EDGE-Net: Efficient Deep-learning Gradients Extraction Network

Nasrin Akbari and Amirali Baniasadi

Department of Electrical and Computer Engineering, University of Victoria, Victoria, Canada

Abstract. Deep Convolutional Neural Networks (CNNs) have achieved impressive performance in edge detection tasks, but their large number of parameters often leads to high memory and energy costs for implementation on lightweight devices. In this paper, we propose a new architecture, called Efficient Deep-learning Gradients Extraction Network (EDGE-Net), that integrates the advantages of Depthwise Separable Convolutions and deformable convolutional networks (DeformableConvNet) to address these inefficiencies. By carefully selecting proper components and utilizing network pruning techniques, our proposed EDGE-Net achieves state-of-the-art accuracy in edge detection while significantly reducing complexity. Experimental results on BSDS500 and NYUDv2 datasets demonstrate that EDGE-Net outperforms current lightweight edge detectors with only 500k parameters, without relying on pre-trained weights.

Keywords: Efficient edge detection, lightweight deep neural network, Enhanced receptive field

1 Introduction

Edge detection refers to the process of identifying significant transitions in an image by detecting discontinuities in texture, color, and brightness, among other attributes. The detected edges provide the boundaries between different regions in the image, and this step serves as the initial stage for numerous computer vision applications, including edge-based face recognition, edge-based target recognition, scene understanding, image segmentation, fingerprint matching, license plate detection, object proposal, and object detection [1]. The widespread use of edge detection in various fields, such as fingerprint recognition in mobile devices [2], satellite image localization to suppress noise and create realistic edge maps, self-driving vehicles that adjust the steering wheel angle based on road images [3], and the identification of pathological objects in medical images [4], underscores the importance of developing effective neural networks for edge detection.

In recent years, the advent of deep learning techniques has significantly boosted edge detection research. While traditional approaches on the BSDS500 dataset often achieve a 0.59 ODS F-measure, DL-based methods can achieve an impressive 0.828 ODS [5]. However, these recently proposed architectures may be computationally inefficient, emphasizing the need for lightweight networks that can reduce the number of parameters while maintaining detection accuracy. Figure 1 provides

DOI: 10.5121/ijaia.2023.14207

insight into the detection accuracy and complexity (model size) of several well-known deep learning-based methods. The orange dot on the graph represents how well our model aligns with human perception in terms of accuracy, given its few parameters.

Many deep learning-based edge detectors use VGGNet (Visual Geometry Group) [6] as their feature-based extractor due to its impressive performance. However, the network's extensive backbone and high parameter count make it more appropriate for complex tasks such as object recognition and image segmentation. Our motivation for this study arises from the fact that edge detection is a low-level image-processing task that does not require complex networks for feature extraction.

To decrease the number of parameters and floating point operations (FLOPs), we leverage depthwise separable convolutions [7], which disentangle the spatial and channel interaction that occurs during regular convolution operations. However, this technique can reduce performance compared to conventional convolution methods. To compensate for this reduction, we increase the receptive field by selecting appropriate lightweight components for edge detection purposes. The details of this approach are discussed in section 3.

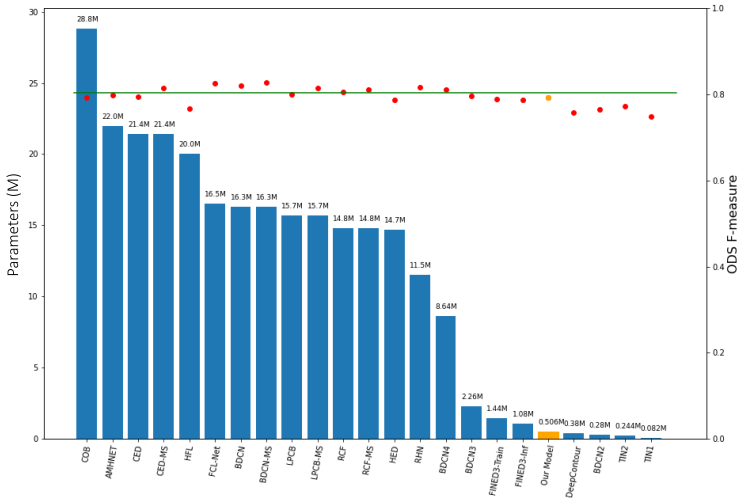

Fig. 1. Comparison of complexity and accuracy performance among various edge detection schemes. Our proposed methods (orange).

The remainder of this manuscript is arranged as follows: Section 2 provides a comprehensive review of pertinent studies and their associated challenges. Section 3 explicates the proposed network architecture. In Section 4, the outcomes

of the experiments are presented, and a comparison is made between the proposed model and state-of-the-art edge detector networks using the Berkeley Segmentation Dataset 500 (BSDS500) [8] and NYUDv2 [9] datasets. Finally, Section 5 presents concluding remarks and outlines future research directions.

2 Related Work

n recent years, a plethora of edge-detection techniques has been proposed. These approaches can be broadly classified into three groups, namely traditional edge detection, learning-based techniques utilizing handcrafted features, and deep learning networks. In this section, we will provide an overview of some of the techniques developed in the past few years.

Early pioneer edge detection methods primarily focused on intensity and color gradients. For instance, the Sobel operator [10] measures the 2-D spatial gradient of an image, emphasizing regions of high spatial frequency that correspond to edges. The Canny algorithm [11] is another well-known multi-stage edge detector that computes the intensity of the gradients by applying a filter based on the derivative of a Gaussian. By removing non-maximum pixels of the gradient magnitude, possible edges are decreased to 1-pixel curves. However, traditional approaches have limitations, including their focus solely on the changes of local intensity, while failing to recognize and remove non-edge textures.

The introduction of learning-based edge detectors partially overcame some of the challenges associated with texture detection in traditional approaches. In this group of detectors, hand-crafted features are initially extracted, and classifiers trained using these features are applied to identify edges. The first data-driven approaches were proposed by Konishi et al. [12], who used images to learn the probability distributions of responses that correspond to the two sets of edge filters. In another work [13], random decision forests were applied to identify the structure presented in local image patches, where color and gradient features were used to obtain high-quality output edges.

Although the aforementioned techniques were developed using handcrafted features, these features are limited in providing high-level information for semantically meaningful edge detection and have a limited capability of capturing edges at different scales. To address these issues, several CNN-based algorithms with strong learning capabilities have been proposed in recent years. HED [14], one of the most influential DNN-based edge detectors, uses fully convolutional neural networks and deeply-supervised nets to determine the edge probability for every pixel. HED uses VGGNet [6] for feature extraction and fuses all side outputs of VGGNet features to minimize the weighted cross-entropy loss function. Since then, various extensions based on HED and VGGNet have been developed, including CED [15], AMH-Net [16], RCF [17], LPCB [18], and BDCN [19].

Although CNN is a successful model, it often requires significant computational power and resources. Therefore, the current trend is to design efficient CNN structures that overcome these issues. Fined [20], dense extreme inception network [21], and TIN [5] have proposed lightweight architectures for edge detection. However, these networks sacrifice detection accuracy for speed. To achieve a better trade-off between accuracy and efficiency for edge detection, it is necessary to optimize the architecture and initial parameters of deep learning models, so that they consume fewer resources while maintaining accuracy. In this paper, we propose a simplified model for feature extraction by simplifying the backbone and selecting appropriate components. This model achieves good edge quality with much simpler architecture compared to other studies.

3 Lightweight Edge Detection Network

This section presents our proposed EDGE-Net, a lightweight neural network that provides high running efficiency. It offers a solution to the efficiency concerns of the models discussed in the previous section. Figure 2 illustrates the architecture of EDGE-Net. We train the network from scratch to optimize its performance. In the following paragraphs, we provide a detailed review of the components utilized in EDGE-Net.

3.1 Efficient Backbone

The majority of deep learning-based edge detectors, such as those proposed in [15–17], utilize VGGNet as their feature extraction backbone. However, we posit that edge detection is a task that can be accomplished using a less complex backbone. We achieve this by incorporating lightweight components that maintain high efficiency. In order to achieve a pyramid structure, we employ three stages, with a max-pooling operation for downsampling the features between stages. This results in a decrease in the dimension of output feature maps as we progress through the stages. As the complexity of the patterns increases in the subsequent stages, we increase the number of feature channels (i.e., the number of filters) to capture a greater number of combinations. The channel numbers for stages 1, 2, and 3 are 16, 64, and 256, respectively. Our backbone comprises mainly deformable and customized depthwise separable convolutions. To create the fused output, we use standard bilinear interpolation to upsample the low-resolution features. The fused output is then formed by concatenating all of the stage outputs. In the following sections, we provide detailed explanations of the layers and components used in EDGE-Net.

Deformable convolution The presence of geometric transformations and variations in natural images poses a significant challenge to feature extraction tasks.

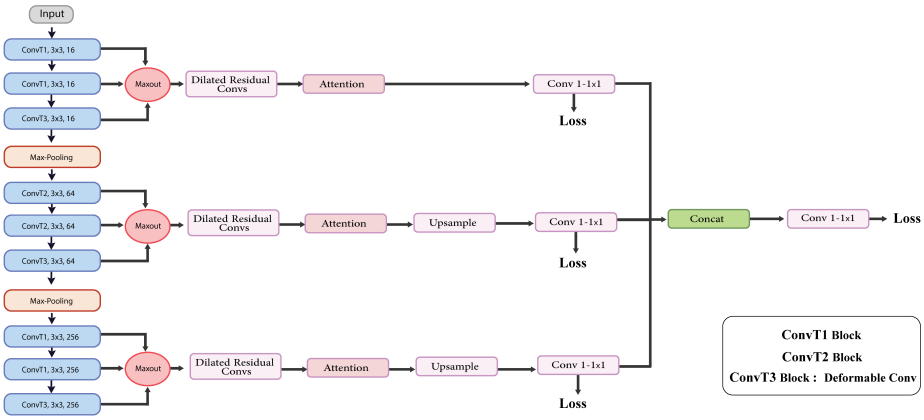

Fig. 2. EDGE-Net architecture

Fig. 3. Convt1 block - Convt2 block

Standard convolution kernels, with their fixed structure, have limitations in capturing such transformations. Deformable convolutions, on the other hand, offer a more efficient solution to this problem. These convolutions possess the ability to adapt their kernel shape and parameters to the image content, thereby accommodating geometric variations. By incorporating 2D offset kernels to the regular sampling location in the standard convolution, deformable convolutions enable the network to have different receptive fields, depending on the scale of the objects. The 2D offset kernels are learned from the preceding feature maps using additional convolutional layers and can be trained end-to-end using normal back-propagation functions. In order to keep the network light in terms of parameters and computation, we add this module at the end of each stage to strengthen the features before transferring them to the next stage [22].

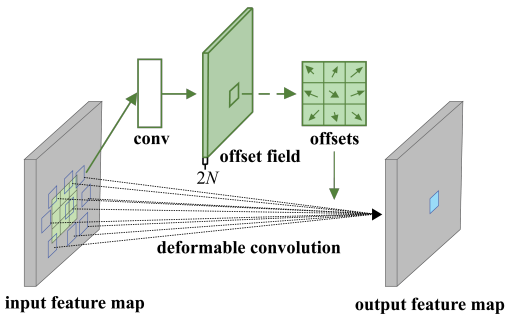

Fig. 4. Visual Representation of deformable Convolution Operation [22]

Depthwise Separable Convolution The conventional convolution operation performs computation across both channel and spatial dimensions simultaneously. In contrast, the Depthwise Separable Convolution (DSC) method [7] partitions the computation into two sequential steps: 1) depthwise convolution, which involves applying a single convolutional filter per input channel, and 2) pointwise convolution, which combines the outputs from the depthwise convolution with a linear combination. While this approach reduces the number of parameters, it also leads to a decline in accuracy. To mitigate this issue, we incorporate auxiliary side blocks to enhance the features while maintaining a minimal number of parameters.

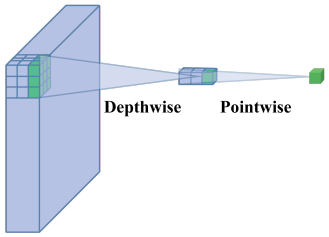

Fig. 5. Visual Representation of depthwise separable convolution [7]

In order to introduce non-linearity and enable complex decision-making in our model (Figure 3 - Convt1), we utilize the Rectified Linear Unit (RELU) activation after each pointwise convolution. To enhance the model's accuracy while minimizing the number of parameters, we have made modifications to Convt1 resulting in Convt2. This new modification involves adding a pointwise convolution with a 1×1 kernel that iterates through every point between two RELU activations. Furthermore, to address the issue of overfitting, we apply batch normalization as a regularizer after each RELU activation.

3.2 Efficient Side Structure

MAxout Layer Before transferring the inputs to the side output layers (from left to right) at each stage, we perform a Maxout operation rather than using the standard concatenation block. The Maxout activation reduces the number of parameters significantly compared to classical dense blocks by inducing competition between feature maps and accelerating network convergence. Instead of stacking the outputs of previous layers on top of each other at each stage, we keep only the maximum value at each position. This approach reduces the number of parameters and improves the model's performance.

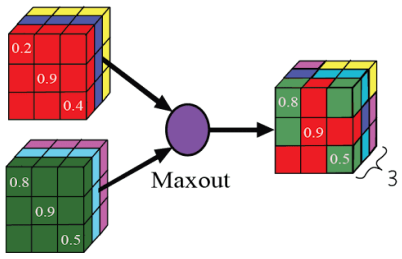

Fig. 6. Visual Representation of Maxout Layer [23]

Dilated Residual Convolution Module To improve the feature extraction process utilizing depth-wise separable convolution in the backbone, we establish a connection between each feature extraction layer and the dilated convolution module, as proposed in [5]. Various dilation sizes are utilized to capture different levels of receptive fields in the image. Specifically, we adopt a dilation sequence of 4, 8, 12, and 16 for all layers, each with 32 filters. Following pixel-wise aggregation, we incorporate hierarchical residual-like connections to enhance the multi-scale representation ability at a more granular level. Importantly, this module can be readily incorporated into state-of-the-art backbones. Figure 10 illustrates the design of the proposed Dilated Residual Convolution (DDR) module.

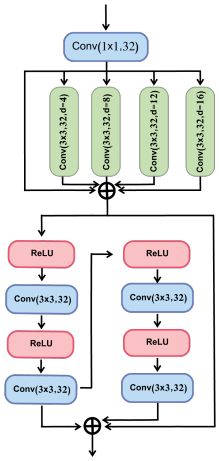

Fig. 7. Visual Representation of dilated residual convolution module

Convolutional Block Attention Module (CBAM) We incorporate a lightweight spatial and channel attention module, as originally proposed by Woo et al. [24], subsequent to the dilated residual convolution block. This module selectively accentuates pertinent features while dampening others.

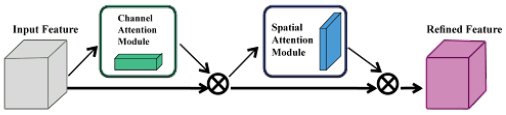

Fig. 8. Visual Representation of attention module

The spatial attention mechanism captures the inter-spatial relationships of features to determine the informative regions within the image, and is achieved through a series of steps. First, average pooling and max pooling are applied to summarize the presence of features and their activations, respectively. Next, a convolutional layer is utilized in conjunction with a concatenated feature descriptor to generate a spatial attention map that specifies the locations to accentuate or diminish features, as described by Woo et al. [24].

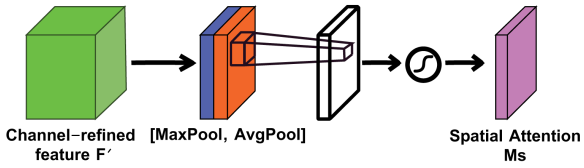

Fig. 9. Visual Representation of spatial attention module

The channel attention block redistributes the channel feature responses to enhance the importance of specific channels while attenuating others. To calculate the channel attention, the spatial dimension of the input feature map is first reduced, a process known as squeezing, as proposed by Woo et al. [24].

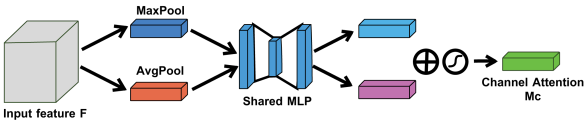

Fig. 10. Visual Representation of channel attention module

3.3 Loss Function

In an image, the distribution of edge and non-edge pixel data is often imbalanced. While CNN models may achieve high accuracy by predicting the majority class, they may overlook the minority class, leading to a misleading accuracy estimate. To address this issue, we adopt the weighted Cross-Entropy loss function proposed by Liu et al. [17].

During network training, we compare all stages and fused outputs to the ground truth. Specifically, we use the following equation to compare each pixel of each image to its corresponding label.

$$L(x_i; W) = \begin{cases} \alpha.log\big(1 - P(x_i; W)\big) & if y_i = 0 \\ 0 & if \leq y_i \leq \eta \\ \beta.logP(x_i; W) & otherwise, \end{cases} \tag{1}$$

in which

$$\alpha = \lambda . \frac{|Y^+|}{|Y^+| + |Y^-|}$$
$$\beta = \frac{|Y^-|}{|Y^+| + |Y^-|} \tag{2}$$

The variables X, P(X), Y, W, and η represent features extracted from the CNN network, the output of the standard sigmoid function, the ground truth edge probability, all the parameters learned in the CNN network, and the percentage of non-edge and edge pixels, respectively. The hyper-parameter η is used to balance the number of positive and negative samples. Since each image is labeled by multiple annotators, and human cognition varies, a predefined threshold is employed to distinguish between edge and non-edge pixels in the edge probability map. If a pixel is labeled by fewer than η of the annotators, it is considered a non-edge pixel.

To generalize the loss function to all pixels in the image (I) across each stage (k) and fused layer, the following loss function is used.

$$L(W) = \sum_{i=1}^{|I|} \Big(\sum_{k=1}^{|K|} L\big(x_i^k; W\big) + L\big(x_i^{fuse}; W\big) \Big) \tag{3}$$

4 Experiments And Discussions

4.1 Implementation Details

We implemented our backbone networks using PyTorch and initialized their stages with a zero-mean Gaussian distribution with a standard deviation of 0.01. The learning rate was set to 0.01 initially and then updated using a linear scaling factor by multiplying 0.1 for every two epochs. We used stochastic gradient descent as the optimizer, and we terminated the training process after eight epochs. We conducted all the experiments on a single GPU, NVIDIA GeForce 2080Ti, which has 11G memory.

4.2 Dataset

To ensure a fair comparison with other published works in tables 1 and 2, we evaluated our proposed network on the same datasets as those studies. Specifically, for the Berkeley Segmentation Dataset (BSDS500) [8] and NYUDv2 [9], we used the same evaluation procedure. The BSDS500 dataset contains 500 images, with 200 for training, 100 for validation, and 200 for testing. We combined the training and validation sets to create our training set, and applied the same data augmentation

techniques as in RCF [17], including using the PASCAL VOC dataset [25] and its flipped images.

The NYUD dataset consists of 1449 densely labeled pairs of aligned RGB and depth images (HHA), captured by Microsoft Kinect cameras in various indoor scenes. The dataset includes 381 training, 414 validation, and 654 testing images. To augment the dataset, we rotated the images and corresponding annotations to four different angles (0, 90, 180, and 270 degrees) and flipped them at each angle, following the approach in RCF [17].

4.3 Performance Metrics

It is noteworthy that the proportion of edge pixels in an image is typically only around 10%, while the remaining 90% of pixels are non-edge. This substantial class imbalance renders accuracy an inadequate metric for evaluating edge detection performance, as a model could achieve high accuracy simply by predicting the majority class of non-edges. To address this, we employ F-Score as a more appropriate evaluation metric. F-Score accounts for both precision and recall, with optimal performance occurring at a score of one and the poorest possible score at zero.

- Recall = TruePositives / (TruePositives + FalseNegatives)
- Precision= TruePositives / (TruePositives + FalsePositives)
- F-Measure = (2 * Precision * Recall) / (Precision + Recall)

There are two methods to calculate the optimal threshold and the corresponding F-score for binarizing the output of the CNN network to make it comparable to the binarized ground truth.

- Optimal Dataset Scale: Iterates over all possible thresholds and set one threshold for the entire dataset. The threshold that gives the best F-score for the dataset is used to calculate ODS score.
- Optimal Image Scale: Finds the best threshold and corresponding F-score for each image. The OIS F-score is calculated by averaging all of the F-scores for all images.

4.4 Comparison with State-of-the-arts.

On BSDS500 dataset - In terms of F-score and number of parameters, we compare our methods to prior edge detection approaches, including both traditional and recently proposed CNN-based models. As shown in Table 1 and Figure 11, our baseline model achieves outstanding results (ODS of 0.792 and OIS of 0.805) while using significantly fewer parameters, which are equal or better than most recent lightweight CNN models such as BDCN2, TIN1, TIN2, FINED3-Inf, and FINED3-Train [20].

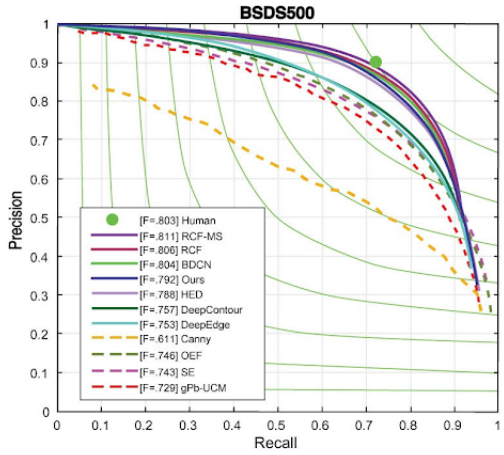

Fig. 11. Precision-Recall curves of our models and some competitors on BSDS500 dataset.

On NYUD dataset - Table 2 presents the comparison results for the NYUD dataset, while figure 12 depicts the precision-recall curves. To test our model on NYUD, we adopt network settings similar to those used for BSDS500. Some studies employ two separate models to train RGB images and HHA feature images of NYUD and report the evaluation metrics on the average of the models' outputs. However, our network is only tested on RGB images. Therefore, to ensure a fair evaluation, we compare our model's output with those of models that were also tested solely on RGB images.

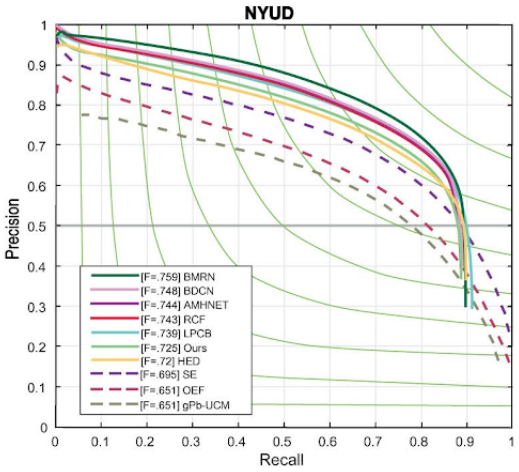

Fig. 12. Precision-Recall curves of our models and some competitors on NYUD dataset.

Method	ODS	OIS	P(Million)
Canny	0.611	0.676	-
OEF	0.746	0.77	-
gPb-UCM	0.72	0.755	-
SE	0.743	0.763	-
AMHNET	0.798	0.829	22
BDP-Net	0.808	0.828	18.7
FCL-Nt	0.826	0.845	16.5
BAN	0.81	0.827	15.6
LPCB	0.815	0.834	15.7
BMRN	0.828	0.81	+14.8
RCF	0.806	0.823	14.8
HED	0.788	0.808	14.7
COB	0.793	0.82	28.8
RHN	0.817	0.833	11.5
CED	0.815	0.834	21.4
DeepEdge	0.753	0.772	-
DeepContour	0.757	0.776	0.38
BDCN	0.82	0.838	16.3
BDCN2	0.766	0.787	0.48
BDCN3	0.796	0.817	2.26
BDCN4	0.812	0.83	8.69
TIN1	0.749	0.772	0.08
TIN2	0.772	0.792	0.24
FINED3-Inf	0.788	0.804	1.08
FINED3-Train	0.79	0.808	1.43
Our Model	0.792	0.805	0.506

Table 1. Comparison to other methods on BSDS500 dataset.

Method	ODS	OIS	P(Million)
OEF	0.651	0.667	-
gpb-UCM	0.632	0.661	-
SE	0.695	0.708	-
SE+NG+	0.706	0.734	-
AMHNET	0.744	0.758	22
BDCN	0.748	0.763	16.3
LPCB	0.739	0.754	15.7
RCF	0.743	0.757	14.8
BMRN	0.759	0.776	+14.8
HED	0.72	0.734	14.7
Our Model	0.725	0.738	0.5

Table 2. Comparison with other methods on NYUD dataset.

5 Conclusion

Efficient architecture design is crucial for the implementation of edge detection, which has practical applications in various fields. Most deep neural networks for edge detection use transfer learning from pre-trained models like VGG16, which have a large number of parameters and are trained for high-level tasks. However, edge detection requires a simple set of features and does not necessitate a large number of convolutional layers for feature extraction. In this study, we introduce a new lightweight architecture that achieves state-of-the-art performance. Our network leverages customized depth-wise separable and deformable convolutions for edge detection and incorporates lightweight components to increase the receptive field of our model, resulting in high-quality edges. Our network architecture is versatile and has the potential to be extended for other vision tasks, such as salient object detection and semantic segmentation.

References

1. Victor Wiley and Thomas Lucas. Computer vision and image processing: a paper review. *International Journal of Artificial Intelligence Research*, 2(1):29–36, 2018.
2. Ronald J Holyer and Sarah H Peckinpaugh. Edge detection applied to satellite imagery of the oceans. *IEEE transactions on geoscience and remote sensing*, 27(1):46–56, 1989.
3. Abhishek Gupta, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.
4. Wei-Chun Lin and Jing-Wein Wang. Edge detection in medical images with quasi high-pass filter based on local statistics. *Biomedical Signal Processing and Control*, 39:294–302, 2018.
5. Jan Kristanto Wibisono and Hsueh-Ming Hang. Traditional method inspired deep neural network for edge detection. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 678–682. IEEE, 2020.
6. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
7. Yunhui Guo, Yandong Li, Liqiang Wang, and Tajana Rosing. Depthwise convolution is all you need for learning multiple visual domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8368–8375, 2019.
8. Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.
9. Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.
10. O Rebecca Vincent, Olusegun Folorunso, et al. A descriptive algorithm for sobel image edge detection. In *Proceedings of informing science & IT education conference (InSITE)*, volume 40, pages 97–107, 2009.
11. Renjie Song, Ziqi Zhang, and Haiyang Liu. Edge connection based canny edge detection algorithm. *Pattern Recognition and Image Analysis*, 27(4):740–747, 2017.
12. Scott Konishi, Alan L. Yuille, James M. Coughlan, and Song Chun Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):57–74, 2003.

13. Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570, 2014.
14. Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.
15. Yupei Wang, Xin Zhao, and Kaiqi Huang. Deep crisp boundaries. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3892–3900, 2017.
16. Dan Xu, Wanli Ouyang, Xavier Alameda-Pineda, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. Learning deep structured multi-scale features using attention-gated crfs for contour prediction. *Advances in neural information processing systems*, 30, 2017.
17. Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3000–3009, 2017.
18. Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 562–578, 2018.
19. Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3828–3837, 2019.
20. Jan Kristanto Wibisono and Hsueh-Ming Hang. Fined: Fast inference network for edge detection. *arXiv preprint arXiv:2012.08392*, 2020.
21. Xavier Soria Poma, Edgar Riba, and Angel Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1923–1932, 2020.
22. Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.
23. Leonie Henschel, Sailesh Conjeti, Santiago Estrada, Kersten Diers, Bruce Fischl, and Martin Reuter. Fastsurfer-a fast and accurate deep learning based neuroimaging pipeline. *NeuroImage*, 219:117012, 2020.
24. Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
25. Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014.