LiteEdge: Lightweight Semantic Edge Detection Network

Hao Wang, Hasan Mohamed, Zuowen Wang, Bodo Rueckauer*, and Shih-Chii Liu
Institute of Neuroinformatics, University of Zurich and ETH Zurich
Winterthurerstrasse 190
CH-8057 Zurich, Switzerland
shih@ini.uzh.ch

Abstract

Scene parsing is a critical component for understanding complex scenes in applications such as autonomous driving. Semantic segmentation networks are typically reported for scene parsing but semantic edge networks have also become of interest because of the sparseness of the segmented maps. This work presents an end-to-end trained lightweight deep semantic edge detection architecture called LiteEdge suitable for edge deployment. By utilizing hierarchical supervision and a new weighted multi-label loss function to balance different edge classes during training, LiteEdge predicts with high accuracy category-wise binary edges. Our LiteEdge network with only $\approx$ 3M parameters, has a semantic edge prediction accuracy of 52.9% mean maximum F (MF) score on the Cityscapes dataset. This accuracy was evaluated on the network trained to produce a low resolution edge map. The network can be quantized to 6-bit weights and 8-bit activations and shows only a 2% drop in the mean MF score. This quantization leads to a memory footprint savings of 6X for an edge device.

1. Introduction

Scene parsing and semantic segmentation [2] are fundamental problems in computer vision research. They can provide information about major landmarks in the surrounding environment, as well as objects of interest in the foreground. The resulting segmented output can be utilized for downstream applications, such as navigation [11] and indoor autonomous mobile robotics [3, 22]. Meanwhile, the classical edge detection task has been shown beneficial for solving many computer vision tasks such as 3d reconstruction [29], 3d shape recovery [20], medical image processing [24], as well as semantic segmentation [4, 6].

Semantic edge detection combines both edge detection and semantic classification associating edge pixels with one or more object categories [13, 33]. For every pixel on an image the task solves whether the pixel lies on an edge and which class(es) it belongs to. It is modeled as a multi-label learning problem [33] since one pixel can belong to multiple classes, e.g. boundaries distinguishing vehicles on a road have the semantic edge class 'vehicle' and 'road' at the same time. Moreover, for each class, the output map will mask out all pixels which do not fall on the outline of this class, resulting in a highly sparse binary output. With rich semantic information in addition to the basic edge location information, semantic edges can be directly used or easily extended to solve many tasks such as refined object detection [12], image-based geopositioning [26], panoptic segmentation [17] and vision restoration applications in brain-machine interface research [9, 27].

Previous work on semantic edge detection [1, 16, 33] use heavy backbone networks. Because many of these networks have a large memory footprint and require high computes, they cannot be deployed for real-time performance on an edge device. This work proposes a lightweight, well-performing end-to-end semantic edge detection network, LiteEdge, which is suitable for deployment on the edge. We implement two ways of keeping the network light but accurate as validated using the Citiscapes dataset. First, we start with the architecture of a state-of-the-art semantic segmentation network [10] as the backbone and further reduce the output image dimensions for lower computes. Second, to improve the accuracy, we incorporate an additional hierarchical supervision architecture to generate sparse edge segmentation maps for each class, as well as edge class weights in the loss computation. We show that our network while maintaining good prediction quality runs at reasonable inference speeds on edge devices. Moreover, by applying quantization aware training (QAT) [19], we are able to compress the model size by 6 times without much loss in accuracy.

*Current affiliation: Donders Centre for Cognition, Radboud University, Nijmegen, The Netherlands.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 899287.

The main contributions of this work are as follows:

- A novel end-to-end semantic edge detection network architecture named LiteEdge, which gives competitive prediction accuracy on Cityscapes and uses only ≈3 M parameters resulting in a high throughput of 112 frames per sec (FPS) on an Nvidia RTX 2080Ti GPU.
- A hierarchical supervision module using only binary edges that improves the semantic edge accuracy of LiteEdge by 12.0%.
- A new weighted multi-label loss function to address the class pixel imbalance problem. This loss takes into consideration the difference of segmentation pixel counts in different classes across the dataset, allowing for improved semantic edge learning.
- By adding one segmentation branch to LiteEdge, the new network (LiteEdgeSeg) outputs both semantic edge and segmentation results simultaneously while maintaining a similar inference speed to LiteEdge.

2. Related works

Edge detection Traditional edge detection algorithms such as Canny [5] use convolutional filters, which generate category-free edges. In addition, a wide variety of deep neural networks are driven towards the edge detection area, such as Deep contour [30] and holistically-nested edge detection (HED) [32]. However, these methods produce low-level edges unlike semantic edge detection which is related to both geometric edges and semantic understanding.

Semantic edge detection The idea of semantic edge detection first comes up in [25]. In the work of [13], the Semantic Boundaries Dataset (SBD) is introduced and an inverse detector is proposed. The inverse detector can detect category-aware semantic edges because it has information from both bottom-up edge and top-down detector. Many semantic segmentation works can be loosely regarded as the semantic edge detection task, since employing an edge detector with the information from segmentation results can produce semantic edges. For example, the "High-for-Low" approach (HFL) [4] employs VGG to extract binary semantic edges with the features from semantic segmentation networks to obtain category labels. However, these methods are typically not end-to-end and need additional postprocessing.

The CASENet [33] architecture is trained end-to-end using ResNet-101 [14] as the backbone. It combines both low- and high-level features with a designed multi-label loss function to produce semantic edges. In the work of STEAL [1], the authors propose a new thinning layer and loss, which can be added on top of any end-to-end edge detector. The DFF model [16] is the first work to to use an adaptive weight fusion module to dynamically generate location-specific fusion weights that are conditioned on the image content. These location-specific weights are applied to fuse both the high-level and low-level response maps in order to predict the semantic edges with higher accuracy.

A few studies have also combined the task of semantic segmentation with semantic edge detection, such as JSENet [18] which simultaneously predicts the semantic segmentation point mask and the semantic edge point map. However, all the architectures mentioned are relatively heavy, either because of the large backbone model or the addition of multiple modules for increasing accuracy. Previous end-to-end trained semantic edge detection models reported inference speeds that are below 10 FPS on a GPU. We believe that LiteEdge is the first model that runs above 10 FPS and reaches more than 100 FPS on a Nvidia RTX 2080Ti GPU.

3. Network

Problem formulation. The main goal of semantic edge detection is to compute the semantic edge maps for each category. Formally, given an input image $x \in \mathbb{R}^{H \times W \times Ch}$ with C defined semantic categories, the model predicts C edge maps $\bar{Y} = (\bar{y}_1, \bar{y}_2, ..., \bar{y}_C)$. The model is trained on input images and ground truth semantic edge images, each represented as $Y = (y_1, y_2, ..., y_C)$. y_c and $\bar{y}_c$ are binary maps with dimensionality $\{0, 1\}^{H \times W}$. Pixels in y_c or $\bar{y}_c$ with value equals to 1 belong to the c-th category and those equal to 0 are not.

3.1. Basic architecture

The basic architecture is a modified form of the semantic segmentation network, LiteSeg [10]. The details of LiteSeg are shown in the light green box of Figure 1. It consists of three main parts, the backbone network, the deeper atrous Spatial Pyramid Pooling (DASPP) module and a decoder module. The input and output of the DASPP module are concatenated by using a short connection, and the output of the 3rd block of the encoder is connected to the decoder by using a long connection.

The backbone network. The task accuracy and computational efficiency of this network are highly dependent on the chosen backbone network which can be any type of convolution network, such as VGG16 [31], MobileNet [15, 28] and ResNet [14]. For real-time segmentation, MobileNet v2 [28] can provide a good trade-off between accuracy and computational efficiency in this architecture.

The DASPP module [10] (Figure 2) is based on the Atrous Spatial Pyramid Pooling (ASPP) module in DeepLabV3 [7]. It comprises a set of convolution blocks with increasing dilation rates, which helps the network to capture object features as well as useful image context at multiple scales. Compared with the ASPP module, there

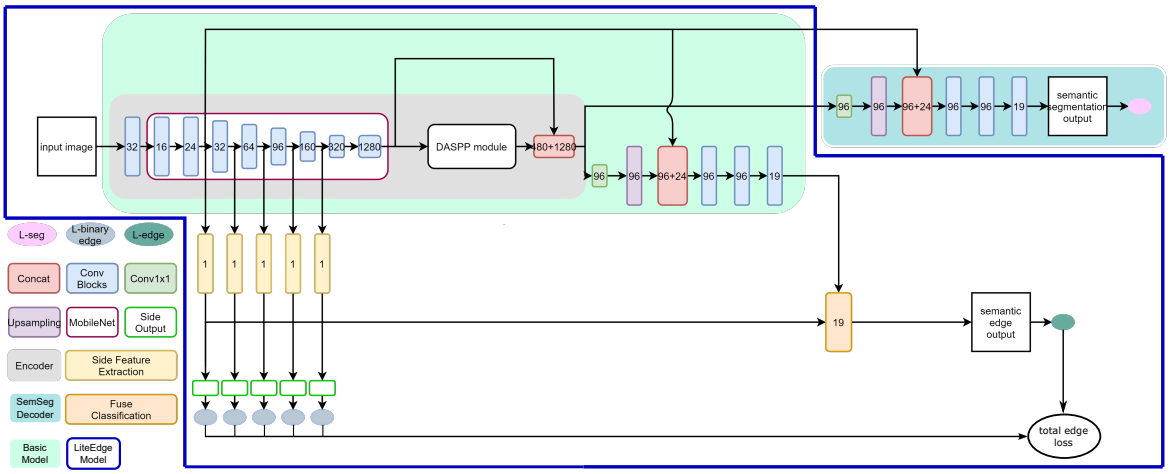


Figure 1: LiteEdge model (in blue outlined polygon) and LiteEdgeSeg. The base model is shown in the box with the light green background. The encoder is shown in the box with the gray background. By adding the semantic segmentation decoder (box with blue background) in upper right of figure, we get LiteEdgeSeg.

is an additional standard 3 x 3 separable convolution after the 3 x 3 atrous separable convolution to refine the features in the DASPP module, and a short residual connection to fuse the features from the input and output of the DASPP module. The number of filters for the convolution layers in ASPP is reduced from 255 to 96 to further improve the efficiency.

The decoder module is modified from Deeplabv3+ [7], it is a simple architecture that only contains four convolutions blocks, one upsample step and a concatenation step. As shown in Figure 1, the concatenation step (long residual connection) combines the information from the 3rd block in the backbone network (MobileNet v2) and the feature map after the upsample step to further merge low-level and high-level features. Fusion of low-level features which often include edges or color blobs from bottom layers and high-level features which often capture semantic information from top layers is helpful for semantic edge prediction.

3.2. LiteEdge architecture

The LiteEdge model shown in Figure 1 uses the basic architecture described in Sec. 3.1 as the backbone network. It incorporates modified versions of the feature extraction and the hierarchical supervision modules of JSENet [18]. These modules are added to the side outputs of the backbone network. We also include a new fuse classification module on the output of the basic architecture.

Hierarchical supervision We use hierarchical supervision to regularize the side feature extraction during learning of the binary edges. This supervision is useful because firstly, the context information learned in the bottom layers

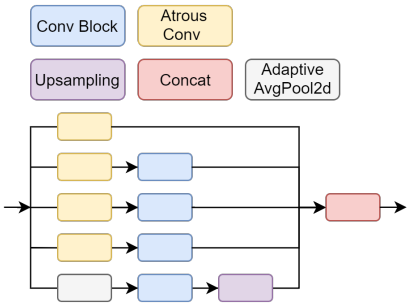
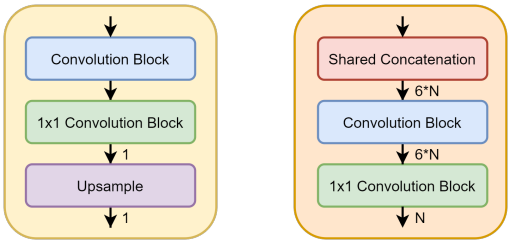


Figure 2: Basic architecture of DASPP module. For the first four branches, we use the atrous convolution with dilation rates of 1, 3, 6 and 9 respectively.

plays a vital role in semantic classification. They help augment top classifications, therefore, merging the information from side outputs can improve the MF score of edge prediction. Secondly, the receptive field of the deeper layers is limited, and the network can lose the detailed pixel-wise information at this stage. Thus, it is beneficial to give a supervision signal about semantic edges in the early stage of the network. Unlike JSENet [18] where the first three side outputs are supervised by binary edges and the last two side outputs are supervised by the semantic segmentation map, our LiteEdge model uses side feature extraction modules that are all supervised by binary edge labels.

Side feature extraction module Motivated by JSENet [18], the features from bottom layers help to improve the accuracy of classification and segmentation task but need



(a) (b)

Figure 3: Architecture of (a) the side output feature extraction module and (b) the fuse classification module.

to be processed to incorporate with the features from the main backbone. The architecture of the side output feature extraction module is shown in Figure 3a, these modules are different from those in JSENet, they include an additional 3x3 convolution block and the deconvolution is replaced by the upsampling block.

Fuse classification module In order to fuse the feature map from side feature extraction and the backbone branch, we add a fuse classification module to the decoder of LiteEdge. This module (see Figure 3b) has a shared concatenation layer [33] and two convolution blocks. The shared concatenation layer fuses the feature maps from side outputs and the features from the main backbone. The last layer of the fuse classification module is a 1x1 group convolution.

3.3. LiteEdgeSeg architecture

LiteEdge can be extended to a new model, LiteEdgeSeg, which has two branches with a shared encoder. The first branch predicts the semantic segmentation map, and the second branch outputs the semantic edge maps. The structure is shown in Figure 1. Besides the fuse classification module, the decoder for the semantic segmentation branch has the same structure as the decoder for the semantic edge detection branch. Compared to LiteEdge, LiteEdgeSeg has only an additional 6% parameters.

3.4. Weighted multi-label loss function

Inspired by the class weighting scheme in [23], we propose a multi-label semantic edge loss term (ℓ_{SE}).

First, we define w_{pos} as the ratio of non-edge pixels to the edge pixels. The value of w_{pos} is calculated per image. Second, we introduce w_{cls} as the weighted class frequency for each class. Algorithm 1 shows how to calculate w_{cls}. The proposed loss term ℓ_{SE} is a modified version of the multi-label loss in [33]. Our proposed loss integrates the class weights $w_{cls} \in \mathbb{R}^C$ into the loss to overcome the class pixel count imbalance problem. Given w_{cls} as the class weights, Algorithm 2 shows how to calculate w_{pos} and the batched semantic edge loss $\ell_{SE\text{-}batch}$.

Algorithm 1 Calculate edge class weights w_{cls}

Input: Training set of input image x, semantic segmentation label s

for class c in $\{1, \ldots, C\}$ **do**
  Define total pixel count for class c as $I_c = 0$
  for s of each image x in the training set **do**
    count pixels in s that belong to class c as I
    $I_c = I_c + I$
  end for
end for
for class c in $\{1, \ldots, C\}$ **do**
  calculate the probability: $p_c = I_c / \sum_{c=1}^{C} I_c$
  calculate the weight: $w_c = 1/\log(1.02 + p_c)$
end for
find the weight of median frequency class w_{median}
standardize the final w_{cls}^c for each class c: $w_{cls}^c = w_c / w_{median}$
return w_{cls}

Algorithm 2 Calculate batched multi-label semantic edge loss $\ell_{SE\text{-}batch}$

Input: Batch with input image x, edge label $Y = (y_1, \ldots, y_C)$

Define semantic edge loss as ℓ_{SE} and set $\ell_{SE\text{-}batch} = 0$
for each $\big(x, Y = (y_1, \ldots, y_C)\big)$ pair in the batch **do**
  $w_{pos} = \frac{\#\text{non-edge pixels in } x}{\#\text{edge pixels in } x}$
  for class c in $\{1, \ldots, C\}$ **do**
    $\ell_{SE}^c = -y_c \cdot \log\big(\sigma(\hat{y}_c)\big) \cdot w_{pos} + (y_c - 1) \cdot \log\big(1 - \sigma(\hat{y}_c)\big) \cdot (1 - w_{pos})$
    $\ell_{SE}^c = \ell_{SE}^c \cdot w_{cls}^c$
  end for
  $\ell_{SE} = \text{mean}(\ell_{SE})$
  $\ell_{SE\text{-}batch} = \ell_{SE\text{-}batch} + \ell_{SE}$
end for
return $\ell_{SE\text{-}batch}$

4. Experiments

We compare our LiteEdge model with other state-of-the-art semantic edge detection models including CASENet [33] and DFF [16], on the Citiscapes dataset. LiteEdge was trained with a 512x1024 input size instead of the original input size because of the lengthy simulation time. The corresponding output size is 128x256 to reduce further needed computes. Both CASENet and DFF were trained on a 1024x2048 input image size. The resulting output of 1024x2048 was downsampled by 4x on both axes for

Model name	mean MF	# params	input size	output size	FLOP	FPS
real-time models						
LiteSeg [10] + Canny	44.4	3.10M	512x1024	512x1024	4.459G	58
LiteEdge (ours)	52.9 $\pm$ 0.32	3.16M	512x1024	128x256	5.022G	**112**
non real-time models						
CASENet [33]	58.3	43.53M	1024x2048	1024x2048	1860.007G	1.56/ 4.42*
DFF [16]	63.4	44.28M	1024x2048	1024x2048	5003.663G	1.43/ 4.28*

Table 1: Results of semantic edge detection models on the Cityscapes evaluation dataset. All MF scores are measured in percent. Mean MF scores for all models were obtained by downsampling or upsampling the network output to 256x512. The ground truth edge maps in the validation set are downsampled to 256x512 for the mean MF score calculation. *For the FPS of CASENet and DFF, we show numbers inferencing on input size 1024x2048/ 512x1024 for comparison with LiteEdge.

the mean MF score calculation. The groundtruth on the 256x512 resolution and the pre-trained CASENet are obtained from this repository[1] and DFF is from this repository[2]. The output of LiteEdge was bilinearly upsampled to 256x512. We also include results from LiteSeg [10] with the Canny edge detector; and the LiteEdgeSeg model which outputs both the semantic edges and segmentation maps. Two other studies are reported: An ablation study of LiteEdge to show the importance of each module; and a model compression study where we quantize this model to reduce the model size. Results reported from LiteEdge and LiteEdgeSeg are averaged over five runs.

4.1. Dataset

We use the Cityscapes [8] dataset which comprises complex and diverse stereo video sequences recorded from 50 different cities in Europe. There are 5000 images that have high-quality pixel-level annotations and 20000 images only have coarse annotations. Out of the 5000 images, 2975 are in the training set, 500 in the validation set and 1525 in the test set. For evaluation, we use the validation dataset.

4.2. Evaluation protocol

The maximum F (MF) measure at the optimal dataset scale (ODS) [13] is the metric used for the evaluation of semantic edges. For each point on the precision-recall curve, the F-score is calculated as $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$, and the MF is the maximum value of these F-scores. The ODS metric uses a fixed threshold value that gives the maximum F-score on the validation dataset. MF is computed for each class, and the mean MF is the average value of the MFs across all classes. The matching pixel distance tolerance is the maximum margin allowed for correct matches of edges to ground truth during evaluation. The distance tolerance is often measured as the proportion to the length of the image diagonal, which we choose as 0.0035 in the experiments as in [16].

[1]https://github.com/anirudh-chakravarthy/CASENet

[2]https://github.com/Lavender105/DFF

4.3. Training setup

We used Stochastic Gradient Descent (SGD) with Nesterov, a momentum value of 0.9 and weight decay of 0.0005. We train the network for 100 epochs using a batch size of 4. The multiple learning rate policy is used. The initial learning rate, lr_1, is set to 0.01, the learning rate changes every 5 epochs. The current epoch's learning rate is computed by $lr_1 * (1 - \frac{e_i}{e_m})^{0.9}$, where e_i indicates the current epoch number and e_m indicates the maximum number of epochs. For data augmentation, we use random horizontal flip with the probability of 0.5, random scaling by a factor between 0.2 to 0.8 and random rotation (-5^o to 5^o) of the image. All the experiments were performed on a single Nvidia RTX 2080Ti GPU and one Intel Xeon(R) W-2195 CPU.

4.4. Results

Table 1 compares the mean MF score, network parameter size, and runtime metrics, i.e., floating point operations (FLOP), frames per sec (FPS), of our LiteEdge model with current state-of-the-art models.

For LiteSeg[3] [10] and Canny, we took the evaluation model with input of 512x1024, performed Canny on the output, before downsampling the output to 256x512. Our LiteEdge model uses at least 10X fewer parameters than DFF and CASENet. The inference time of our LiteEdge model is 22X faster than CASENet for the same input resolution. The mean MF score of LiteEdge falls behind the heavy backbone CASENet by around 5% while enabling the capability of deployment on edge devices. Directly adding a Canny edge detector on a semantic segmentation network will dramatically decrease the processing speed. Note that Canny edge detection runs on the CPU.

Figure 4 shows the MF scores of LiteEdge on different classes. In many small object classes where it is difficult to extract intact and clean edges, LiteEdge has the best MF (ODS) score. The comparison to other models is further described in Section 4.5.

[3]https://github.com/tahaemara/LiteSeg

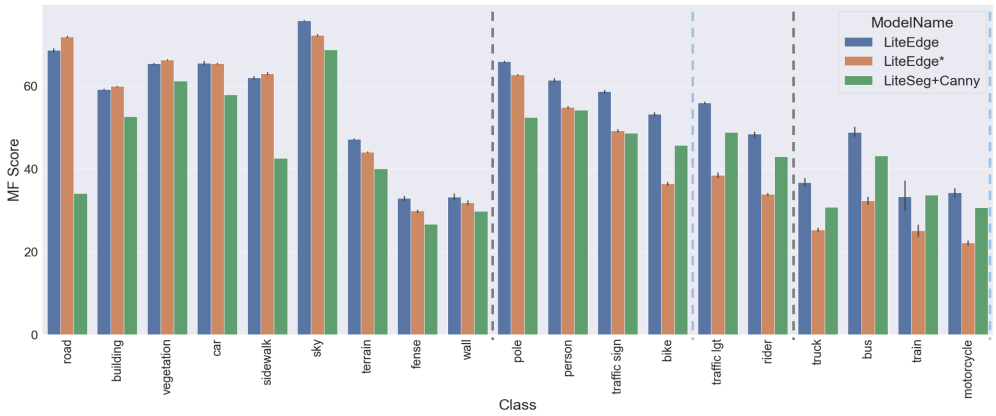


Figure 4: Class-wise MF scores of semantic edge detection models. All MF scores are measured in percent. For evaluation, the input size is 512×1024. The predicted edge maps and ground truth edge maps are interpolated to 256×512. LiteEdge* indicates LiteEdge trained without class weight. Classes between gray dashed lines are small object classes; classes between two blue dashed lines are low-frequency occurring classes ($\leq 0.5\%$).

Figure 5 compares the predictions of the different networks on 5 example scenes from Cityscapes. The predictions of LiteSeg with a Canny detector show that the network cannot distinguish individual objects of the same class, for example, the edges of multiple cars in the middle column form the edge of one connected object.

LiteEdgeSeg. By adding one extra branch, we create a new model which can produce the semantic edges and the semantic segmentation map simultaneously. The pixel accuracy (PA), mean intersection of union (MIoU) and frequency weighted intersection of union (FWIoU) are three common evaluation metrics of semantic segmentation task. PA reports the percentage of pixels in an image that are correctly classified; IoU measures the percentage overlap between the target mask and the prediction mask, and MIoU is the average value across all classes; FWIoU multiply the IoU with the frequency for each class and sum up overall classes. The PA, MIoU and FWIoU of LiteEdgeSeg's semantic segmentation results are 93.80%, 66.94% and 88.83%, which are close to the performance of the LiteSeg model. The mean MF of semantic edge evaluation for LiteEdgeSeg is 53.0%, which is as good as the LiteEdge model's performance (52.9%).

4.5. Ablation study

We performed an ablation study to determine the importance of the class weight and hierarchical supervision modules. The results are presented in Table 2. By using class weight labels (following the calculation method in Section 3.4) to train the network, we encourage the network to better predict edges on rare or small classes.

Figure 4 gives the MF score of each class. The scores on small objects (such as traffic light, person, rider and bike) and the classes that are rare in the training set (such as truck and bus) are improved, but influences the prediction on big objects (such as road and building). The overall MF score increases around 6.4% with the use of class weighted labels.

Class weight	Hierarchical supervision	Mean MF
×	×	34.5 ± 0.58
×	✓	46.5 ± 0.16
✓	✓	52.9 ± 0.32

Table 2: Ablation study of LiteEdge model.

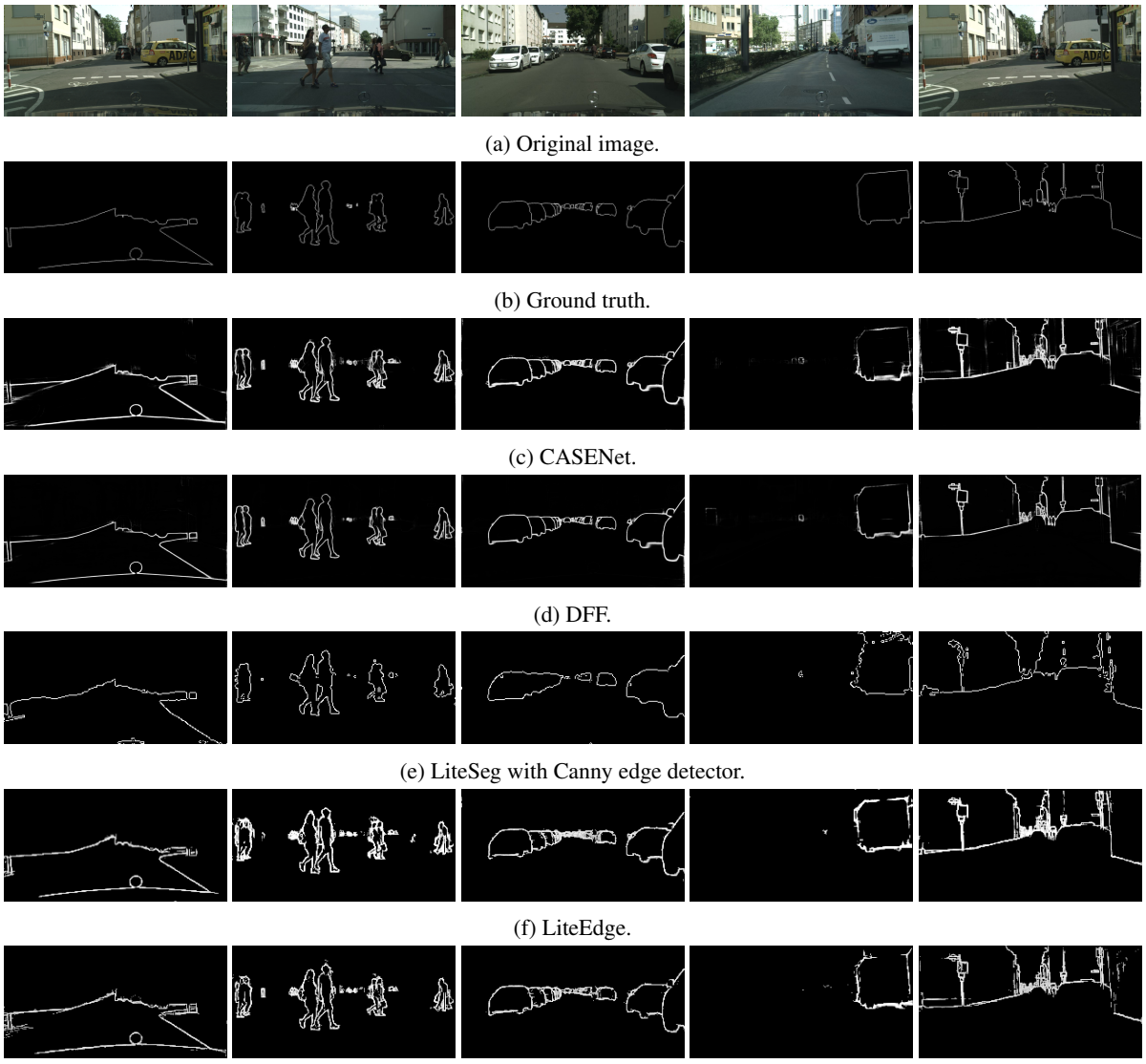
To determine the need for the hierarchical supervision on bottom side outputs, we directly concatenate the side outputs with the decoder without supervising them to learn binary edges. Comparing the results in the first two rows of Table 2, we see that the hierarchical supervision improves the mean MF score by 11%.

4.6. Model compression study

Quantization has become a significant method for optimizing deep-learning models so that they can accelerate inference when deployed on embedded systems with restricted memory footprint and computing resource. In this work, we use the Neural Network Intelligence toolkit[4]. To increase the network accuracy, we use the quantization aware training method (QAT) [19], where we started with the trained model and further refined the model with quantized parameters for 100 epochs.

The results in Table 3 show that the quantized model with 8-bit weights and activations only has a small drop of

[4]https://nni.readthedocs.io/en/stable/Overview.html



(a) Original image.

(b) Ground truth.

(c) CASENet.

(d) DFF.

(e) LiteSeg with Canny edge detector.

(f) LiteEdge.

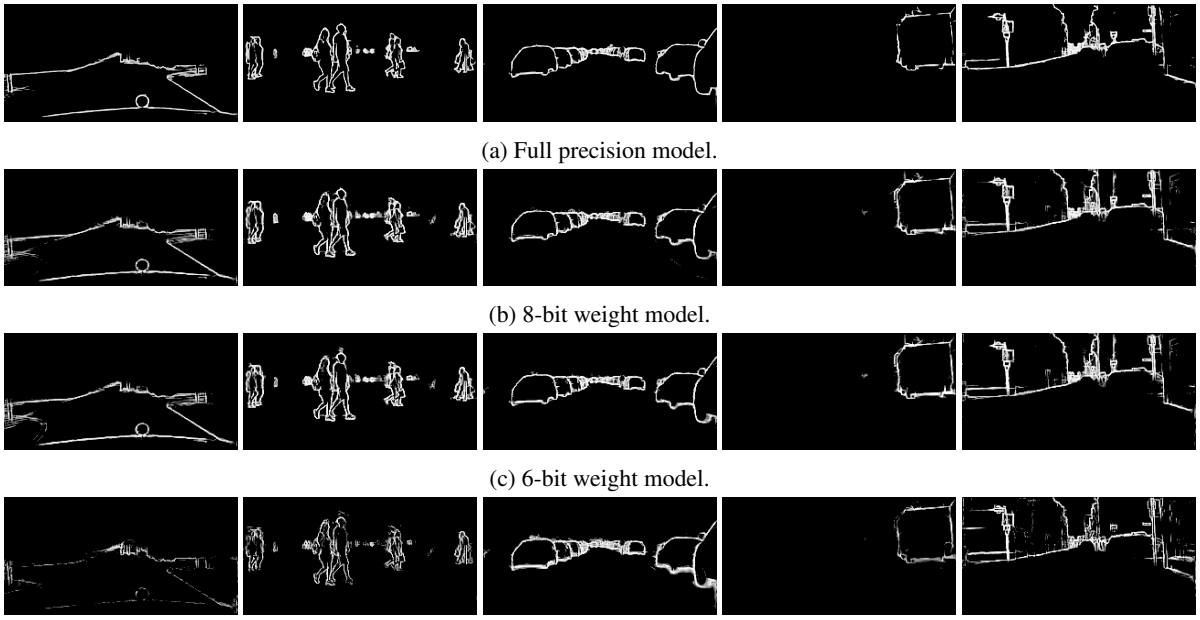
(g) LiteEdgeSeg.

Figure 5: Class-wise results of semantic edge detection networks. From top to bottom: original image, ground truth, CASENet, DFF, LiteSeg with Canny, LiteEdge (trained without class weights) and LiteEdgeSeg.

2.2% in the mean MF score. The model size of the quantized network is around 4X smaller than the full precision network. In addition, QAT brings weight sparsity, where 2.65% of the parameters are zero after the quantization. The results for the 6-bit and 4-bit weight quantized models are also presented in Table 3. Compared with the 8-bit weight quantized model, the 6-bit weight quantized model shows a drop of 0.2% in the mean MF score but achieves around 4 times weight sparsity. The 4-bit weight quantized network shows a larger drop in the mean MF score but has more than 38% zero parameters. In addition, its model size is 13X smaller than the full precision model.

Model	Mean MF	Model size (MB)	Zero parameters
Full precision	52.9 ± 0.32	12.647	131 (4.1e-05%)
8-bit weights	50.7 ± 0.94	3.078	84K (2.65%)
6-bit weights	50.9 ± 0.99	2.128	323K (10.23%)
4-bit weights	42.7 ± 0.77	0.983	1.2M (38.07%)

Table 3: Model quantization results on LiteEdge. Activations are quantized to 8-bits for all quantized models.



(a) Full precision model.

(b) 8-bit weight model.

(c) 6-bit weight model.

(d) 4-bit weight model.

Figure 6: Class-wise results of model quantization experiments on LiteEdge. From top to bottom: original image, ground truth, full precision model, 8-bit weight model, 6-bit weight model and 4-bit weight model.

4.7. Real-time edge performance

The proposed models are further deployed on an Nvidia Jetson Nano[5] to evaluate their runtime performance on an edge device. This device comes with a 128-core integrated Nvidia Maxwell GPU and a quad-core 64-bit ARM CPU. It has 5W and 10W power modes. TensorRT[6] 7.1.3 is used to generate optimized FP16 run-time engines for the models. The 10W power mode is activated before inference. The results are averaged over 100 runs. Table 4 shows the frame rates for the full precision models deployed on the Jetson Nano. Both LiteEdge and LiteEdgeSeg can achieve a frame rate of 15-18 FPS when using an image resolution of 256×512. By comparison, the inference frame rate of CASENet is 10X lower.

Model	# params	Input resolution	FPS
CASENet [33]	43.53M	512×1024	0.39
		256×512	1.5
LiteEdge	3.16M	512×1024	5
		256×512	18
LiteEdgeSeg	3.35M	512×1024	4
		256×512	15

Table 4: Prediction frame rate for proposed models on Jetson Nano. CASENet results are shown for comparison.

5. Conclusion

We present LiteEdge, an end-to-end lightweight semantic edge detection model suitable for edge deployment. It achieves a mean MF score of 52.9% on the Cityscapes validation set with a reduced input and output size to address the accuracy versus compute tradeoff. The model gives 22X and 10X higher frame rate compared to previous models on a desktop GPU and an edge device respectively. By adding the hierarchical supervision module and a new multi-class weight label loss, we could increase the mean MF of this network which has a lower output resolution. By adding one additional semantic segmentation branch, we extend LiteEdge to LiteEdgeSeg which outputs both the semantic edge and semantic segmentation maps. The 6-bit weight quantized LiteEdge model shows only a small drop of 2% in mean MF score and has a memory footprint savings of 6X. The added modules to LiteSeg [10] can also be applied towards other segmentation networks. Preliminary results show that when they are added to a recent reported segmentation network (FSFNet [21]), the mean MF score of the edge prediction increases by 1.2% compared to using the Canny edge detector on segmented maps.

[5]https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/

[6]https://developer.nvidia.com/tensorrt

References

[1] David Acuna, Amlan Kar, and Sanja Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11075–11083, 2019.

[2] Nadeem Atif, Manas Bhuyan, and Shaik Ahamed. A review on semantic segmentation from a modern perspective. In *2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–6, 2019.

[3] Jonathan C Balloch, Varun Agrawal, Irfan Essa, and Sonia Chernova. Unbiasing semantic segmentation for robot perception using synthetic data feature transfer. *arXiv preprint arXiv:1809.03676*, 2018.

[4] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 504–512, 2015.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8:679 – 698, 12 1986.

[6] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4545–4554, 2016.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 06 2016.

[9] Jaap de Ruyter van Steveninck, U. Güçlü, R. V. van Wezel, and M. V. van Gerven. End-to-end optimization of prosthetic vision. *bioRxiv*, 2020.

[10] Taha Emara, Hossam Munim, and Hazem Abbas. Liteseg: A novel lightweight convnet for semantic segmentation. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7, 12 2019.

[11] Di Feng, Christian Haase-Schuetz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[12] Vittorio Ferrari, Frederic Jurie, and Cordelia Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 87(3):284–303, 2010.

[13] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998, Nov 2011.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[16] Yuan Hu, Yunpeng Chen, Xiang Li, and Jiashi Feng. Dynamic feature fusion for semantic edge detection. *arXiv preprint arXiv:1902.09104*, 2019.

[17] Yuan Hu, Yingtian Zou, and Jiashi Feng. Panoptic edge detection. *arXiv preprint arXiv:1906.00590*, 2019.

[18] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In *Proceedings of the 16th European Conference on Computer Vision (ECCV), Part XX 16*, pages 222–239. Springer, 2020.

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[20] Kevin Karsch, Zicheng Liao, Jason Rock, Jonathan T Barron, and Derek Hoiem. Boundary cues for 3d object shape recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2163–2170, 2013.

[21] Minjong Kim, Byungjae Park, and Suyoung Chi. Accelerator-aware fast spatial feature network for real-time semantic segmentation. *IEEE Access*, 8:226524–226537, 2020.

[22] Wonsuk Kim and Junhee Seok. Indoor semantic segmentation for robot navigating on mobile. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 22–25, 2018.

[23] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[24] Reza Pourreza, Ying Zhuge, Holly Ning, and Robert Miller. *Brain Tumor Segmentation in MRI Scans Using Deeply-Supervised Neural Networks*, pages 320–331. 02 2018.

[25] Mukta Prasad, Andrew Zisserman, Andrew Fitzgibbon, M. Kumar, and Philip Torr. Learning class-specific edges for object detection and segmentation. In *Computer Vision, Graphics and Image Processing*, volume 4338, pages 94–105, 01 2006.

[26] Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, and Matthew Brand. Skyline2gps: Localization in urban canyons using omni-skylines. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3816–3823, 2010.

[27] Melani Sanchez-Garcia, Ruben Martinez-Cantin, and Josechu Guerrero. Semantic and structural image segmentation for prosthetic vision. *PLOS ONE*, 15:e0227677, 01 2020.

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 06 2018.

[29] Qi Shan, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M. Seitz. Occluding contours for multi-view stereo. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4002–4009, 2014.

[30] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991, 2015.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[32] Saining Xie and Z. Tu. Holistically-nested edge detection. *International Journal of Computer Vision*, 125:1–16, 12 2017.

[33] Zhiding Yu, Chen Feng, Ming-Yu Liu, and Srikumar Ramalingam. Casenet: Deep category-aware semantic edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5964–5973, 2017.