is in random order.

In addition to the class predictions, we added some features in order to increase the correction performance. We mixed 2D and 3D point features by accumulating them along the laser ray through the 2D images. As there is only one 3D point related to n_j 2D points, we concatenated the following 3D features to every list entry:

- The normalized reflectance of the laser ray r_j.
- The estimated point normals $\vec{n}_j$ of the 3D point.
- A value between 1 and 14, which is the *campaign count*, denoted as d_j. It gives a measure of how 'dynamic' a 3D point is by counting how often we measured a specific voxel in distinct mapping campaigns (14 in total). If the point belongs to a voxel that was measured only once, we can assume it relates to a very dynamic or occluded object. Vice versa, a point belonging to a voxel which we captured

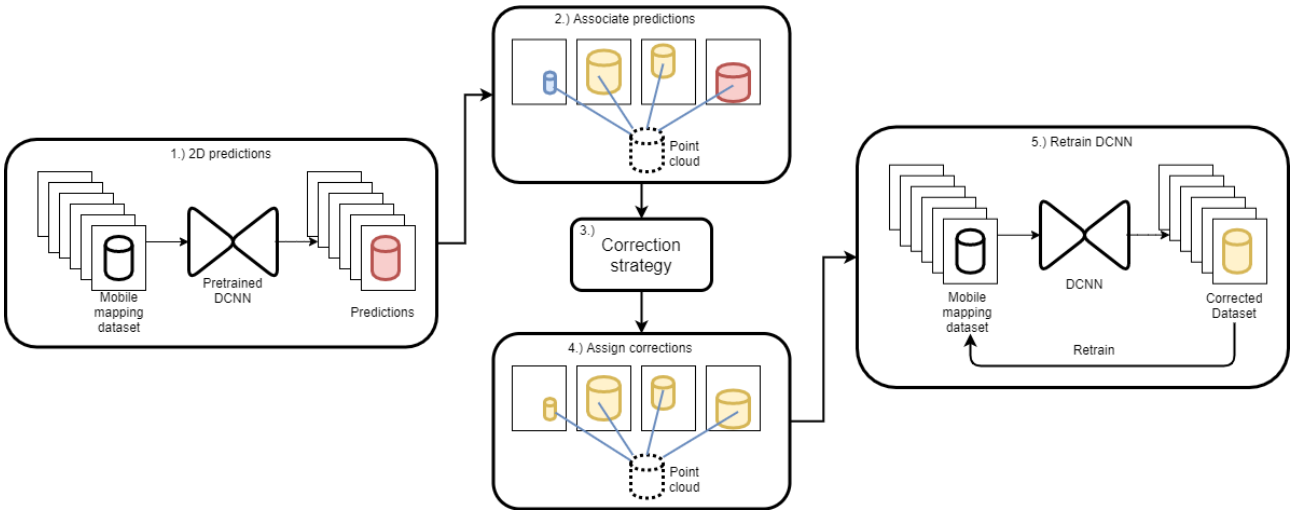

Figure 3. Methodology for retraining the DCNN on a new dataset. Colors are indicating different class predictions. The cylinder shows the same object observed in multiple views.

in every mapping campaign belongs to a static object with a high probability.

Moreover, we added a number of 2D point features to every list entry:

- The predicted class distribution h by the DCNN.
- The RGB value RGB at the image coordinate, as well as the RGB values of the adjacent pixels, 27 values in total.
- The distance δ between the camera center and the 3D point.
- Finally, we calculated the distance β between the image center and the 2D pixel coordinate, which allows to capture cases where a specific field of view in the image is responsible for misclassifications.

Overall, every list entry in $z_{i,j}$ is a vector that contains the following values: $(r_j, \vec{n}_j, d_j, h_{i,j}, RGB_{i,j}, \delta_{i,j}, \beta_{i,j})$. In total the feature vector has a length of 53.

To assign a ground truth label Y_j for the respective input list z_j, we sum up all manually labeled classes along the associated 2D pixel coordinates and take the class with the highest occurrence count as ground truth label.

Please note that we are not able to use all manually annotated labels, since not every 2D pixel belongs to a 3D point. This is due to the sparsity of the 3D point cloud, which does not cover the entire image. Therefore, we used the ground truth labels that are not associated with a 3D point as a validation and test set for the DCNN retraining, see figure 3, box 4).

4.1 Network architecture

This section shows our solution for step 3.) in figure 3. We need to solve the classification problem by training a classifier that approximates $F(z_j) \rightarrow Y_j$.

A naive solution to the problem would be to reduce the list z_j along the first axis to a vector of fixed size 1×53 using an operation that does not take into account the order of the lists, like max or $mean$. In the case of a list of associated class probabilities in multi view images predicted by the DCNN, the $mean$ operation on the first axis would result in a 'majority decision' and the max operation would favour the predictions in the list with the highest probability. A multi-layer perceptron (MLP) could then perform a prediction on the reduced list to classify the actual point classes on all 2D images. However, we believe that this solution discards much valuable information, especially since operations like $mean$ are not robust to outlier predictions.

Our solution is inspired by PointNet (Qi et al., 2017), which is actually used for the classification of 3D point clouds. Similarly, it solves this problem by first increasing the point feature size by using an element-wise MLP and then using a reduction function that produces a fixed size feature vector that can be fed into a final classification layer. We have adapted the PointNet architecture to our needs, as shown in figure 4. We first feed each element of the input to four consecutive MLPs, increasing the point feature size to 1×1024. These features are then reduced to a global feature vector and fed to an MLP whose output is concatenated to each list element. After a final reduction, the fixed-size feature vector is then fed to four successive MLPs, which finally output the class distribution probabilities.

4.2 Training

For training, we noticed that selecting the training, validation, and test set by randomly sub-sampling is not suitable, as the classifier overfits to the test data. We therefore selected them by dividing the data sets into different 'scenes' based on their location.

Due to heavy class imbalances, the network was trained using a multi-class focal loss $FL(p_t)$.

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} log(p_t) \quad (1)$$

Whereby p_t in equation 1 is the predicted target label probability. We used dropout after every layer, with a dropout probability of 50%. Additionally, we augmented the data by randomly dropping up to $n_j - 2$ input list elements.

Since the scenes can differ between image and laser scan, it is only possible to correct static objects. We have therefore discarded all dynamic object classes, including 'sky', in the training process.

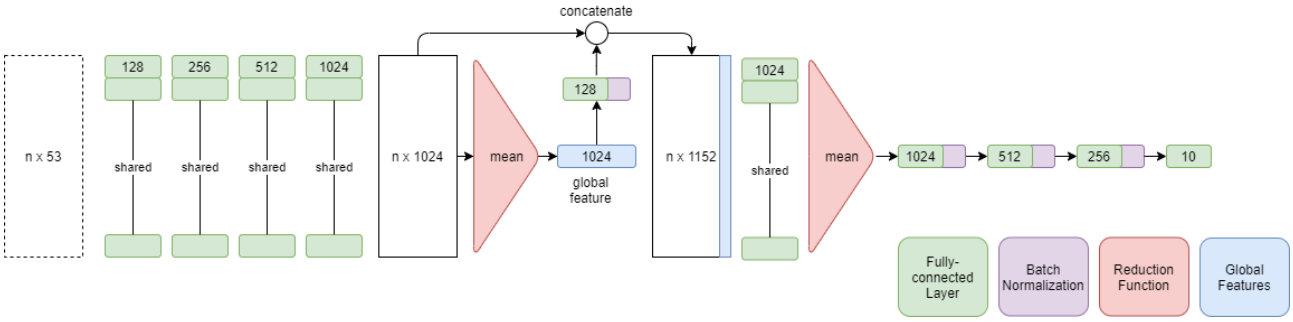

Figure 4. Proposed network architecture for learning to classify the correct label class for a list of multi-view predictions.

4.3 Correcting outliers

This section shows the assignment of the corrected predictions as shown in box 4.), figure 3. The correction network $F(z_j)$ predicts only one class label for all 2D pixels related to a 3D point. Since we assume that every connected 2D pixel belongs to the same object except that a dynamic object appears in the image, we propagate the corrected class $F(z_j)$ to every associated image point as long as the originally predicted class is not dynamic. By doing this, we preserve the spontaneous appearance of dynamic objects in the scene. The corrected label $\hat{Y}_i$ is created in the following way:

$$\hat{Y}_i = \begin{cases} \text{DCNN}(\mathcal{I}_k)_{x_{i,j}} & \text{if DCNN}(\mathcal{I}_k)_{x_{i,j}} \in \text{dynamic} \\ F(z_j) & \text{otherwise,} \end{cases} \quad (2)$$

where $\text{DCNN}(\mathcal{I}_k)_{x_{i,j}}$ is the prediction of the DCNN for image $\mathcal{I}_k$, taken at pixel index $x_{i,j}$ (k being the index of the image containing pixel $x_{i,j}$), and $F(z_j)$ is the corresponding corrected class label. At the end, we reassemble all corrected class labels in the images to create the data set for retraining the DCNN. Whereby 2D pixels, which were predicted as static but do not belong to any 3D point, are removed from the retraining dataset. This leads in some areas to very sparse training examples as can be seen in figure 5.

5. DATA PREPARATION

The data sets used were recorded with a Riegl VMX-250 mobile mapping system. This system captures a maximum of 600'000 3D points per second and has (in our case) two cameras, which capture images at a rate of 1 Hz each. The image size is 2048×2560 in contrast to Cityscapes which has a image size of 1024×2048. The points are acquired with a LiDAR accuracy of 1 cm, with absolute accuracies typically in the range of 10 to 20 cm. In order to project the 3D points into the images we used the calibration matrices and poses obtained by the Riegl system.

To handle occlusion, we have implemented a full ray tracing to connect multi-view image pixel-coordinates. All 3D points are sorted into a voxel grid. Then, when determining the image coordinates of a 3D point, the ray is traced to each camera center in this grid and the point is considered occluded if an occupied cell is found along the ray. We have used 10 cm grid cells for this operation. In addition, we computed a voxel pyramid to speed up the calculation. Although this method is only a rough approximation of the complex interactions between laser and image beams, it should generally improve the data set. Because the computational complexity increases with the distance we rejected 3D points after a maximum distance of 150 m to the camera center.

For the ground truth we have manually annotated 23 images, which can be seen in figure 1. These images belong to four different locations, selected to cover most classes in Cityscapes. We would like to point out that our data set was recorded in Hannover Germany, a city that is also included in the Cityscapes data set.

Finally, to get the uncorrected predictions for our MMS images, we used HRNet, which was trained on Cityscapes. The network and weights were taken from the official repository of the authors[1].

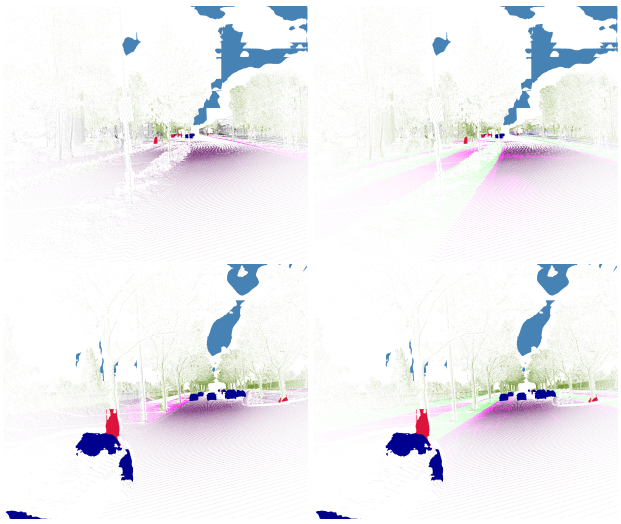

Figure 5. Examples for uncorrected (left) and corrected (right) predictions. (Best viewed digitally.)

6. CORRECTION RESULTS

We divided the data into 85% training, 9% validation and 6% test set. The data consists of several multi-view feature lists $n_j \times 53$, and for each list a class name Y_j. The network was trained with a batch size of 512, for 100 epochs. Table 2 shows that we were able to increase the mIoU by $\sim$ 20%. We calculated the HRNetv2 prediction by extracting the majority decision from the multi-view predictions.

In figure 5 we show examples of images before and after the multi-view correction step. The image classes are merged according to the strategy in equation 2. We have not changed the images in any way to show the sparsity of the corrected images. However, it should be evident that our network was able

[1] https://github.com/HRNet/HRNet-Semantic-Segmentation/tree/HRNet-OCR

Table 1. The table shows the results of the pre-trained DCNN (uncorrected), the retrained DCNN (corrected) and the merged results (merged). The best results per class are shown in bold.

method	road	sidewalk	building	wall	fence	pole	t. light	t. sign	vegetation	terrain	sky	person	rider	car	truck	motorcycle	bicycle	mIoU
uncorrected	0.784	0.136	0.61	0.12	0.489	**0.381**	**0.175**	**0.692**	0.767	0.02	0.642	0.167	0.	0.786	0.1	0.	**0.263**	0.36
corrected	0.822	0.576	0.724	0.363	0.546	0.205	0.09	0.524	0.742	**0.734**	0.877	0.079	0.	0.637	0.089	0.	0.038	0.414
merged	**0.824**	**0.577**	**0.74**	**0.383**	**0.574**	0.2	0.155	0.635	**0.792**	0.73	**0.878**	0.167	0.	**0.787**	**0.101**	0.	0.258	**0.46**

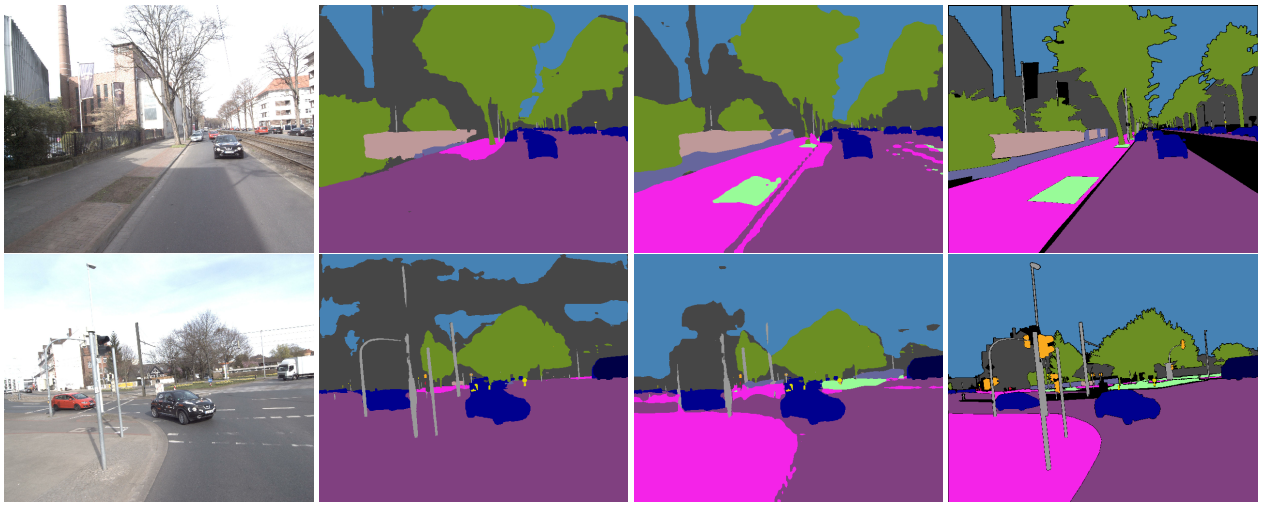

Figure 6. Examples for successfully corrected predictions after retraining. The figure shows, from left to right, the input image, the uncorrected prediction, the corrected prediction, and the manually labelled ground thruth.

Table 2. Results on the test set before (HRNetV2) and after correcting (Corrected) multi view predictions.

Class	HRNetV2	Corrected	Support
road	0.72	**0.84**	883947
sidewalk	0.01	**0.56**	240465
building	0.77	**0.88**	108556
wall	0.0	**0.32**	6709
fence	0.21	**0.38**	1365
pole	**0.41**	0.34	21624
t. light	0.2	**0.32**	7490
t. sign	**0.58**	0.54	850
vegetation	**0.82**	0.78	48390
terrain	0.0	**0.75**	21839
mIoU	37.2	**57.3**	

to successfully restore classes such as sidewalk (pink) and terrain (light green).

Interestingly, some results became worse after the correction step. This could be due to calibration errors between camera and laser scanner. Especially with small objects, this is a problem, because correct labels can 'overflow' into neighboring image areas. For example, a pole could be assigned to the surrounding street class if the calibration is not correct. This type of error leads to label noise in the data set, which can be a reason for the reduced performance.

A special case is the class 'vegetation'. The image label policy of Cityscapes requires that areas visible behind treetops, such as building facades, must be labeled with the tree label. In the work of (Peters, Brenner, 2019) it was shown that many of the laser beams pass through the treetops and thus accumulate tree and facade labels depending on the viewing angle. This could make it difficult for the classifier to assign the correct image labeling, since it is unclear whether a facade or a treetop is shown in the multi-view images.

7. RETRAINING THE DCNN AND RESULTS

This section explains the retraining step as shown in box 5.) in figure 3. To retrain the DCNN, we merged our corrected static classes with the predicted dynamic classes of the DCNN as shown in figure 6. In this section we introduce the following terms for better readability. By the term 'pre-trained DCNN', we mean the DCNN before each correction, see figure 3, box 1.), which outputs 'uncorrected predictions'. Consequently, 're-trained DCNN' means the DCNN after step 5.) in figure 3 which outputs 'corrected predictions'.

For the retraining process we used HRNetv2 and Deeplabv3+ with a Resnet50 encoder. However, HRNetv2 performed slightly worse than Deeplabv3+, so we did not include the network in our results and focused on the results of Deeplabv3+.

First, we concatenated the uncorrected predictions and the RGB values as input to retrain the DCNN. However, we found that in this case the network converges to a local optimum where it just outputs the uncorrected prediction. We therefore decided to retrain the network using RGB input only. The advantage of this approach is that the DCNN generalizes better on the new data. The downside is that some classes perform slightly worse than the uncorrected predictions. However, we think it is a valid approach to merge both, the uncorrected and corrected predictions to get the best of both worlds. We will therefore present three results, the performance of the uncorrected predictions, the corrected predictions and the merged prediction.

The results in figure 6 clearly show that the pre-trained DCNN could not classify the sidewalks (pink), the sky (blue), wall (grey blue) and the terrain (light green). After retraining the DCNN, we successfully restored these classes. Table 1 shows in detail that we were able to increase the Intersection over Union (IoU) in most cases. Furthermore, it shows that after retraining, the mean IoU was increased by 10% on the new dataset.

The table also shows that the IoU slightly decreases for some classes after the correction step. Those cases are mostly belonging to classes of objects with relatively small physical size. We think that our correction step fails in these cases because of calibration errors between camera and LiDAR, as described above.

8. CONCLUSION

We presented a framework for improving semantic segmentation in 2D images on a new data set. Our framework uses aligned point cloud and image data to accumulate multi-view predictions for the same 3D object points. These predictions are then corrected using our neural network and projected back into the images. The corrected images serve as a supervision signal for the pre-trained DCNN to adapt it to the new data and improve its performance in almost all classes.

In the future, we would like to improve the mIoU by creating more advanced techniques for noise correction. We can imagine that by including some spatial information of surrounding 3D points we can improve the correction step. In addition, it might be helpful to adapt the pre-trained neural network to our cameras using a domain adaptation technique like Cycada (Hoffman et al., 2017). The improved classification error would minimize label noise at an early stage of our framework, which could then improve the correction step. If the prediction quality in the images is sufficient, we could image that outliers can be corrected even in an unsupervised way, using a strategy like majority voting.

ACKNOWLEDGEMENTS

This work was funded by the German Research Foundation (DFG) as a part of the Research Training Group GRK2159, 'Integrity and collaboration in dynamic sensor networks (i.c.sens)'.

REFERENCES

Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12), 2481–2495.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., 2018a. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 834–848.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018b. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2), 303–338.

Floros, G., Leibe, B., 2012. Joint 2d-3d temporally consistent semantic segmentation of street scenes. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2823–2830.

Hermans, A., Floros, G., Leibe, B., 2014. Dense 3d semantic mapping of indoor scenes from rgb-d images. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2631–2638.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A. A., Darrell, T., 2017. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*.

Liu, W., Su, F., Huang, X., 2019. Unsupervised Adversarial Domain Adaptation Network for Semantic Segmentation. *IEEE Geoscience and Remote Sensing Letters*, 1-5.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Ma, L., Stückler, J., Kerl, C., Cremers, D., 2017. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 598–605.

Peters, T., Brenner, C., 2019. Automatic generation of large point cloud training datasets using label transfer. *Publikationen der DGPF, Band 28*, 68–82.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Tzeng, E., Hoffman, J., Saenko, K., Darrell, T., 2017. Adversarial discriminative domain adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7167–7176.

Van Engelen, J. E., Hoos, H. H., 2020. A survey on semi-supervised learning. *Machine Learning*, 109(2), 373–440.

Xie, J., Kiefel, M., Sun, M.-T., Geiger, A., 2016. Semantic instance annotation of street scenes by 3d to 2d label transfer. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3688–3697.

Yuan, Y., Chen, X., Wang, J., 2019. Object-Contextual Representations for Semantic Segmentation. *CoRR*, abs/1909.11065.

Zhang, R., Li, G., Li, M., Wang, L., 2018. Fusion of images and point clouds for the semantic segmentation of large-scale 3D scenes based on deep learning. *ISPRS journal of photogrammetry and remote sensing*, 143, 85–96.

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2017. Pyramid scene parsing network. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2881–2890.

Zhou, X., Karpur, A., Gan, C., Luo, L., Huang, Q., 2018. Unsupervised domain adaptation for 3d keypoint estimation via view consistency. *Proceedings of the European Conference on Computer Vision (ECCV)*, 137–153.