Adversarial Network for edge detection

Zhiliang Zeng, Ying Kin Yu, Kin Hong Wong
Department of Computer science and Engineering
The Chinese University of Hong Kong
zlzeng@cse.cuhk.edu.hk

***Abstract*—Edge detection is a fundamental problem in computer vision and has been explored for many decades. Due to the rapid development of machine learning techniques and their applications to image processing, there is a proliferation of neural network-based approaches to solve the edge detection problem. These methods have good performance and even outperform human beings. Most of the existing neural network-based systems use the convolutional network or its variant. They usually produce thick edges and the application of non-maximum-suppression to suppress the edge is necessary. In this paper, we explore another type of neural network called the conditional generative adversarial network (cGAN) to address the edge detection problem. cGAN is an innovative framework to do the image synthesis task. It can generate an image close to the real one. After training, our network can produce an edge map that contains more detailed information and thinner edges compared to the state-of-the-art methods that require the well-known non-maximum-suppression for post-processing. The proposed approach is able to produce a high quality edge map directly without further processing. Our solution is computation efficient. It can achieve a speed of 59 and 26 frames per second (fps) for an image resolution of 256x256 and 512x512, respectively.**

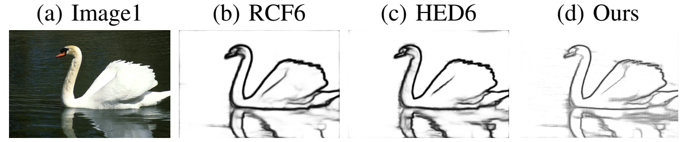
I. Introduction

Edge detection is a crucial element in many computer vision tasks. It aims to extract the boundaries of the objects in natural images. The edge information can be used in many image processing applications, such as texture removal [30], semantic segmentation [4] and object proposal [32].

Even though the edge detection problem has been explored for a number of decades, there is still a large room for improvement. Usually, an edge can be defined to be semantically meaningful. Many of the existing methods are trying to capture semantic information for better results. Some examples of the pioneering work are Sobel detector [18] and Canny detector [7]. They are based on the gradient of colour or brightness to predict the contour. Later work like [24] and [1] uses hand craft features to distinguish the foreground contour from the background. Machine learning-based methods are also popular. They aim to capture more complex semantic representations from the images and make edge prediction better, for examples [11] and [10].

More recent approaches employ neural networks. Due to the success of the convolutional networks in various computer vision tasks like image classification [27], neural network has been shown to have a powerful capability to extract high level features. Some examples are DeepEdge [5], DeepCoutours [16], N^4 field [13], DCAN [8], HED [28] and RCF [23].

(a) Image1 (b) RCF6 (c) HED6 (d) Ours



(e) Image2 (f) RCF6 (g) HED6 (h) Ours

Fig. 1. We have built a simple generator network based on UNET [25] to produce the edge map. Two samples of the resulting edge images are shown. The results of RCF [23] and HED [28] are denoted by RCF6 and HED6 in the above figures. Since RCF and HED produce multiple outputs, we only compare the final output of both networks. It is obvious that our results contain more detailed information. Our method is able to produce thinner edges even without the application of non-maximum suppression.

They can achieve better results than previous methods that solve the problem in an analytical way.

Most neural network-based methods use convolutional network or its variant and usually contain pooling layers. The pooling layer reduces the resolution of the feature map by filtering out the small details using a 2x2 smoothing filter. This step is useful in saving computer and graphic card memory. However, it causes feature loss. Useful features may be discarded while pooling.

Instead of training a hierarchical convolutional network to extract meaningful features for the prediction of the final result, researcher has recently proposed a different type of network to generate the results more directly. This network is known as the GAN [14] network. GAN network is fundamentally different from convolutional network in way that it simultaneously trains two adversarial models. One is a generator and the other is a discriminator. After training, the generator is be able to generate the target result.

Many common computer vision tasks, like edge detection, can be formulated as an image to image transformation problem. Given a natural image, we would like to compute an edge map as the output. The work described in cGAN [17] shows that the GAN framework is good at solving this kind of problems.

In this paper, we explore the cGAN framework to tackle the edge detection problem. We first try to train the vanilla cGAN with dataset BSD500 [2]. Since the original BSD500 dataset is a small one, we use augmentation to enlarge the size of the dataset similar to the method described in HED [28]. After training, the generator of cGAN can achieve an optimal dataset scale (ODS) score upto 0.749, and an optimal image scale (OIS) score up to 0.777. Moreover, if we apply the multiscale image prediction scheme, we can further improve the ODS to 0.772 and OIS to 0.797. The results are comparable to other machine learning-based methods, such as HED [28], (ODS=0.788, OIS=0.808). Fig 1 shows the comparison of our method with other neural network-based approaches.

Our paper is organized as follows. Section 2 discusses the background of our work. In Section 3, the theory and design methodology are described. The implementation details and experimental results are illustrated in Section 4. The conclusion is found in Section 5.

II. Related work

There is a proliferation of researches in the application of neural networks to computer vision problems, especially in image classification. They have great improvement over the traditional approaches that tackle the problem analytically. It is believed that the hierarchical structure of the convolutional network has a powerful capability to capture high level features, such as semantic and context features. These features are important for image classification, so as the edge detection problem. Recent work based on neural network, such as HED [28], and RCF [23], shows that their results are better than previous algorithms. Machine learning-based methods, like the solutions by Doller [11] [10], also show significant improvements.

Instead of only using the features from the last layer of the trained convolutional neural network like the previous methods [5] [8] [13] [16], HED [28] tends to use the features from each stage of convolutional layers to get a better result. RCF [23] goes one step forward. It employs all features from the convolutional layers to improve the performance. Both HED [28] and RCF [23] apply a deeply supervised method [20] to maximize their ability to capture image features. The labels are not only used to supervise the prediction but are also used to supervise hidden status. After each training step, their networks are able to optimize the hidden features and predictions. However, the problem is that adding so many semantic features also brings too much background information into the fused features. The final prediction contains too much detailed background information that is hard to be removed. Both RCF and HED require the use of non-maximum-suppression for post-processing in order to remove the background details from the final results and, at the same time, make the resulting edges thinner.

In recent years, there is a new branch of neural network structure, known as the generative adversarial network (GAN) [14]. Such a technique has been successfully applied to image synthesis problems. There are a number of variants of GAN network. One is called the conditional GAN structure. It has been shown that it is good at handling many image generation tasks [17]. More specifically, GAN consists of two networks. One is the generator (G) and the other is the discriminator (D). After training, the generator (G) can produce images that are very close to the real images in the dataset. Inspired by this property, we apply cGAN to solve the edge detection problem. Our approach is similar tothe pix2pix method [17]. It is shown in a later section that our trained cGAN generator can output a high quality edge map even without the non-maximum-suppression procedure.

III. Theory

A. The adversarial network

The generative adversarial network is a new approach and has beenn recently getting more attention. It is different from the conventional convolutional neural network in a way that there are two networks operating simultaneously in the GAN structure. One is known as the generator represented by G while the other is known as the discriminator represented by D [14]. We define the images from the dataset as real images and the images produced by the generator are fake images. The objective of D is to distinguish whether the input image is real or not while the goal of G is to produce a fake image that looks similar to the real one. Actually, the GAN is hard to train. Therefore, researchers devised an alternative method called the conditional GAN (cGAN) to alleviate the problem. Recently, cGAN has been successfully applied to generating images [17] in many real-life applications. Inspired by this paper, we have adopted their framework [17] to solve the edge detection problem. An overview of our edge detection algorithm can be found in Fig 2.

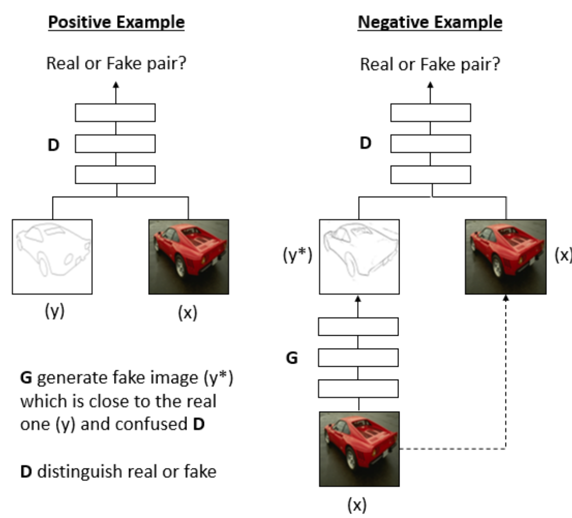


Fig. 2. Instead of directly predicting whether y or $y*$ is real or fake, our discriminator is used to predict whether a pair of (x, y), or $(x, y*)$ is real pair or faker. This is different from the original version in [17]. We use the generator to map input image from x to y.

B. The Proposed Method

Our cGAN framework is similar to the one described in [17]. In our design, we do not feed any noises to input z of the generator as shown in Fig. 2. In vanilla GAN, G is the mapping noise z to one label image y in dataset, i.e. $G : z \rightarrow y$. D is trained to detect fake images. The objective of vanilla GAN can be expressed as

$$
\begin{aligned}
\mathcal{L}_{GAN} &= \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] \\
&+ \mathbb{E}_{z\sim p_z(z)}[1 - log(D(G(z)))]
\end{aligned} \tag{1}
$$

GAN is hard to be implemented because equation 1 cannot provide sufficient weight gradient information for the generator to learn well [14] when training starts. Instead of mapping z to label image y in the geneartion process, conditional GAN maps real image x in the dataset to y. The objective of conditional GAN can be expressed as

$$
\begin{aligned}
\mathcal{L}_{cGAN} &= \mathbb{E}_{(x,y)\sim p_{data}(x,y)}[logD(x,y)] \\
&+ \mathbb{E}_{x\sim p_{data}(x)}[log(1 - D(x, G(x)))]
\end{aligned} \tag{2}
$$

According to the literature in [17], adding L1 distance measurement to the cGAN objective function is beneficial for training. Since the generator targets to predict the edge map, there are only two classes in the map. The positive class represents the edges while the negative class represents the background. Normally, most of the pixel values belong to the negative class in one edge map. When calculating the loss, we need to balance the errors from both the positive and negative classes. We have implemented this idea using the L1 distance, which is formulated as

$$
\mathcal{L}1 = \begin{cases} \alpha \|G(x_i) - y_i\|, & \text{if} y_i > \eta \\ 0, & 0 < y_i \leq \eta \\ \beta \|G(x_i) - y_i\|, & \text{otherwise} \end{cases} \tag{3}
$$

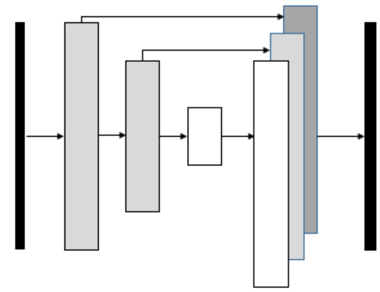
If the corresponding pixel value y_i in the label is greater than η, it is set to 0.5 in our program. Then one hyper parameter α is used to enlarge the L1 error of the pixel value x_i in the positive class of the predicted edge map. If y_i equals to zero, it is a background pixel. Likewise, another hyper-parameter β is applied to reduce the negative log-likelihood error. Inspired by RCF [23] and HED [28], we compute the hyper parameter α based on the number of positive and negative class labels. Below shows the equations of these parameters.

$$
\begin{aligned}
\alpha &= 1 + \frac{Y^-}{Y^+ + Y^-} \\
\beta &= \frac{Y^+}{Y^+ + Y^-}
\end{aligned} \tag{4}
$$

Our final objective loss function can be expressed as

$$
\mathcal{L}_{obj} = arg \min_G \max_D [\mathcal{L}_{cGAN} + \lambda \mathcal{L}1] \tag{5}
$$

Here, we use weight λ, which is set to 100.0, to enlarge $\mathcal{L}1$ distance [17].



Modified UNET

Fig. 3. We follow the UNET [25] framework in the design of our generator. We have simplified the decoder part of the original UNET by directly concatenating all the hierarchical features at each stage. The modified network can achieve similar results as the original UNET. All the convolutional layers in our network output features having the same size as in the input stage.

C. Our Network Architecture

Our generator is modified from the UNET [25] framework and is shown in Fig 3. The classical VGG16 [27] configuration is used in our discriminator. It is changed in a way that there is only one neuron in the output layer, as our network is only required to distinguish real and fake images. The other parts are built according to the vanilla VGG16 [27] structure.

The pix2pix network has already demonstrated that the UNET structure is sufficient for the image generation task [17]. However, UNET has a symmetric structure. If it contains N convolutional layers for encoding, there are also N deconvolutional layers for decoding in the network. Besides, UNET needs to combine corresponding features in the convolutional and deconvolutional layer. The feature sizes should be the same. Otherwise, cropping or padding is required to fulfill such a requirement. In other words, we need to maintain N pairs of conv-deconv layer having the same feature size.

To simplify the procedure, feature propagation is discarded in the deconvolutional layer and a skip layer is used to combine all the hierarchical features into a single deep feature cube. To ensure that all feature sizes are the same, we simply apply cropping and padding operation to resize all the feature maps to the input size.

We have changed the deconvolutional layer to a periodic shuffle layer, which is similar to settings in [26]. The advantage of using a shuffle layer is that it contains no additional variables, making the model size smaller. We have also modified the convolutional layer in the fifth stage in our proposed network. We use a dilated convolutional layer to fill up some of the holes into the convolutional kernel. In this way, the dilated layer can extract semantic information without reducing the resolution of the feature map. [31].

IV. Experiments

We used the BSD500 [2] dataset to evaluate our generator. It is a widely adopted benchmark in edge detection. There are totally 500 examples in the dataset, in which 200 for training, 200 for testing, and 100 for validation. Since the dataset is small, many researchers in the previous projects, such as

RCF [23] and HED [28], suggest to use the augmentation method to enlarge the dataset. In this paper, we applied the same augmentation strategy as HED [28] by rotating and scaling the images. Finally, we obtained a total of 28800 examples for training. We also generated multiscale images to form a pyramid for edge detection. The final prediction was an average of all the multiscale predictions.

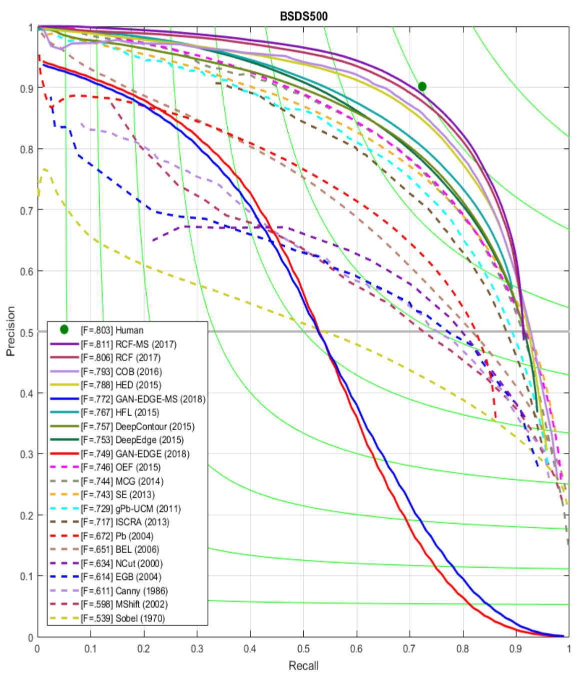


Fig. 4. The precision-recall curves of the algorithms under comparison. BSD500 dataset [2] was adopted in the evaluation. Both of our single-scale and multiscale approaches achieved good performances that were comparable to the other latest methods.

To compare with other edge detection methods, we used the same evaluation metric to illustrate our edge detection results. Normally, a threshold is necessary to produce the final edge map when an edge probability map is given. There are various methods to determine the threshold, out of which two well-known evaluation metrics can be used. The first one is called the optimal dataset scale (ODS), which applies a fixed threshold for all edge probability maps. The second one is known as the optimal image scale (OIS), which tries to apply different thresholds to the images and then selects the optimal one from the trial values. For both ODS and OIS, we used F-measure to compare the algorithm performances. The formula can be expressed as $(\frac{2 \cdot Precision \cdot Recall}{Precision + Recall})$.

Figure 4 shows the precision-recall curve and Table I shows the scores of the methods under comparison. The proposed method outperformed a number of tradiational methods. It was able to achieve a score similar to other state-of-the-art detectors. Our single-scale model got a F-measure score of 0.749, which was quite close to DeepEdge [5] and DeepContour [8]. Our multiscale method got a score of 0.772, which was comparable to that of the HED [28] approach.

TABLE I
A Comparison on the Performances of the Latest Edge Detection Algorithms Using BSDS500 [2] Dataset. † denotes the computation time with GPU

Method	ODS	OIS	FPS
Canny [7]	.611	.676	28
EGB [12]	.614	.658	10
MShift [9]	.598	.645	1/5
gPb-UCM [1]	.729	.755	1/240
Sketch Tokens [21]	.727	.746	1
MCG [3]	.744	.777	1/18
SE [11]	.743	.763	2.5
OEF [15]	.746	.770	2/3
DeepContour [8]	.757	.776	1/30†
DeepEdge [5]	.753	.772	1/1000†
HFL [6]	.767	.788	5/6†
N^4-Fields [13]	.753	.769	1/6†
HED [28]	.788	.808	30†
RDS [22]	.792	.810	30†
CEDN [29]	.788	.804	10†
MIL+G-DSN+MS+NCuts [19]	.813	.831	1
RCF [23]	.806	.823	30†
RCF-MS [23]	.811	.830	8†
GAN-EDGE (Ours)	.749	.777	26†
GAN-EDGE-MS (Ours)	.772	.797	1/18†

V. Conclusion

In this paper, we have devised an innovative edge detection algorithm based on generative adversarial network. Our approach achieved ODS and OIS scores on natural images that are comparable to the state-of-the-art methods. Our model is computation efficient. It took 0.016 seconds to compute the edges from an image having a resolution of $224 \times 224 \times 3$ with GPU. For a $512 \times 512 \times 3$ image, it took 0.038 seconds. Our algorithm is devised based on the UNET and the conditional generative adversarial neural network (cGAN) architecture. It is totally different from the convolutional networks in a way that cGAN can produce an image which is close to the real one. Therefore, the edges resulting from the cGAN generator is much thinner compared to that from the existing convolutional networks. Even without using any pre-trained network parameters, the proposed method is still able to produce high quality edge images.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011. 1, 4

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011. 1, 3, 4

[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014. 4

[4] P. Arbelez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, June 2014. 1

[5] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. *CoRR*, abs/1412.1123, 2014. 1, 2, 4

[6] G. Bertasius, J. Shi, and L. 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. 4
[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986. 1, 4
[8] H. Chen, X. Qi, L. Yu, and P. Heng. DCAN: deep contour-aware networks for accurate gland segmentation. *CoRR*, abs/1604.02677, 2016. 1, 2, 4
[9] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002. 4
[10] P. Dollr and C. L. Zitnick. Structured forests for fast edge detection. In *2013 IEEE International Conference on Computer Vision*, pages 1841–1848, Dec 2013. 1, 2
[11] P. Dollr and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570, Aug 2015. 1, 2, 4
[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004. 4
[13] Y. Ganin and V. S. Lempitsky. N^4-fields: Neural network nearest neighbor fields for image transforms. *CoRR*, abs/1406.6558, 2014. 1, 2, 4
[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 1, 2, 3
[15] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1732–1740, 2015. 4
[16] J. Hwang and T. Liu. Pixel-wise deep learning for contour detection. *CoRR*, abs/1504.01989, 2015. 1, 2
[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2, 3
[18] J. Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1(1):37 – 42, 1983. 1
[19] I. Kokkinos. Surpassing humans in boundary detection using deep learning. *CoRR*, abs/1511.07386, 2015. 4
[20] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In G. Lebanon and S. V. N. Vishwanathan, editors, *AISTATS*, volume 38 of *JMLR Proceedings*. JMLR.org, 2015. 2
[21] J. J. Lim, C. L. Zitnick, and P. Dollar. Sketch tokens: A learned mid-level representation for contour and object detection. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3158–3165, Washington, DC, USA, 2013. IEEE Computer Society. 4
[22] Y. Liu and M. S. Lew. Learning relaxed deep supervision for better edge detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 231–240, 2016. 4
[23] Y. Liu, X. H. Ming-Ming Cheng, K. Wang, and X. Bai. Richer convolutional features for edge detection. In *IEEE CVPR*, 2017. 1, 2, 3, 4
[24] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, May 2004. 1
[25] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 1, 3
[26] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016. 3
[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 3
[28] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015. 1, 2, 3, 4
[29] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. 2016. 4
[30] Q. Yang. Semantic filtering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1
[31] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 3
[32] L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In *ECCV*. European Conference on Computer Vision, September 2014. 1