Supplementary document for IROS submission: *CP-loss*

March 5, 2021

Zhenhua Xu, Yuxiang Sun, Ming Liu

1 Introduction

In this supplementary document, the details of the loss function design and hyper-parameter tuning are discussed. More equations and figures are provided.

1.1 Motivation

To relieve the occlusion issue and the limitation of computation resources of online road curb detection, we propose to detect road curbs offline as prior information to assist the autonomous vehicles. The detected road curbs could be used to create high-definition (HD) maps. Normally, we first predict the segmentation map of the input aerial images and then conduct a series of post-processing to extract the graph structure of road curbs. CP-loss is designed to improve the quality of segmentation maps by preserving connectivity, which can effectively improve the topological correctness of the final result. In our experiment, CP-loss is implemented by CUDA, thus it has almost the same efficiency performance as other loss functions.

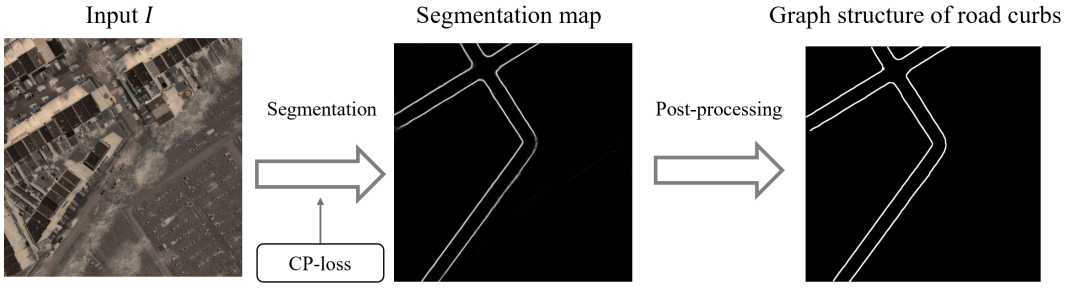


Figure 1: The schematic diagram of offline road curb detection. CP-loss is utilized to train the semantic segmentation network for better connectivity. The obtained graph of road curbs can be used to create HD maps for autonomous vehicles, and it is quite helpful to accelerate the deployment of autonomous driving systems under large scenarios. For better visualization, the graph structure of road curbs are widened but its width is actually one-pixel.

1.2 Evaluation results

Please note that the evaluation results of ablation studies and comparative experiments in Tab. 1 and Tab. 2 of the original paper are trade-off results. We should consider Precision, Recall, F1-score and SCM at the same time. Thus, directly comparing a single evaluation metric is not appropriate. The evaluation curves in Fig. 3 of the original paper should be a better option for comprehensive performance comparison. In Fig. 3, our proposed CP-loss outperforms all other loss functions.

2 The Design of CP-loss

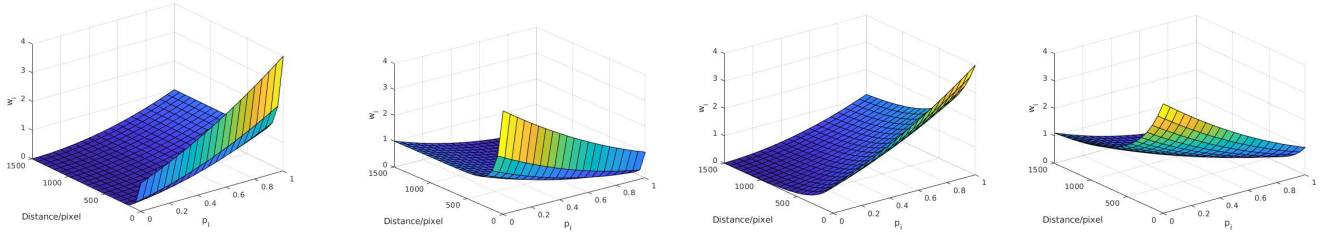
Our CP-loss consists of a weighted cross-entropy loss L_{CE} and a weighted dice loss L_{Dice}. The main issue here is the design of weights assigned to both loss functions. Before analyzing the loss function design, we first define some variables and symbols. For a pixel x_i of an input image, its ground-truth label is g_i and the predicted probability by UNet is p_i. The difference between g_i and p_i is denoted as Δ_i, and smaller Δ_i indicates more precise prediction. Besides, we define the shortest distance between x_i and corresponding wrong skeletons as d_i (i.e., $d_i = \mathit{min\_dis}(p_i, Skel_f)$). Smaller d_i means x_i is closer to the wrong skeleton and should receive more attention.

2.1 Weighted L_{CE}

The final adopted L_{CE} combines the distance function with Focal loss by addition and is shown in the following equation:

$$
\begin{aligned}
L_{CE} &= \sum_i [-u_i g_i log(p_i) - v_i(1-g_i)log(1-x_i)] \\
u_i &= [1 + exp[-\frac{\mathit{min\_dis}(x_i, Skel_{fG})}{\sigma}] - p_i]^2 \\
v_i &= [exp[-\frac{\mathit{min\_dis}(x_i, Skel_{fG} + Skel_{fP})}{\sigma}] + p_i]^2
\end{aligned}
\tag{1}
$$

where σ is a hyper-parameter to control the derivative and $\mathit{min\_dis}(x_i, Skel)$ calculates the shortest distance between a pixel p and a skeleton $Skel$. Since the input image is 1000×1000-sized, the range of the function $\mathit{min\_dis}(x_i, Skel)$ is $[0, \lceil(1000\sqrt{2})\rceil = 1415)$. Then the range of the distance function $exp[-\frac{\mathit{min\_dis}(x_i, Skel)}{\sigma}]$ is $(exp[-\frac{1415}{\sigma}], 1]$. By tuning parameters in the preliminary experiments, we finally set σ as 100 in our evaluation. Taking p_i and d_i as two variables, we obtain the distribution of the weight, which is visualized in Fig. 2.



(a) $\sigma = 100, g_i = 0$ (b) $\sigma = 100, g_i = 1$ (c) $\sigma = 500, g_i = 0$ (d) $\sigma = 500, g_i = 1$

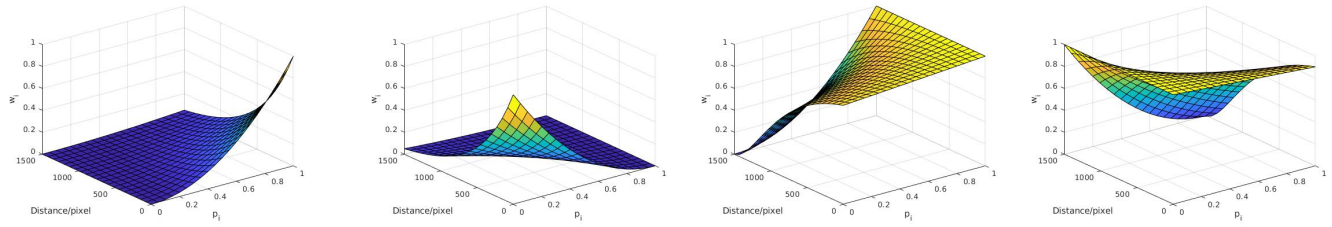
Figure 2: The weights distribution of CP-loss. (a) and (b) have $\sigma = 100$ so they have more steep mesh while (c) and (d) have $\sigma = 500$. (a) and (c) are for background pixels ($g_i = 0$). (b) and (d) are for foreground pixels ($g_i = 1$). σ is finally set as 100 in our evaluation.

Based on Fig. 2, we list three advantages of the adopted weight coefficients:

1. It can focus on harder pixels (Δ_i is large) and pixels closer to the wrong skeletons (d_i is large).
2. For pixels with low Δ_i, if they have small d_i, the weights are still relatively high. In this way, the pixels are further pushed to the label value until they are exactly the same (i.e., Δ_i is pushed to 0). This is important for foreground pixels and background pixels near $Skel_G$ since it greatly reduces uncertain pixels.

3. For pixels with large d_i, if they also have large Δ_i, the weight remains high. This property is important for background pixels because it can punish false positive predictions no matter how far they are from the skeletons. Without this property, the prediction tends to be noisy and have more false-positive predictions.

There are also some other methods to design the weight coefficients: (1) the naive multiplication method and (2) the positive-exponential multiplication method. Their weights are visualized in Fig. 3.



(a) *Naive*, $g_i = 0$ (b) *Naive*, $g_i = 1$ (c) *Pos-exp*, $g_i = 0$ (d) *Pos-exp*, $g_i = 1$

Figure 3: The weights distribution of other methods. For better visualization, σ is set to 500. The naive multiplication method (*Naive*) does not meet the second and third properties in the above analysis. It leads to very unstable training, and the result has much more false predictions. For the positive-multiplication method (*Pos-exp*), the gradient of pixels closed to the skeleton disappears and the network makes less accurate predictions.

The naive multiplication method refers to simply multiplying the distance function and Focal loss. The weight coefficients are:

$$\begin{aligned} u_i &= exp[-\frac{min\_dis(x_i, Skel_{fG})}{\sigma}] \cdot (1-p_i)^2 \\ v_i &= exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP})}{\sigma}] \cdot p_i^2 \end{aligned} \tag{2}$$

The major problem of this method is that if Δ_i is small or d_i is large, the weight becomes zero. So this method only focuses on false-predicted and nearby pixels, but ignores all other cases. Therefore this method always leads to much more false predictions, including false-positive and false-negative.

The positive-exponential method changes the exponential element from $exp(-d_i)$ to $exp(d_i)$. And the equations are:

$$\begin{aligned} u_i &= [1 - exp[\frac{min\_dis(x_i, Skel_{fG}) - \mu}{\sigma}] \cdot p_i]^2 \\ v_i &= [1 - exp[\frac{min\_dis(x_i, Skel_{fG}) - \mu}{\sigma}] \cdot (1-p_i)]^2 \end{aligned} \tag{3}$$

where μ is a hyper-parameter which is equal to $\lceil max(d_i) \rceil = 1415$. This method has a critical problem: the exponential element $exp[\frac{min\_dis(x_i, Skel) - \mu}{\sigma}]$ multiplied with p_i becomes very small if d_i is small. Then the gradient of the loss function on x_i disappears. This issue mainly affects the foreground pixels, and the network trained by this method makes predictions with much more false-negative predictions.

With the aforementioned analysis and visualization, we clarify the correctness of the design of weighted L_{CE} in CP-loss.

2.2 Weighted L_{Dice}

The final adopted weighted L_{Dice} also combines the distance function and Focal loss by addition, and the equation of it is:

$$\begin{aligned} L_{Dice} &= 1 - 2\frac{\sum_i \beta_i p_i g_i + smooth}{\sum_i (\beta_i p_i)^2 + \sum_i g_i^2 + smooth} \\ \beta_i &= \xi[1 + exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP})}{\sigma}] - \eta p_i] \end{aligned} \tag{4}$$

where *smooth* is a small number to prevent zero denominator. Compared with L_{CE}, L_{Dice} here has more coefficients (i.e., ξ and η), which can make the gradient of L_{Dice} as what we expect. Before discussing more about the extra coefficients, some differences between L_{CE} and L_{Dice} should be noted. Because L_{CE} is a pixel level loss function, the weight on every pixel can directly affects the gradient of L_{CE}, thus we can simply analyze the weight distribution. But for the image-level L_{Dice}, the gradient of a single pixel x_i is affected by the whole image (i.e., the loss calculation is not independent), so we need to do more calculation and talk about the gradient of each pixel.

Suppose the gradient of pixel x_i is $\bigtriangledown_i$, and for convenience, we replace the distance exponential element with w_i (i.e., $w_i = exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP})}{\sigma}]$. Also, we set the *smooth* term as 0 during the analysis. Inspired by the analysis method used in [1], we only consider a single sample x_i, then we have the dice loss l:

$$\begin{aligned} l &= 1 - 2\frac{\beta_i p_i g_i}{(\beta_i p_i)^2 + g_i^2} \\ \bigtriangledown_i &= \frac{\partial l}{\partial p_i} \\ &= \frac{\partial l}{\partial \beta_i p_i}\frac{\partial \beta_i p_i}{\partial p_i} \\ &= -2\frac{g_i^2 - (\beta_i p_i)^2}{[(\beta_i p_i)^2 + g_i^2]^2} \cdot \xi(1 + w_i - 2\eta p_i) g_i \end{aligned} \tag{5}$$

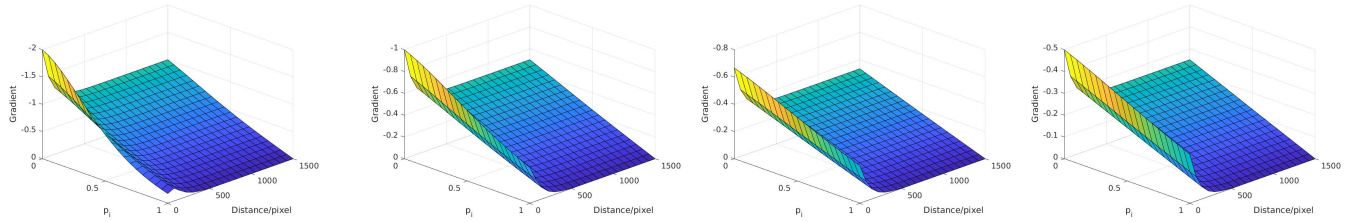
Apparently, for background pixels ($g_i = 0$) the gradient becomes zero. In our formal experiment, the gradient of background pixels is preserved by the *smooth* term, which is ignored for analysis convenience. So we only consider the foreground pixels ($g_i = 1$) and Eq. 5 becomes:

$$\begin{aligned} \bigtriangledown_i &= -2\frac{1 - (\beta_i p_i)^2}{[(\beta_i p_i)^2 + 1]^2} \cdot \xi(1 + w_i - 2\eta p_i) \\ &= -2\frac{1 - [\xi(1 + w_i - \eta p_i)]^2}{[(\beta_i p_i)^2 + 1]^2} \cdot \xi(1 + w_i - 2\eta p_i) \end{aligned} \tag{6}$$

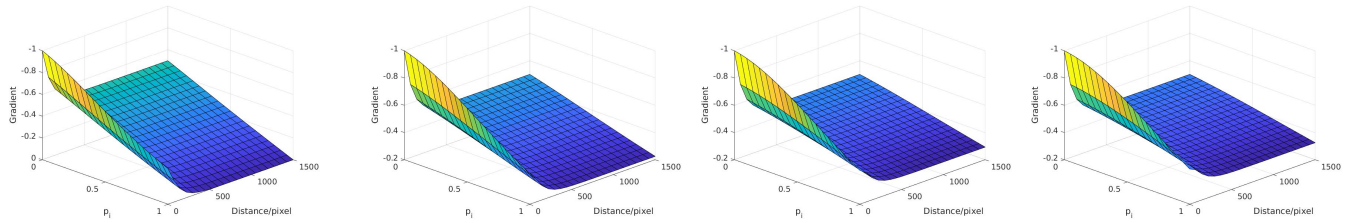
For $\bigtriangledown_i$, we want to keep it non-positive for all p_i and d_i. Because if $\bigtriangledown_i$ has positive values, it means there are non-boundary solutions to $\bigtriangledown_i = 0$ (i.e., there exist $p_i \in (0, 1)$ that makes $\bigtriangledown_i = 0$ with some d_i). Then the network would be trapped at sub-optimal state and Δ_i cannot be pushed to zero as much as possible, which causes more wrong predictions. To keep $\bigtriangledown_i$ non-positive, we have:

$$\begin{aligned} \xi(1 + w_i - 2\eta p_i) &\geqslant 0 \\ 1 - [\xi(1 + w_i - \eta p_i)]^2 &\geqslant 0 \\ s.t. \xi \geqslant 0 \text{ and } \eta &\geqslant 0, \forall p_i, w_i \end{aligned} \tag{7}$$

By solving the inequality, we have $0 \leqslant \xi \leqslant \frac{1}{2}$ and $0 \leqslant \eta \leqslant \frac{1}{2}$. The visualization of the gradient with different parameter choices is shown in Fig. 4. After trying multiple combination of parameters, we have



(a) $\xi = \frac{1}{2}, \eta = \frac{1}{2}$ (b) $\xi = \frac{1}{4}, \eta = \frac{1}{2}$ (c) $\xi = \frac{1}{6}, \eta = \frac{1}{2}$ (d) $\xi = \frac{1}{8}, \eta = \frac{1}{2}$



(e) $\xi = \frac{1}{4}, \eta = \frac{1}{2}$ (f) $\xi = \frac{1}{4}, \eta = \frac{1}{4}$ (g) $\xi = \frac{1}{4}, \eta = \frac{1}{6}$ (h) $\xi = \frac{1}{4}, \eta = \frac{1}{8}$

Figure 4: The gradient distribution of weighted L_{Dice} on p_i and d_i. The first row shows results with different ξ and the second row shows the results of different η. For ξ, the gradient of $\xi = \frac{1}{2}$ is closed to 0 when $p_i = 1, d_i = 0$, which does not meet the second advantage mentioned in L_{CE} analysis. While $\xi = \frac{1}{6}$ and $\xi = \frac{1}{8}$ make the gradient too small in general, so we finally set ξ as $\frac{1}{4}$. For η, the smaller η is, the more flat the distribution will be, so the network cannot focus on disconnectivity very well. Thus η is set as $\frac{1}{2}$.

$\xi = \frac{1}{4}, \eta = \frac{1}{2}$, and we finally have the weights for L_{Dice}:

$$\beta_i = \frac{1}{4}[1 + exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP})}{\sigma}] - \frac{1}{2}p_i] \tag{8}$$

Similar to L_{CE}, the naive multiplication method and the positive-exponential multiplication method were also considered for the weights of L_{Dice}, but these two methods cannot produce satisfactory gradients for training.

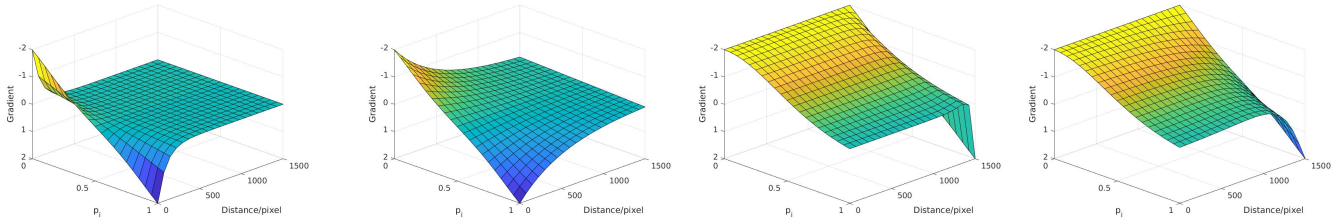
The equation of the weights of the naive multiplication method is:

$$\begin{aligned} \beta_i &= exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP})}{\sigma}](1 - p_i) = w_i \cdot (1 - p_i) \\ \bigtriangledown_i &= -2\frac{1 - [w_i p_i (1 - p_i)]^2}{[(\beta_i p_i)^2 + 1]^2} \cdot w_i(1 - 2p_i) \end{aligned} \tag{9}$$

Suppose $w_i = exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP}) - \mu}{\sigma}]$, then the equation of the weights of the positive-multiplication method is:

$$\begin{aligned} \beta_i &= (1 - exp[-\frac{min\_dis(x_i, Skel_{fG} + Skel_{fP}) - \mu}{\sigma}]p_i) = 1 - w_i p_i \\ \bigtriangledown_i &= -2\frac{1 - (\beta_i p_i)^2}{[(\beta_i p_i)^2 + 1]^2} \cdot (1 - 2w_i p_i) \end{aligned} \tag{10}$$

Both of these methods are not sufficient for our task. The visualization and explanation are shown in Fig. 5.



(a) *Naive*, $\sigma = 100$ (b) *Naive*, $\sigma = 500$ (c) *Pos-exp*, $\sigma = 100$ (d) *Pos-exp*, $\sigma = 500$

Figure 5: The gradient distribution of other methods. Both the naive multiplication method (*Naive*) and the positive-multiplication method (*Pos-exp*) produce gradient that crosses zero on non-boundary pixels. Thus in our experiments, the weights obtained from these methods usually cause inferior results.

References

[1] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, and J. Li, "Dice loss for data-imbalanced nlp tasks," 2020.