Error Correction for Dense Semantic Image Labeling

Yu-Hui Huang[1] Xu Jia[2] Stamatios Georgoulis[1]
Tinne Tuytelaars[2] Luc Van Gool[1,3]

[1]KU-Leuven/ESAT-PSI, Toyota Motor Europe (TRACE) [2]KU-Leuven/ESAT-PSI, IMEC
[3]ETH/DITET-CVL

Abstract

*Pixel-wise semantic image labeling is an important, yet challenging task with many applications. Especially in autonomous driving systems, it allows for a full understanding of the system's surroundings, which is crucial for trajectory planning. Typical approaches to tackle this problem involve either the training of deep networks on vast amounts of images to directly infer the labels or the use of probabilistic graphical models to jointly model the dependencies of the input (*i.e. *images) and output (*i.e. *labels). Yet, the former approaches do not capture the structure of the output labels, which is crucial for the performance of dense labeling, and the latter rely on carefully hand-designed priors that require costly parameter tuning via optimization techniques, which in turn leads to long inference times.*

To alleviate these restrictions, we explore how to arrive at dense semantic pixel labels given both the input image and an initial estimate of the output labels. We propose a parallel architecture that: 1) exploits the context information through a LabelPropagation network to propagate correct labels from nearby pixels to improve the object boundaries, 2) uses a LabelReplacement network to directly replace possibly erroneous, initial labels with new ones, and 3) combines the different intermediate results via a Fusion network to obtain the final per-pixel label. We experimentally validate our approach on two different datasets for semantic segmentation, where we show improvements over the state-of-the-art. We also provide both a quantitative and qualitative analysis of the generated results.

1. Introduction

The problem of assigning dense semantic labels to images finds application in many tasks, like indoor navigation [17, 25], human-computer interaction [31], image search engines [39], and VR or AR systems, to name a few. For example, in the autonomous driving task it enables the system to understand what and where the surrounding objects are, in order to plan its path accordingly. The goal in each case is to assign a class label to every pixel, from a pre-defined set of labels.

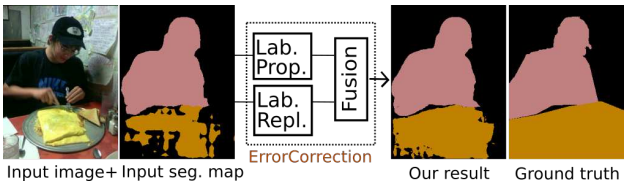

Figure 1. The pipeline of the proposed method. Given an input image and a corresponding initial segmentation map, our model predicts a refined segmentation map by implicitly considering the dependencies in the joint space of both the input (*i.e*. images) and output (*i.e*. labels) variables.

In the literature, several methods [23, 16, 2, 13, 34] have been proposed to tackle this problem. Recently, Deep Convolutional Neural Networks (DCNNs) have become the mainstream for dense semantic image labeling, starting with the Fully Convolutional Network (FCN) proposed by Long *et al*. [28, 36]. Despite their great representational power, feed-forward DCNN-based approaches tend to produce overly smooth results near the object boundaries and do not consider the relations among nearby pixels when predicting the semantic labels [1]. Different strategies have been proposed to cope with these issues. One popular way is to apply probabilistic graphical models, like dense Conditional Random Fields (CRFs) [19, 7], as a post-processing step as is done in [6]. The pairwise potentials in the CRF impose the consistency of labeling between nearby pixels, and the fully connected CRF delineates the object boundary. Although dense CRFs perform well on the refinement of the segmentation results, these pairwise potentials have to be carefully hand-designed in order to model the structure of the output space and it takes quite some parameter hyper-tuning to arrive at a satisfactory result with considerable computation time.

To mitigate these restrictions, we look into ways of

achieving the same goal in a more efficient way. Our starting point is a variant of the current problem: given an RGB image and an initial estimate of the segmentation map, derived from any dense labeling approach, we seek to estimate a refined segmentation map. By doing so, we can exploit the dependencies in the joint space of input image and output labels. We propose a parallel architecture, based on encoder-decoder networks, to deal with the two main sources of error coming from the initialization. First, a *LabelPropagation* network exploits the context information to predict a pair of displacement vectors $(\Delta x, \Delta y)$ per pixel, *i.e*. a 2D displacement field, in order to propagate labels from nearby pixels to refine the object's shape. Obviously, propagating existing labels would not correct cases where the initial labels of all nearby pixels are erroneous and new ones need to be generated. In this case, a second *LabelReplacement* network, which runs in parallel with the *LabelPropagation* network, generates new labels directly from the input pair of RGB image and initial segmentation map. As a final stage, a *Fusion* network combines the results of these parallel branches by predicting a mask to obtain the optimal label for each pixel. Fig. 1 gives an overview of our pipeline.

Our contributions can be summarized as follows: (1) We introduce an efficient post-processing technique for error correction in dense labeling tasks, that works on top of any existing dense labeling approach. (2) We propose an end-to-end pipeline that employs different correction strategies by propagating correct labels to nearby pixels (*LabelPropagation* network), replacing the erroneous labels with new ones (*LabelReplacement* network) and fusing the intermediate results (*Fusion* network) in a multi-task learning manner. Different from other work [11], our method tackles the problem in a parallel rather than sequential way. (3) We show that our model is able to improve two semantic segmentation models for two different tasks, object segmentation and scene parsing.

The paper is organized as follows. Sec. 2 positions our work w.r.t. earlier work. Sec. 3 describes the proposed architecture for error correction in dense semantic labeling. Experimental results are presented in Sec. 4. Sec. 5 concludes the paper.

2. Related Work

The literature on dense semantic labeling is substantial. We consider three main categories of related papers.

Deep learning The great success of deep learning techniques, such as DCNNs [22], in the image classification and object recognition tasks [20, 37, 38] has motivated researchers to apply the same techniques for dense labeling tasks, like semantic segmentation. First, Long *et al*. [28, 36] transformed existing classification CNN models into FCNs by replacing fully connected layers with convolutional ones such that the network can output label maps. Next, Badrinarayanan *et al*. [3] proposed an encoder-decoder architecture with skip connections to up-sample the low-resolution feature maps to pixel-wise predictions for segmentation. Many following works explore to include more context knowledge. On the one hand local information is important for pixel-level accuracy; on the other hand integrating information from global context can help with local ambiguities. The most characteristic works involve the use of dilated convolutions [41, 7, 32], multi-scale prediction [9, 35, 42], attention models [5, 1] and feature fusion [26, 33]. Despite the great representational power of DCNNs, their inability to capture the structure of the output labels negatively affects the performance of dense labeling tasks, especially near the object boundaries. In particular, feed-forward DCNNs do not explicitly consider the relations among nearby pixels in a local neighborhood of the label space.

Probabilistic graphical models As explained above, the DCNNs' inherent invariance to spatial transformations also limits their spatial accuracy in semantic segmentation tasks. A second line of work explicitly handles this inability by trying to jointly model the dependencies of both the input (*i.e*. images) and the output (*i.e*. labels) variables. The most common approach is to apply CRFs [21] as a post-processing stage of a DCNN. The DeepLab models [6, 7] use the fully connected pairwise CRF by Krähenbühl and Koltun [19] to refine the segmentation result of the DCNN. Their models incorporate prior knowledge about the structure of the output space in the pairwise potential term to enforce consistency among neighboring or "similarly-looking" pixels. In general, in all CRF-based approaches the pairwise potentials have to be carefully hand-designed in order to model the structure in the output space and it takes expensive parameter hyper-tuning to arrive at a satisfactory inference. Another relevant work is CRF-RNN [43], which uses a neural network to approximate the dense CRF inference process to obtain a good semantic segmentation result. However, their model still requires considerable time to do inference.

Error correction Most recently, a third line of work goes beyond the restrictions imposed by DCNNs and CRFs and tries to model the joint space of input and output variables. These approaches solve a variant of the traditional dense labeling task: given the input image and an initial estimate of the output labels a network is trained to predict new refined labels, thus being implicitly enforced to reason about the joint input-output space. These methods come in two flavors, the transform-based ones [41, 24, 40, 30] that learn to directly predict new labels from the initial estimate, and the residual-based ones [4] that estimate residual corrections which are added to the initial estimate. Gidaris and Komodakis [11] combined these two flavors for the dense disparity estimation task by proposing a sequential DCNN

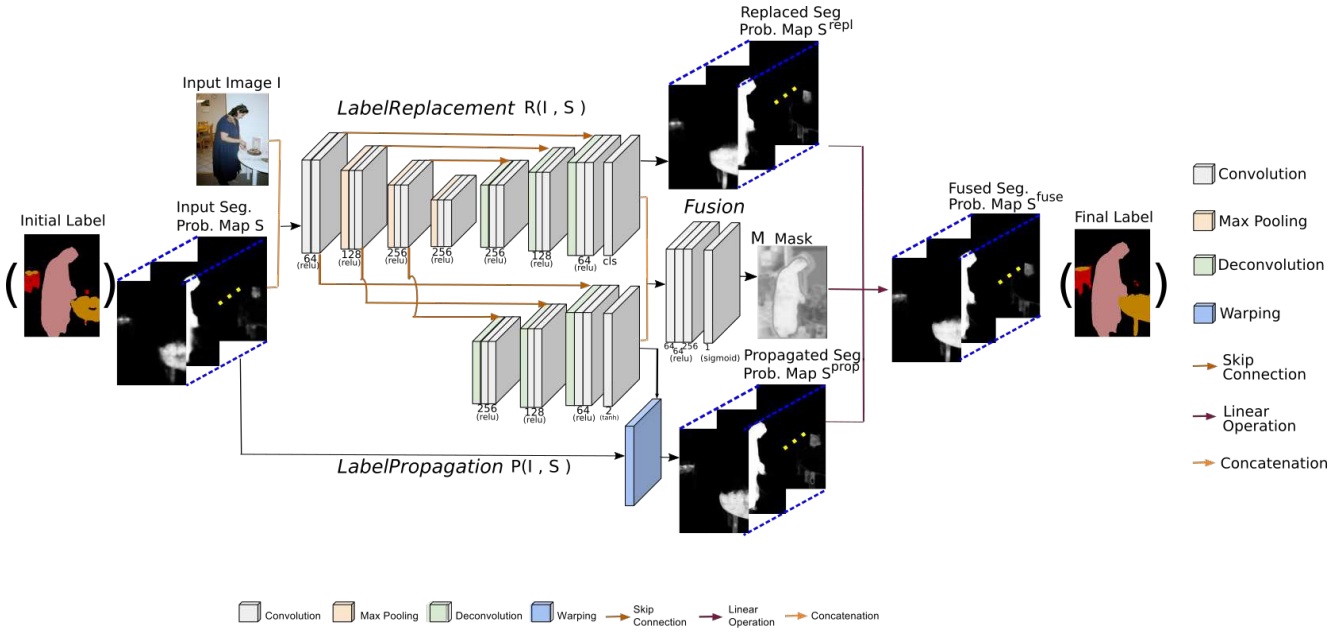

Figure 2. The architecture of our pipeline. The *LabelPropagation* network P propagates probability distributions from nearby pixels to refine the object boundaries. In parallel, the *LabelReplacement* network R predicts a new segmentation probability map directly from the input pair of RGB image and initial segmentation map. Finally, the *Fusion* network M combines the results of these branches with a predicted mask to obtain the optimal labeling. The image in the parenthesis denotes the colored label map.

architecture that is end-to-end trainable. Their approach detects the errors in the initial labels, then replaces the incorrect labels with new ones, and finally refines the labels by predicting residual corrections. Although this method provides good results for improving the continuous values in the dense disparity estimation task, its residual correction stage is difficult to apply to discrete, dense labeling tasks such as semantic segmentation. Different from that method, we elaborate two branches that account for different types of errors: one for propagating existing labels from nearby pixels and the other for predicting new labels. Finally a fusion module is added to take advantage of both branches. Moreover, these two branches run in parallel instead of sequentially, thus allowing for faster inference times.[1]

3. Our Approach

Given an input RGB image I and an initial segmentation probability map S, we propose an end-to-end pipeline for error correction (see Fig. 2) which is built upon three networks, *i.e*. the *LabelPropagation*, *LabelReplacement* and *Fusion* networks. This section provides the details.

[1]Since the source code for their method is not available, and it is not trivial to directly apply it to dense labeling tasks as it was originally designed for discrete labels, we can not generate numerical or visual comparisons in the experiments section.

3.1. LabelPropagation network

We propose to estimate a displacement vector $(\Delta x, \Delta y)$ for each pixel, *i.e*. a 2D displacement field, in order to propagate labels from nearby pixels. A warping layer is followed to apply the estimated displacements in order to arrive at an improved segmentation probability map. Inspired by [44], we adopt an encoder-decoder architecture with skip connections for the displacements estimation, which is denoted as *LabelPropagation* network P. Our work resembles flow-based networks [44, 27], but unlike those our network learns to predict the displacements from the joint space of both the input and the output variables instead of finding correspondences among different views.

To sum up, given an input image I and the initial segmentation probability map S our goal is to train a network P that computes an improved segmentation probability map S^{prop} by re-sampling S according to the predicted 2D displacement field. It can be formulated as minimizing the loss function between S^{prop} and the ground truth segmentation map S^{gt},

$$\mathcal{L}_{prop} = \frac{1}{|\mathcal{D}|} \sum_{<I,S,S^{gt}> \in \mathcal{D}} \mathcal{L}(S^{gt}, P(I,S)), \quad (1)$$

where $\mathcal{D}$ is the training dataset, $P(\cdot)$ refers to the *LabelPropagation* network whose parameters we aim to optimize, and $\mathcal{L}$ denotes the cross-entropy loss.

The *LabelPropagation* network P aims at leveraging the context information from the probability distribution of nearby pixels to predict a pair of displacement vectors $(\Delta x, \Delta y)$, one for each direction, such that a pixel's probability distribution can be re-estimated with respect to its neighbors. Here, $(\Delta x, \Delta y)$ denotes the displacement vectors where the model samples the probability distribution from. For every pixel (x_i, y_i) in S, the coordinates w.r.t. the ones after propagation (x_i^{prop}, y_i^{prop}) are associated as,

$$x_i = x_i^{prop} - \Delta x_i, y_i = y_i^{prop} - \Delta y_i. \quad (2)$$

Finally, the initial probability map S is warped according to the estimated displacement vectors to generate the refined probability map S^{prop}. Regarding the warping operation, we use the bilinear sampling kernel in the same way as in [15] to allow for end-to-end training,

$$S_i^{prop} = \sum_{k \in N(x_i, y_i)} S_k (1 - |x_i - x_k|)(1 - |y_i - y_k|), \quad (3)$$

where S_i^{prop} denotes the value of the i-th pixel at (x_i^{prop}, y_i^{prop}) in the output S^{prop}, and $N(x_i, y_i)$ is the 4-neighborhood region of the pixel at (x_i, y_i) in the input S. Its gradients w.r.t. the parameters for displacement estimation can be efficiently computed as in [15].

3.2. LabelReplacement network

As explained in the previous section, the *LabelPropagation* network P is able to correct the segmentation error by propagating the possibly correct labels into their neighborhood. However, it fails to correct the labels when almost all pixels in a region have initially wrong labels. To deal with this case, we propose to feed both the input image I and the initial segmentation probability map S into a fully convolutional *LabelReplacement* network R to directly recompute a new segmentation probability map S^{repl}. The network re-estimates a probability vector for each pixel, but this time based on both its appearance and the probability distribution of its neighbors. Following the same encoder-decoder architecture as in our *LabelPropagation* network, we replace the last layer of the *LabelPropagation* network with a convolutional layer to output the new segmentation probability map.

In short, given an image I and its corresponding initial segmentation probability map S, we train a network *LabelReplacement* network R to predict a new segmentation probability map S^{repl} based on the initial one S. The task can be formulated as minimizing the cross-entropy loss between the newly generated segmentation map S^{repl} and corresponding ground truth labels S^{gt},

$$\mathcal{L}_{repl} = \frac{1}{|\mathcal{D}|} \sum_{<I,S,S^{gt}> \in \mathcal{D}} \mathcal{L}(S^{gt}, R(I, S)). \quad (4)$$

3.3. Fusion network

The *LabelPropagation* and *LabelReplacement* networks work in parallel and are specialized at correcting different types of errors. On the one hand, the *LabelPropagation* network P takes into account the nearby pixels and their corresponding class probabilities to propagate the probability vector based on the appearance similarity. On the other hand, the *LabelReplacement* network R re-estimates the class labels pixel by pixel. To get the best of both worlds, we combine the outputs of these two parallel branches using a *Fusion* network M, and train the whole architecture jointly. Since the two branches complement each other, our combined model can benefit from a joint training by enforcing each branch to focus on what they are specialized at and leave for the other branch what they are not good at. The overall pipeline, including all three networks, can be found in Fig. 2.

Design-wise, we use a shared encoder to learn features for both sub-tasks, *i.e*. the *LabelPropagation* and *LabelReplacement* networks, and to also reduce the total number of parameters to be optimized. The network then splits into two different decoders in a branched manner, one for predicting the displacement and the other for directly predicting new labels. At the final stage, to combine the intermediate results from the two branches, we add the *Fusion* network M that takes those intermediate results as input, and predicts a mask m to generate the final segmentation result. The final result is then computed as a weighted average of the two branches' output in pixel-level,

$$S^{fuse} = m \odot S^{prop} + (1 - m) \odot S^{repl}, \quad (5)$$

where S^{prop} and S^{repl} are the intermediate segmentation probability maps of the two branches and $\odot$ denotes element-wise multiplication. Now, the overall loss function can be formulated as:

$$\mathcal{L}_{fuse} = \mathcal{L}(S^{gt}, S^{fuse}) + \mathcal{L}(S^{gt}, S^{prop}) + \mathcal{L}(S^{gt}, S^{repl}). \quad (6)$$

3.4. Network architecture

The *LabelPropagation* and *LabelReplacement* networks share the base architecture, which is based on fully-convolutional encoder-decoders. For the encoder, there are four blocks with each one containing two convolutional layers with kernel size 3x3 and a max pooling layer. For the decoders, there are three blocks containing one bilinear upsampling layer and two convolutional layers with kernel size 3x3. We add three skip connections at the beginning of the three blocks to incorporate information from different resolutions. This has been shown to be helpful in producing more details in the decoding process [29, 27]. The *Fusion* network predicts a mask to combine both the *LabelPropagation* and *LabelReplacement* networks. It has three

convolutional layers with kernel size 3x3 and another convolutional layer to generate the one-channel mask. More details on the network hyper-parameters (*e.g.* feature map size, number of channels) can be found in the supplementary material of our arXiv paper [14].

3.5. Training details

Regarding the training details, we initialize the weights in our networks with Xavier initialization. To learn the network parameters, we adopt the ADAM optimizer [18] with a learning rate of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a batch size of 8. The overall training procedure includes about 20,000 iterations. For data augmentation, we adopt random mirror, resize between 0.5 and 1.5 for all datasets, and crop to a fixed size according to each dataset. The input image is then normalized to [-1,1] and the corresponding initial segmentation probability map is applied using the softmax operation.

4. Experiments

To demonstrate the effectiveness of the proposed method, we evaluate it on two dense labeling tasks, that is, object semantic segmentation and scene parsing. We also analyze the influence of each component by comparing their performance when trained independently. In the following experiments we apply our models on top of semantic segmentation results of several state-of-the-art approaches.

4.1. Datasets

To evaluate the performance of our method on the object semantic segmentation task, we choose the PASCAL VOC2012 segmentation benchmark [10] as our testbed. In total, it consists of 20 classes plus the background class. The performance is measured in terms of mean intersection-over-union (IoU). Including the additional augmented set annotated by [12], there are 10,582 images for training, 1,449 images for validation and 1,456 images for testing.

Regarding the scene parsing task, we work on the Cityscapes dataset [8]. It contains 19 classes, representing objects observed in driving scenarios, plus 1 void class, which is not taken into account during evaluation. In total, there are 5,000 images, divided into a training set with 2,975 images, a val set with 500 images and a test set with 1,525 images. The images are fully annotated with pixel-wise labels. There are also 20,000 coarsely annotated images provided, but we do not use them in our experiments. For evaluation, we also adopt the metric of mean IoU.

4.2. PASCAL VOC 2012 benchmark

For the PASCAL VOC2012 segmentation benchmark, we apply our networks on DeepLab v2-ResNet (multi-scale) [7]. In particular, we first run the inference of the model to obtain the initial segmentation probability maps on the train+aug and val sets. Note that, the model was trained on the training set without finetuning on the val set[2]. Using the image and corresponding initial segmentation probability maps as input, we train our models on the training set and evaluate them on the val set.

Table 1. Results of applying our error correction models on top of DeepLabv2-ResNet on the PASCAL VOC 2012 val set.

Method	Training	mIoU
Deeplab v2-ResNet (multi-scale)	independently	76.5
+ Dense CRF [7]		77.7
+LabelPropagation (ours)	independently	77.9
+LabelReplacement (ours)	independently	77.0
+Full model (ours)	jointly	**78.2**

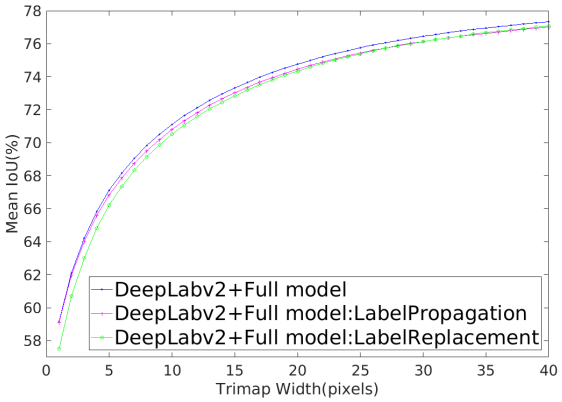

Figure 3. Trimap plot of our full model and its intermediate branches, *i.e. LabelPropagation* and *LabelReplacement* networks, on PASCAL VOC 2012.

Table 1 summarizes the results of our ablation study. Here, the different proposed networks are trained independently and applied on top of the DeepLab v2-ResNet segmentation result. From this table, we can see that adding only the *LabelPropagation* network on top of DeepLab brings 1.4% improvement compared to the baseline, while adding only the *LabelReplacement* network brings 0.5% improvement. When we train the *LabelPropagation* and *LabelReplacement* networks together with the *Fusion* network, which from now on will be referred to as our full model, this brings the highest improvement, 1.7%.

So far, we have evaluated the performance of the *LabelPropagation* and *LabelReplacement* networks when trained independently. Next, we investigate the intermediate results generated by these two networks when training them jointly with the *Fusion* network. In this case, the *LabelPropagation*

[2]The model is provided by the authors.

Table 2. Quantitative results in per-class IoU on the PASCAL VOC 2012 test set.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
Deeplab	91.5	58.8	90.3	64.6	76.0	94.3	88.6	91.4	33.6	87.5	67.3	88.3	91.3	86.6	86.2	61.7	87.9	58.8	86.0	73.5	79.1
+dense CRF [7]	92.6	60.4	91.6	63.4	76.3	95.0	88.4	**92.6**	32.7	**88.5**	67.6	**89.6**	92.1	87.0	**87.4**	**63.3**	88.3	60.0	**86.8**	74.5	79.7
Ours (+full model)	**92.9**	**63.2**	**91.8**	**66.7**	**77.3**	**95.4**	**89.1**	92.3	**35.4**	88.0	**69.5**	89.1	**92.3**	**87.2**	87.3	**63.3**	**88.6**	**61.8**	86.6	**75.1**	**80.4**

Table 3. Quantitative results on the Cityscapes val set.

Method	road	sidewalk	building	wall	fence	pole	tr.light	tr.sign	vege.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
ENet[32]	94.4	71.2	85.2	**46.1**	44.2	45.1	43.2	53.7	87.8	52.4	89.6	61.1	**42.2**	87.8	42.7	60.5	46.0	**29.0**	59.5	60.1
Ours (+full model)	**96.5**	**75.1**	**87.7**	46.0	**45.8**	**49.6**	**48.2**	**59.8**	**88.6**	**58.1**	**92.0**	**65.0**	41.7	**88.0**	**45.4**	**62.0**	**48.8**	28.6	**63.0**	**62.6**

network scores 77.8% while the *LabelReplacement* network scores 78.0%. For joint training, the *LabelReplacement* network shows 1% improvement compared to an independent training, while the performance of the *LabelPropagation* network remains roughly the same. The improvement of our full model is 1.7% compared to the baseline. We conclude that a joint training brings further improvement compared to an independent training.

Since the *LabelPropagation* and *LabelReplacement* networks complement each other, we hypothesize that we benefit from their joint training by enforcing each network to focus on what they are specialized at and leave for the other network what they are not good at. To prove this point, we show the trimap result in Fig. 3 which quantifies the performance at the object boundary region (details of trimap are described in Sec. 4.4). It shows that the *LabelPropagation* branch outperforms the *LabelReplacement* branch at pixels near the boundary region, which indicates that our full model indeed relies more on this branch for the object's boundaries. When we train the two networks jointly, the *LabelPropagation* branch focuses on the object boundaries, and as such the *LabelReplacement* branch can pay less attention to these regions where it does not perform well and put more emphasis on the object's inner part.

Visualizations of the segmentation results using our full model, and baseline models (DeepLabv2-ResNet and DeepLabv2-ResNet + Dense CRF), can be found in Fig. 4. In the first and the third row, the human leg becomes more delineated after applying our method. Also for the *car* class, the images from the rows other than the second row have better shape compared to both baseline methods. For the *bus* in the second and fifth row of the figure, our result looks closer to the ground truth.

Regarding the performance on the test set of PASCAL VOC 2012, we directly apply our networks (full model) on top of precomputed semantic segmentation results on the test set. Table 2 summarizes the per-class performance of the compared methods based on the DeepLab-v2-ResNet. For DeepLab-v2-ResNet, adding Dense CRF brings the performance from 79.1% up to 79.7%, while adding our full model further improves it to 80.4% (+1.3%). Our method achieves top performance in classes related to autonomous driving (cf. gray columns in Table 2), which indicates its added value in such tasks.

Compared to DeepLab-v2-CRF, our full model scores 0.5% and 0.7% higher on the PASCAL VOC val and test set, respectively. In terms of speed, the 10-iteration mean-field dense CRF implementation takes 2,386 ms/image on average on an Intel i7-4770K CPU, while our full model takes 396 ms/image on average on an NVIDIA Titan-X GPU, which is about six times faster than dense CRF. In addition to computational efficiency, our model is able to be plugged into any deep segmentation network for end-to-end training. There is no extra hyper-parameter tuning specific to our model.

4.3. Cityscapes benchmark

For the Cityscapes segmentation benchmark, we apply our full model on top of the ENet [32] segmentation results. For that, we first run the inference of ENet on the downsized images (512x1024) from Cityscapes train/val set to obtain the initial segmentation probability maps. The results of the val set serve as our baseline. Using the corresponding images and probability maps from the train set as inputs, we train our full model, and then test it on the val set. As mentioned previously, we only adopt the train set with the fine annotations for training our error correction models. For evaluation, we first upsample the segmentation result back to the original size and use the official evaluation scripts to obtain the mean IoU performance.

Table 3 presents the mean IoU of our full model and the baseline on the Cityscapes val set. The mean IoU of our model scores 62.6% while the baseline method scores 60.1%. In terms of class-wise performance, our method is superior in almost every class.

Fig. 5 presents qualitative results on the Cityscapes val set. We observe that our method is able to find a large portion of initially wrongly classified parts on the road, which are removed after applying our method. For example, the bike and its rider in the first row are better delineated compared with the baseline method. The same goes for the skyline above the buildings in the second row.

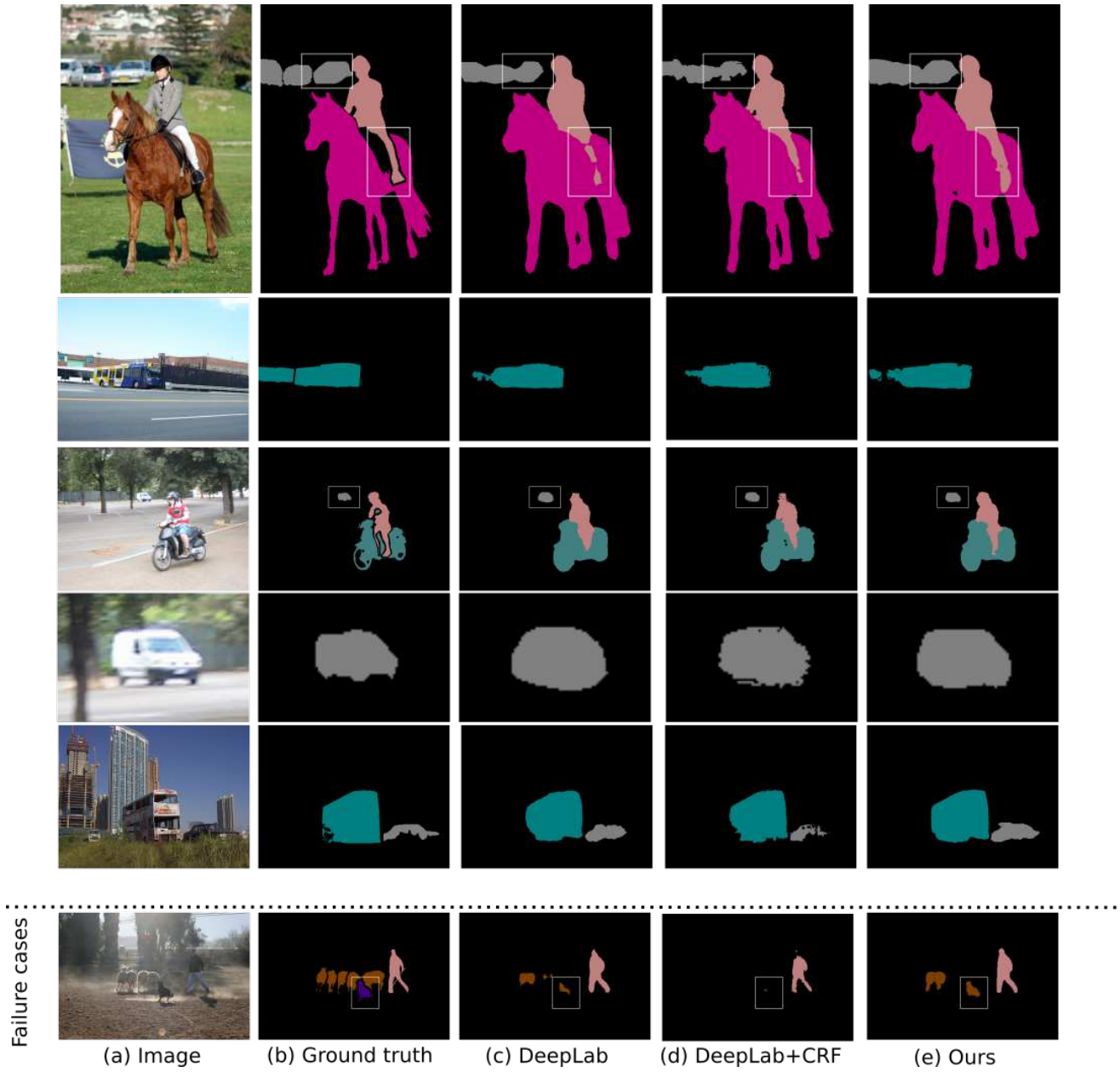

Figure 4. Visualization results on the PASCAL VOC 2012val set. The first five rows present successful cases while the last row present failure cases. For each row, we present (a) input image, (b) ground truth label, (c) baseline DeepLab-v2 result, (d) DeepLab-v2+Dense CRF result, (e) result after applying our full model.

In general, the numerical and visual improvements for both PASCAL VOC and Cityscapes val/test sets are consistent and show a steady increase in performance when applying our full model.

4.4. Error analysis

In this section, we analyze the improvement our method brings to the object boundaries and discuss its failure cases.

Trimap Following previous works [7, 1], we quantify the performance near object boundaries. For the PASCAL VOC 2012 dataset, we compute the performance on the narrow band ('trimap') near the object boundary. Two examples can be found in the left part of Fig. 6. The right part illustrates a plot of mean IoU versus various trimap widths ranging from 1 to 40 pixels. It shows that our full model (in blue dotted line) outperforms the baseline and CRF-based method by a certain margin near the object boundaries.

Failure cases Here, we further analyze some failure

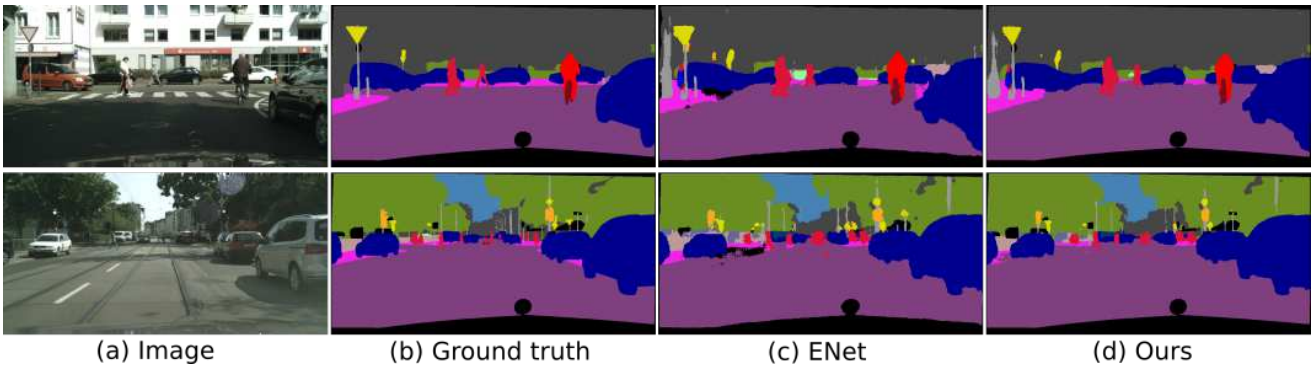

Figure 5. Visualization results on the Cityscapes val set. For each row, we present (a) input image, (b) ground truth label, (c) baseline ENet result, (d) segmentation result with our full model.

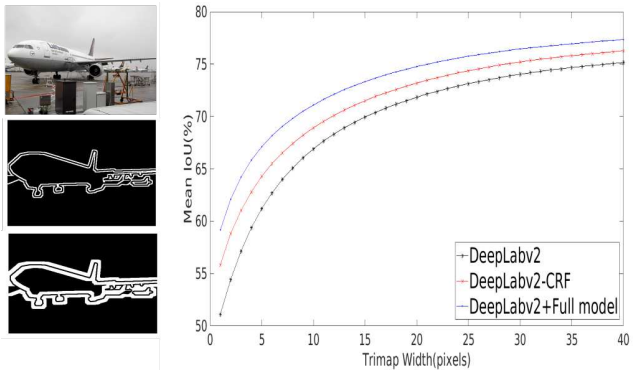

Figure 6. Performance in mean IoU near object boundaries('trimap'). The left side illustrates two examples of trimap size in three (middle) and in ten (bottom). The right side shows the mean IoU at different trimap sizes from 1 to 40 on PASCAL VOC 2012.

cases and conclude that our method can better delineate the boundary but has difficulties in correcting the wrong class labels when a major part of the object is initially wrongly labeled. The last row of Fig. 4 illustrates one typical such example. Our model can better recover the shape of the dog but with the wrong class label.

4.5. Joint training with the segmentation model

In this section, we explore the possibility to train the error correction model end-to-end together with the semantic segmentation model that provides the initial probability maps. We first train DeepLab-v2 model on the PASCAL VOC 2012 train set for 20,000 iterations. After that, we add our *LabelPropagation* module right after the Atrous Spatial Pyramid Pooling (ASPP) module of DeepLab-v2 and jointly train both parts for another 20,000 iterations.

The mean IoU of the DeepLab-v2 (single scale) scores 74.6 / 75.9% before / after adding the *LabelPropagation* module. If we train the *LabelPropagation* module without joint training, the mean IoU is 75.4%. The result shows that our error correction module has further potential when jointly trained with a semantic segmentation network.

5. Conclusion

We have presented two strategies for error correction in dense labeling prediction, and a final model that combines the advantages of these two strategies. Our experiments show that our full model improves over state-of-the-art semantic segmentation models for the object semantic segmentation and scene parsing tasks. Compared to other post-processing methods, our approach provides a simpler solution by considering nearby context information for label propagation and at the same time it directly generates new labels for initially wrongly labeled regions. In the future, we plan to further reduce the network's size in order to allow for even faster inference times.

Acknowledgments

This work was supported by Toyota Motor Europe and the FWO project SFS: Structure From Semantics. We would also like to acknowledge the NVIDIA Academic Hardware Grant for providing us with GPUs.

References

[1] I. K. Adam W Harley, Konstantinos G. Derpanis. Segmentation-aware convolutional networks using local attention masks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 7

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition*, 2014. 1

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 2

[4] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742, 2016. 2

[5] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3640–3649, 2016. 2

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Representations*, 2015. 1, 2

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 1, 2, 5, 6, 7

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 5

[9] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015. 2

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 5

[11] S. Gidaris and N. Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7187–7196, 2016. 2

[12] B. Hariharan, P. Arbelaez, L. D. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 991–998, 2011. 5

[13] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*. Springer, 2014. 1

[14] Y.-H. Huang, X. Jia, S. Georgoulis, T. Tuytelaars, and L. V. Gool. Error correction for dense semantic image labeling. *arXiv preprint arXiv:1712.03812*, 2017. 5

[15] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer network. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 4

[16] J. Kim and K. Grauman. Shape sharing for object segmentation. In *European Conference on Computer Vision*. Springer, 2012. 1

[17] J. Kim and H. Jun. Vision-based location positioning using augmented reality for indoor navigation. *IEEE Trans. on Consum. Electron.*, 54(3):954–962, Aug. 2008. 1

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[19] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011. 1, 2

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[21] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001. 2

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[23] V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *European Conference on Computer Vision*. Springer, 2008. 1

[24] K. Li, B. Hariharan, and J. Malik. Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3659–3667, 2016. 2

[25] Y. Li and S. Birchfield. Image-based segmentation of indoor corridor floors for a mobile robot. In *IROS*, pages 837–843. IEEE, 2010. 1

[26] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 2

[27] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. *arXiv preprint arXiv:1702.02463*, 2017. 3, 4

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1, 2

[29] X. Mao, C. Shen, and Y. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems*, 2016. 4

[30] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 2

[31] M. Oberweger, P. Wohlhart, and V. Lepetit. *Hands Deep in Deep Learning for Hand Pose Estimation*, pages 1–10. 2015. 1

[32] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 2, 6

[33] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016. 2

[34] A.-I. Popa and C. Sminchisescu. Parametric image segmentation of humans with structural shape priors. In *Asian Conference on Computer Vision*. Springer, 2016. 1

[35] A. Roy and S. Todorovic. A multi-scale cnn for affordance segmentation in rgb images. In *European Conference on Computer Vision*, pages 186–201. Springer, 2016. 2

[36] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):640–651, 2017. 1, 2

[37] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 2

[39] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 157–166. ACM, 2014. 1

[40] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. 2016. 2

[41] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2

[42] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[43] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015. 2

[44] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European conference on computer vision*, 2016. 3