Learning an Optimisable Semantic Segmentation Map with Image Conditioned Variational Autoencoder

Pengcheng Zhuang[1(✉)], Yusuke Sekikawa[1,2], Kosuke Hara[1], and Hideo Saito[1]

[1] Graduate School of Science and Technology, Keio University, Yokohama 223-8522, Japan
zpc0113@keio.jp

[2] Denso IT Laboratory, Tokyo 150-0002, Japan

Abstract. Recent semantic segmentation systems have achieved significant improvement by performing pixel-wise training with hierarchical features using deep convolutional neural network models. While the learning process usually requires pixel-level annotated images, it is difficult to get desirable amounts of fine-labeled data and thus the training set size is more likely to be limited, often in thousands. This means that top methods for a dataset can be fine-tuned for a specific situation, making the generalization ability unclear. In real-world applications like self-driving systems, ambiguous region or lack of context information can cause errors in the predicted results. Resolving such ambiguities is crucial for subsequent operations to be performed safely.

We are inspired by work from CodeSLAM where optimizable pixel-wise depth representation is learned. We modify the regression method to work on the pixel-wise classification problem. By training a variational auto-encoder network conditioned with a color image, the computed latent space works as a low-dimensional representation of semantic segmentation, which can be efficiently optimized. As a consequence, our model can correct the error or ambiguity of the prediction during the inference phase given useful scene information. We show how this approach works by giving partial scene truth and perform optimization on the latent variable.

Keywords: Semantic segmentation · Variational autoencoder · Optimization

1 Introduction

Semantic segmentation task aims to predict the categorical label for every pixel in the image, which has been one of the grand challenges in computer vision domain. It is a topic of broad interest because of potential applications like automatic driving. Most of the state-of-the-art semantic segmentation works [4,17,19] are based on FCN [10] liked networks, where performing end-to-end training is possible along with useful hierarchical features to catch both context and localization information. However, the successful training of these networks usually requires large number of annotated training samples, which are usually beyond reach in many cases. That leads to a problem that for most of the current FCN based models, the generalization ability is questioned when applied to dynamic and complex scenes. Ambiguities and errors are common in the results, and in real-world applications, enhancing the prediction accuracy is important before further actions are made.

In this paper, we use image-conditioned variational auto-encoder to encode semantic segmentation map, making it possible to optimize predicted result with changeable latent space. While some models [2,3,16] adopt conditional random fields (CRFs) as a post-processing step to refine the prediction result, our proposed model maintains a trainable end-to-end system by incorporating the stochastic neurons inside the network architecture as in [9,12,15]. Besides, since our work models the conditional distributions of semantic segmentation output by using sampled latent variable, different modes of result can be inferred as the variants change. That is to say, unlike other deterministic models, our model can make diverse predictions and thus initial prediction can be optimized as long as additional scene information provided.

Our work is inspired by CodeSLAM from Bloesch et al. [1] where pixel-wise depth information is encoded with the latent space named code. The code is a compact representation of dense scene geometry, which can be jointly optimized with camera poses when multiple overlapping frames are available. While [1] is a regression model using encoded information to retrieve shape parameters that can't be predicted with a single image, our model is its counterpart in the classification task. Provided additional information of the scene, we can perform optimization on latent space using constraints like consistency of semantic context or geometry. In self-driving systems, depth data, ego-motion or methods like white line detection can serve to provide such additional information, making it able to resolve ambiguities in semantic segmentation results. In our current implementation, instead of combining with other methods or sensors to attain additional information, we manually provide partial scene truth by giving true label of one certain class and evaluate optimization result. This is practical in a similar way to active learning introduced in [11].

Two key contributions of our paper are:

- The extension of the architecture in CodeSLAM that makes it possible to optimize semantic segmentation result on learned models.
- The design of a unique cost function to attenuate error term in learning process using heteroscedatic uncertainty.

In the rest of this paper, we will explain our model and then we show how the learned multimodal distribution of semantic segmentation model can resolve ambiguities and errors of prediction.

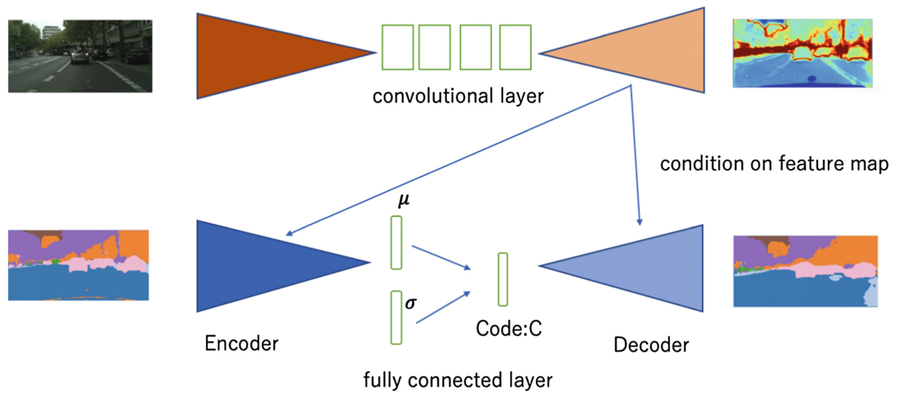

Fig. 1. Illustration of our learning framework. Top part: U-net [13]. Bottom part: VAE. Top left: input color image. Bottom left: input ground truth label. Top right: prediction uncertainty. Bottom right: reconstruction of semantic segmentation (Color figure online)

2 Color Image Conditioned VAE

Similar to CodeSLAM, our model uses an image conditioned variational auto-encoder (CVAE) to learn the semantic representation. The basic learning framework is shown in Fig. 1. The bottom part is a variational auto-encoder network that learns the conditional probability densities of semantic segmentation, with the conditioning feature maps provided by the U-Net [13] on the top part. As described in [1,14], a naive implementation of auto-encoder will be limited by the information bottleneck, making only major traits of input able to be retrieved. By conditioning with feature maps from color image, the auto-encoder part no longer need to encode full scene information. The semantic segmentation prediction SS thus becomes a function (decoder) of image I and c, where c(named code in CodeSLAM) is the latent space composed of a N-dimensional vector.

$$SS = f(I, c) \tag{1}$$

We can say that common semantic segmentation architectures solve a none-code version of the above problem. Due to the introduction of the latent variants (code c), the model becomes able to efficiently produce unlimited number of hypotheses. Our model is a combination of common FCN based semantic segmentation and probabilistic generative model (VAE), where every random sample on the latent space can produce a corresponding segmentation map. Provided useful information about the scene, we can find the optimized code to resolve ambiguities during the inference phase.

2.1 Network Architecture

Our detailed network architecture is provided in Fig. 2. Ground truth segmentation labels are one-hot encoded before being inputted to the Conditional Varia-

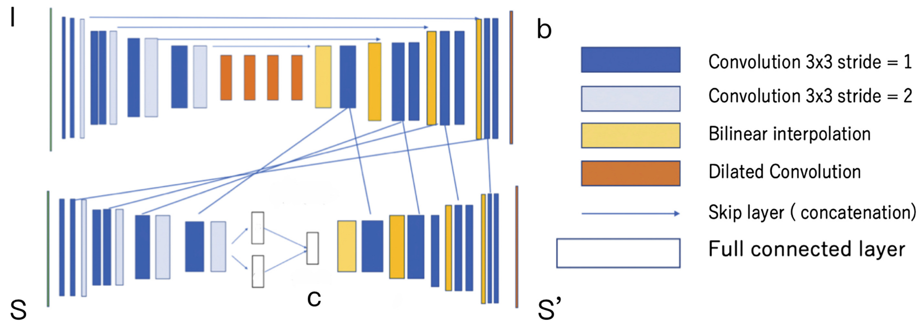
I
b
S
c
S'
Convolution 3x3 stride = 1
Convolution 3x3 stride = 2
Bilinear interpolation
Dilated Convolution
Skip layer (concatenation)
Full connected layer

Fig. 2. Network architecture of our CVAE model. The computed feature maps are concatenated to the corresponding layers of both encoder and decoder (conditioning). The bottleneck of VAE is composed of two fully connected layer, each meaning mean and variance. Then latent space is sampled according to Gaussian distribution. We use stride length 2 to perform down-sampling, and bilinear interpolation to perform up-sampling. While the VAE part outputs S' as the reconstructed segmentation map, the U-Net part outputs b as uncertainty.

tional Auto-encoder. Then latent space is calculated through two fully connected layers in the bottleneck part, each with a dimension of N, computing mean and variance respectively. Then, we sample the latent space according to the Gaussian distribution. During the learning process, a KL-divergence cost is added to reconstruction loss as described in [8]. In the equation below 2, N stands for the length of latent vector (code size). μ and σ represent mean and variance respectively.

$$L_{kld} = \sum_{x=1}^{N} (-\log(\sigma^x) - 1 + \sigma^x + \mu^2) \tag{2}$$

We perform condition on the VAE by concatenating feature maps calculated by the U-Net to corresponding layers (same resolution) in both encoder and decoder part. At each downsampling step we double the number of channels for the feature maps, and at every upsampling step we half the number of feature channels. Between every downsampling and upsampling pair, skip layer connection is used to take advantage of both coarse but discriminative features and fine, shallow features. Unlike the U-Net architecture in CodeSLAM [1] or original paper [13], we stop downsampling at $\times 16$ times and replace the bottleneck part with recently popular dilated convolution [4,17–19]. Dilated convolution enlarges the receptive field of the network while maintaining the resolution by inserting holes in the convolution kernels, which effectively prevents the information loss caused by repetitive downsampling operations.

Besides the prediction of segmentation map S', we predict a pixel-wise uncertainty b to attenuate the cost function. While [1] uses a cost term computing negative log-likelihood of the observed depth, we follow the classification model introduced in [7] to form our own loss function. For every pixel i, S_i' is the predicted logits before the final softmax layer. And b_i is the predicted uncertainty (variance). We sample from these logits T times following Gaussian distribution, each represented by $S_{i,t}$. With the ground truth label being c for pixel i, Eq. 4 gives the uncertainty aware loss function for our model.

$$S_{i,t} = P'_i + b_i\epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, I) \tag{3}$$

$$L_S = \sum_i \log \frac{1}{T} \sum_t \exp(\hat{S}_{i,t,c} - log \sum_{c'} \exp \hat{S}_{i,t,c'}) \tag{4}$$

The network will learn and adjust the pixl-wise uncertainty b so that loss attenuation can be achieved to get better segmentation result, in a similar way to its regression counterpart in CodeSLAM. We also use ReLu as activations for most of the layers, except for the computation of fully connected layer in latent space, the decoder part of VAE and final output where identity activations are used. Especially, the decoder part without nonlinear activations are regarded as linear decoder, allowing the pre-computation of the Jacobians.

2.2 Training

Our network is trained on the CityScapes Dataset [5]. The Cityscapes Dataset is a large dataset mainly focusing on semantic understanding of urban street scenes. The fine labeled dataset we use contains 5000 images, with the training, validation, and test set contains 2975, 500, and 152 images respectively.

We choose the ADAM optimiser with an initial learning rate of 10^{-4}. With a batch size of 8, we train the network for 80 epochs. We also adopt batch normalization [6] for smoother training process.

As is mentioned in Sect. 2.1, our model's loss function is composed of both a KL-divergence cost and an attenuated segmentation loss, given by Eqs. (3) and (5) respectively. We put a relative weight W_k on KL-divergence and found a code size of 128 and weight factor of 8 work fine in our current training model.

$$Loss = L_S + W_k L_{kld} \tag{5}$$

3 Inference and Refinement

During the inference phase, the encoder part of VAE is not used due to the absence of ground truth label. We assign the code with 0 value at first and thus the initial prediction is given in Eq. 6.

$$SS = f(I, 0) \tag{6}$$

This zero code prediction can be regarded as a model close to common single image semantic segmentation, where ambiguities and errors may remain. As we perform optimization on latent space with additional information of the scene, we can get a refined segmentation map. Although it's able to be combined with framework like CodeSLAM to have joint optimization by keeping both geometric and semantic consistency, we use a simpler refinement method to evaluate the optimization validity of our model. In our current work, we give partial truth information of a certain scene and perform optimization on that truth-aware region. Let c be the class that truth is given when M class exists. For every pixel j that belongs to the truth-aware area belong to c, T_i is the true one-hot label, and P_i gives the predicted probability vector.

$$T_i = [0, 0, ...1, ..0] \tag{7}$$

$$P_i = [p_0, p_1, ...p_c, ...p_{M-1}] \tag{8}$$

We use the Euclidean distance between two vectors to measure the semantic error. Especially, the probability vector is computed from code c and image I through the linear decoder according to Eq. 1, derivative of the probability w.r.t the code thus can be computed to allow optimization. The low-dimensional latent space learnt should also encode the semantic correlation among different classes, what we are expecting is that by performing optimization to reduce the semantic error on truth-aware area, prediction of both adjacent and non-adjacent area can also be refined.

4 Experiments

In this section, we first present how our model works by showing different output examples in Fig. 3. Although the zero-code segmentation results are close to those predicted from code encoded by ground truth label, errors and ambiguities appear. Especially in the left and middle columns of Fig. 3, we can observe that zero code prediction show errors in areas close to sidewalk and road, and get confused with bus and car. These errors are corrected in the output predicted from encoded information, i.e., the encoded information brought higher accuracy to the model. The output uncertainty is learned during training to attenuate cost in difficult regions as described in Sect. 2.1.

4.1 Optimization on Partial Region

Since the semantic error term is differentiable w.r.t to the latent space, additional semantic information can serve to optimize the code enhance the prediction. Due to the use of indentity activations in decoder part of VAE, the derivative of probability maps w.r.t to the code are effectively pre-computed, accelerating the evaluation of Jacobians when performing optimization. This may be helpful for faster runtime when realtime performance is required to perform joint optimization, especially in self-driving systems. As mentioned in Sect. 3, we use partial scene truth to perform optimization in our current simple experiment. In the following sections, we will first show the qualitative refined results when given road class truth. Then we compare the quantitative evaluation of class mIoU and pixel accuracy when optimizing using different class truth.

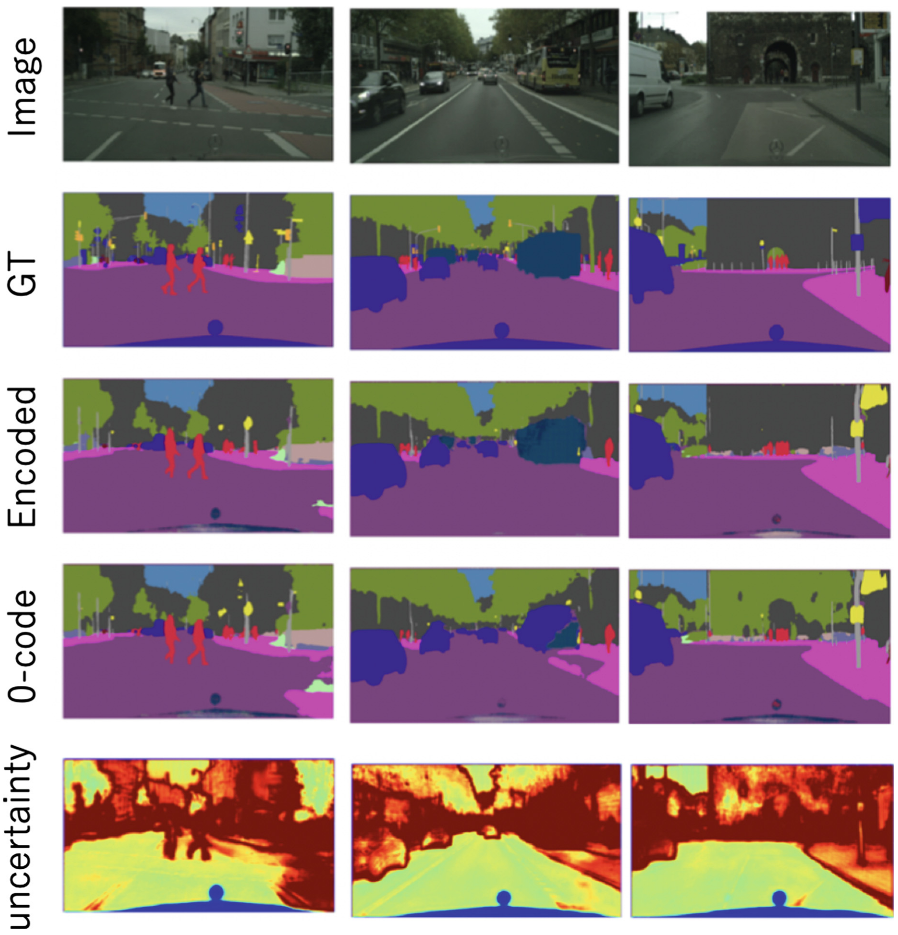

Fig. 3. The learnt output examples of our model on CityScapes Dataset. From top to bottom: color image, ground truth label, reconstructed segmentation map with encoded code, segmentation with zero code, learnt uncertainty. (Color figure online)

4.2 Qualitative Result

In Fig. 4 we compare segmentation results before and after optimization is performed. Although the prediction with zero-code captures scene objects correctly in general, we can see obvious errors occur near boundary regions. After providing road class truth to optimize the code, we can see how errors are corrected in the bottom line of optimized result. In the left example there are no other big changes except for the road and sidewalk class. However, in the left half result, not only road or adjacent regions, details of vegetation and cars' internal areas are also refined (high light with red squares). Although not as accurate as ground truth segmentation, we can observe how the optimized result get refined towards those decoded from ground truth label encoded information.

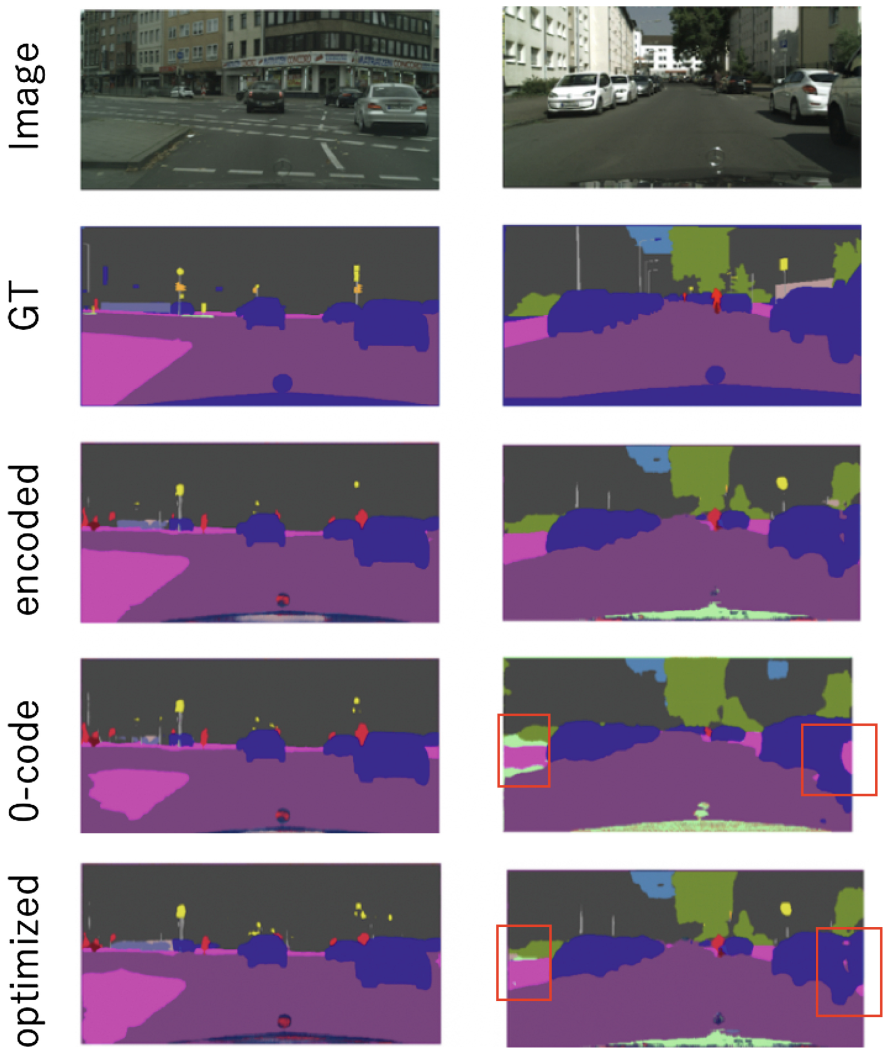

Fig. 4. The qualitative optimization examples on CityScapes Dataset. From top to bottom: color image, ground truth label, reconstructed segmentation map with encoded code, segmentation with zero code, optimized result given truth on road class. (Color figure online)

4.3 Quantitative Evaluation

To evaluate the effectiveness of optimization on partial region, we calculate IoU and pixel accuracy before and after the optimization, using 500 validation images on CityScapes Dataset. The results are shown in Fig. 5 and Table 1.

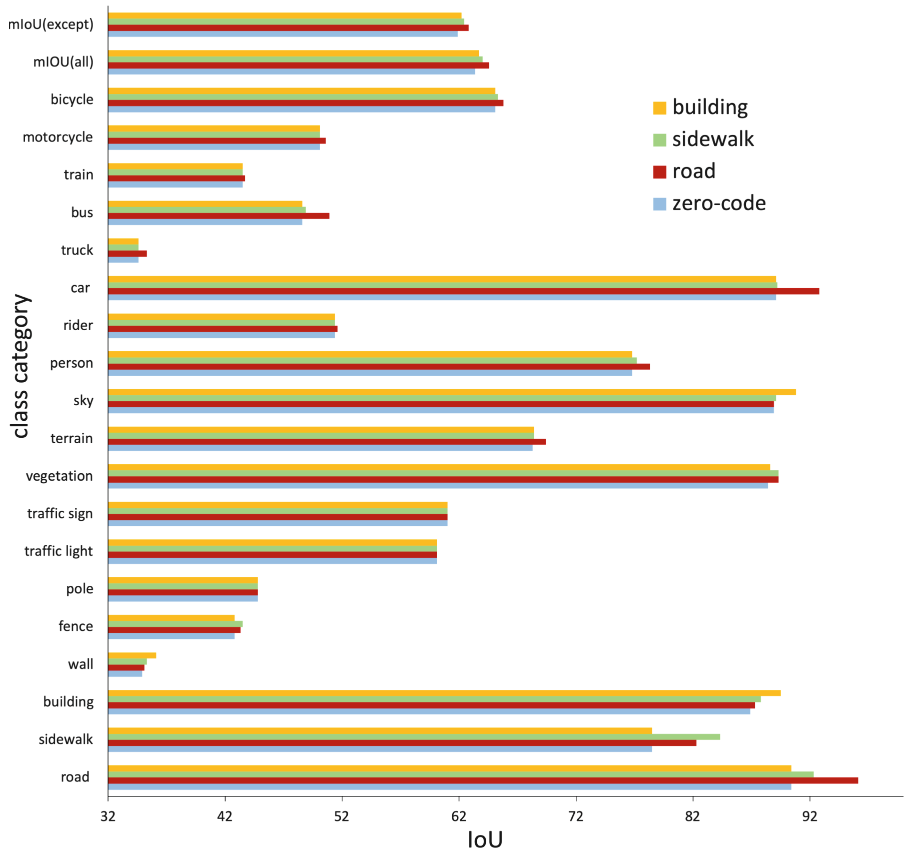

Fig. 5. IoU of our optimization result given different class truth. Besides class-wise IoU, we also calculate the mIoU for all classes (second row) and mIoU excluding truth-given class (first row)

We can see from the IoU result that, by giving one specific class truth as additional information, the refined result naturally improves impressively at both that specific class and adjacent class regions (e.g. road and sidewalk). What's more, in areas that may seem unrelated to hardly get any influence, we can still observe improvement with IoU. Comparing the mean IoU for classes excluding the optimized one (first row of the graph), we find that a general refinement of result is possible with our partial optimization method. On the other hand, we compare the pixel accuracy before and after the optimization, and also calculate the pixel accuracy when simply replacing partial prediction with true labels, without any optimization. Besides an obvious improvement when comparing with result from zero-code prediction, we also find that partial optimization in all the three cases (road, sidewalk and building) performs better than simply replacing data with true labels. That means that optimization on one class region expands its influence to other regions. The low dimensional semantic encoding in our model learns the underlying correlation between different classes, and optimizing on one leads to the refinement of others, even those seemingly unrelated.

Table 1. Pixel accuracy results with different class truth provided.

Method	Pixel acc.
0-code	90.1
Road-replaced	91.9
Road-optimized	**92.4**
Sidewalk-replaced	91.5
Sidewalk-optimized	**92.1**
Building-replaced	90.8
Building-optimized	**91.6**

5 Conclusions

We have shown that our learned image-conditioned representation for semantic segmentation can be effectively optimized with small number of parameters. Our compact representation of segmentation map can retrieve information that is useful for errors and ambiguities to be corrected.

In our future work, for more applicability, we will combine with methods like white line detection to offer truth information about the road, working in a similar way to sensor fusion. Also, combining with CodeSLAM framework to form a joint optimization of both semantic label and scene geometry is also possible. In our current implementation, we use U-Net to extract features, which may not be a good choice for complex urban scene segmentation. In the longer term, we would like to combine state-of-the-art semantic segmentation architecture with our probabilistic inference module. If we can refine the prediction result even on top of those high accuracy methods, our compact representation of segmentation may serve as an optional optimization technique for general models, similar to CRF but different in that our method being unified in an end-to-end inference framework, adjustable as scene and provided information change.

References

1. Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., Davison, A.J.: CodeSLAM-learning a compact, optimisable representation for dense visual SLAM. In: CVPR, pp. 2560–2568 (2018)
2. Chandra, S., Kokkinos, I.: Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian CRFs. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 402–418. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_25

3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. arXiv preprint arXiv:1412.7062 (2014)
4. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 833–851. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_49
5. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding (2016)
6. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
7. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: NIPS, pp. 5574–5584 (2017)
8. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Proceedings of the International Conference on Learning Representations (ICLR) (2014)
9. Kohl, S., et al.: A probabilistic u-net for segmentation of ambiguous images. In: NIPS, pp. 6965–6975 (2018)
10. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
11. Mackowiak, R., Lenz, P., Ghori, O., Diego, F., Lange, O.: Cereals - cost-effective region-based active learning for semantic segmentation. In: BMVC (2018)
12. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st International Conference on Machine Learning PMLR, vol. 32, pp. 1278–1286, 22–24 June 2014
13. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
14. Rumelhart, D.E., Hinton, G.E., William, R.J.: Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318–362. MIT Press (1986)
15. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: NIPS, pp. 3483–3491 (2015)
16. Vemulapalli, R., Tuzel, O., Liu, M.Y., Chellappa, R.: Gaussian conditional random field network for semantic segmentation. In: CVPR (2016)
17. Wang, P., et al.: Understanding convolution for semantic segmentation (2018)
18. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)
19. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR, pp. 2881–2890 (2017)