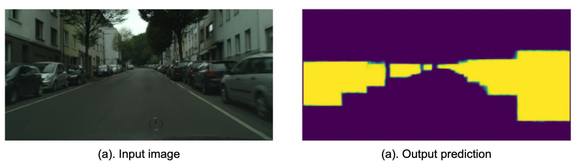
KU LEUVEN

WEAKLY-SUPERVISED SEMANTIC SEGMENTATION BY LEARNING LABEL UNCERTAINTY

Robby Neven, Davy Neven, Bert De Brabandere, Marc Proesmans and Toon Goedemé

TRAINING A PIXEL-PERFECT SEGMENTATION MODEL WITH BOUNDING-BOX LABELS

While achieving outstanding results, the downside of deep neural networks is that they are very data hungry. Especially segmentation models need highly curated datasets, which are very costly. A much cheaper way to label data are bounding boxes, since these only require 2 clicks per object. However, training a segmentation model on bounding box labels yields unexpected results (Figure 1).



(a). Input image (a). Output prediction

Figure 1. Output of a model trained with bounding-box labels.

Training segmentation models with bounding box labels is an active research fields. Most of the current works rely on box-to-mask proposal generators. However, these are generally not tuned to the dataset at hand. Other works try to first learn a better box-to-mask proposal generator on similar datasets, but for specific use-cases, these are mostly not available.

In this work, implemented a new loss function that leverages **label uncertainty** to perform **online bootstrapping** on the bounding-box labels during training, eliminating the need to have generic mask generators. Due to the improving labels during training, the model's segmentation performance increases overtime.

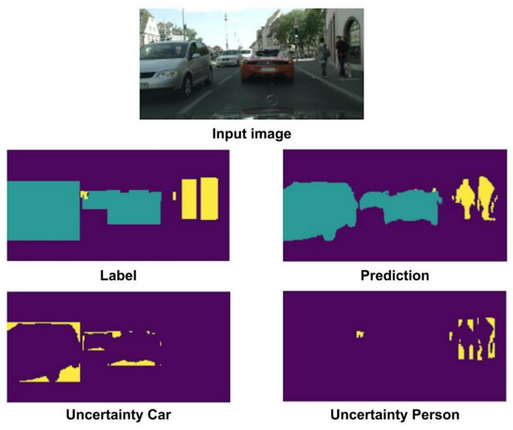


Figure 2. Model trained with our loss function. The model indicates it's label uncertainty which is used to bootstrap the label.

LEARNING LABEL UNCERTAINTY

A standard loss function cannot cope with bounding box labels right out of the box. Therefore, we introduced two new concepts to the a standard BCE loss for segmentation:

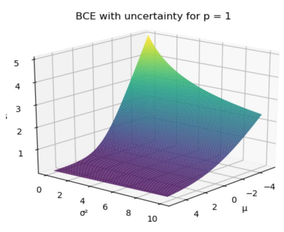
- **Learning aleatory uncertainty (label uncertainty)**
- **Performing online bootstrapping**

ALEATORY UNCERTAINTY

We incorporate aleatory uncertainty into the network by outputting a logit distribution $l \sim N(\mu, \sigma^2)$. By introducing aleatory uncertainty to the BCE loss, the network can learn to increase its uncertainty for pixels which target is highly unlikely e.g. background in bounding box labels. This extra parameter σ allows the loss to decrease, while the network outputs a label opposite to the target.

$$E[\hat{p}(l)] = sigmoid(\frac{\mu}{\sqrt{1+\pi\sigma^2/8}})$$

$$\mathcal{L}(\hat{p}, p) = p \log E[\hat{p}] + (1-p)\log(1-E[\hat{p}])$$



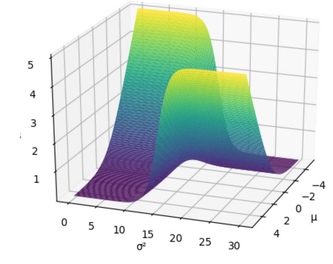
ONLINE BOOTSTRAPPING

While the aleatory uncertainty enables the model to contradict the target by increasing it's uncertainty, the gradients are still pointing towards the (erroneous) target. With bootstrapping, we can alter the targets and change the gradients towards the network's belief.

The bootstrapping loss is a weighed sum of the original and "flipped" target. The weight is based on the uncertainty learned from the uncertainty loss. For high uncertainties, the target (and loss) gets flipped, changing the gradients towards the network's output.

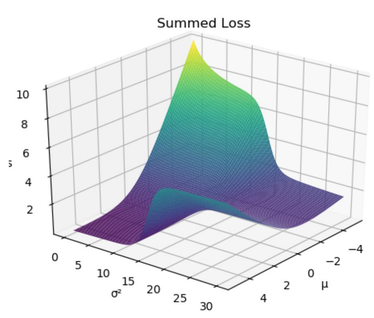
$$W = sigmoid\left(\frac{\tau - \sigma^2}{0.2}\right)$$

$$\mathcal{L} = W \cdot BCE(\mu, y) + (1-W) \cdot BCE(\mu, y^*)$$



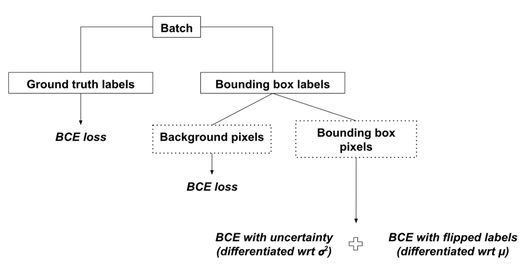
COMBINED LOSS FUNCTION

The combined loss function now has two local minima, enabling the network to lower the loss while contradicting the label. This can only be done by increasing it's uncertainty.



LEARNING THE UNCERTAINTY

The aleatory uncertainty learns the label uncertainty. However, if the model only sees bounding box labels, there is no uncertainty to learn. Only a small portion of pixel-perfect labels is needed to learn the correct uncertainty for erroneous pixels within the bounding box.



RESULTS

BINARY SEGMENTATION

We tested our method on the CityScapes dataset (car class), and compared the results to a model (U-Net) trained with a standard BCE loss. Table 1 shows the results. It is clear that our loss function enables the model to achieve near optimal IoU (trained with 100% pixel-perfect labels), when trained on a dataset consisting of a majority of bounding-box labels and only a small portion of pixel-perfect labels.

Table 1. Binary segmentation results.

Loss	Dataset		IoU (%)
	PP	BB	Car
Standard BCE	2780	/	86.9
	500	/	78.6
	/	2780	69.18
	500	2280	**69.43**
BCE Loss With Uncertainty	500	2280	**85.45**
	400	2380	85.32
	300	2480	85.42
	200	2580	84.37
	100	2680	81*

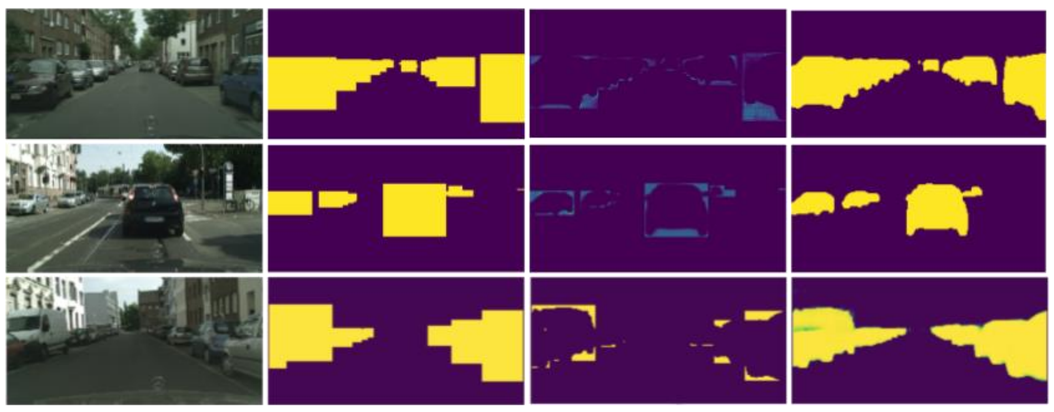


Figure 3. From left to right: input image, bounding-box target, label uncertainty and segmentation output.

MULTI-CLASS SEGMENTATION

We extended our binary loss to a multi-class setup by extending the single logit distribution characterised by μ and σ^2 to a distribution of the logit vector **l** characterised by a mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. The loss with aleatory uncertainty now is the cross-entropy loss of the expected probability.

$$\mathcal{L} = -p_y \log(E[\hat{p}_y])$$

With for $E[\hat{p}]$

$$E[\hat{\boldsymbol{p}}] = \int softmax(\boldsymbol{l}) \cdot P(\boldsymbol{l}) d\boldsymbol{l}$$

The bootstrapping loss consists now of a single term instead of a weighed sum: the target for the bootstrapping loss is changed when a certain class's variance exceeds a predetermined value.

$$y = \begin{cases} \text{argmax}(\boldsymbol{\mu}) & \text{if } \max(\boldsymbol{\sigma^2}) > \tau \\ y & \text{if } \max(\boldsymbol{\sigma^2}) < \tau \end{cases}$$

We tested the multi-class loss on the CityScapes cars and person instances. Again, we compared our results to a fully pixel-perfect dataset (Table 2).

Table 2. Multi-class segmentation results.

Loss	Dataset		IoU (%)			
	PP	*BB*	*Mean*	*BG*	*Car*	*Person*
CE	2780	/	79.80	98.21	85.82	55.68
	500	/	73.36	97.55	79.94	42.58
Ours	500	2280	77.75	98.01	82.05	53.2

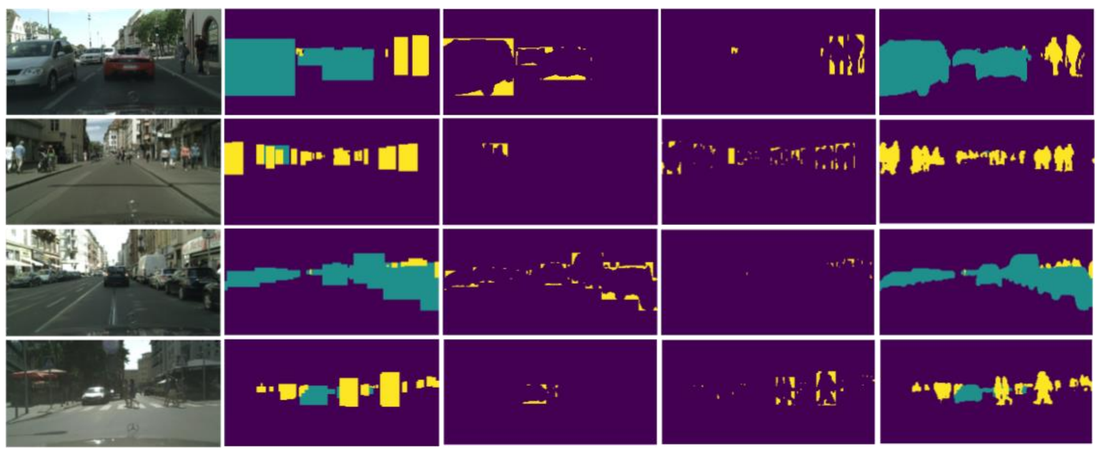


Figure 3. From left to right: input image, bounding-box target, car label uncertainty, person label uncertainty and segmentation output.

CONCLUSION

In this work, we proposed a new loss function for training a segmentation model with bounding box labels without the need of box-to-mask proposal generators. Our method requires only a small subset of pixel-perfect labels, which drastically reduces the annotation cost. However, due to the sampling nature of the MCI in our loss function, extending our method to more than 3 classes is still work in progress.