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
zizeng@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 computationally 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], DeepContours [16], N^4 field [13], DCAN [8], HED [28] and RCF [23].

Fig. 1. We have built a simple generator network based on U-NET [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 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
are very close to the real images in the dataset. Inspired by (D). After training, the generator (G) can produce images that are 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 property, we apply cGAN to solve the edge detection problem. Our approach is similar to the 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 been 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 property, we apply cGAN to solve the edge detection problem. An overview of our edge detection algorithm can be found in Fig 2.

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 to the 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 been 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 property, we apply cGAN to solve the edge detection problem. An overview of our edge detection algorithm can be found in Fig 2.

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 to the 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 been 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 property, we apply cGAN to solve the edge detection problem. An overview of our edge detection algorithm can be found in Fig 2.

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 to the 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 been 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 property, we apply cGAN to solve the edge detection problem. An overview of our edge detection algorithm can be found in Fig 2.
trained to detect fake images. The objective of vanilla GAN can be expressed as

\[
\mathcal{L}_{GAN} = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log(D(G(z)))]
\]  

(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 generation process, conditional GAN maps real image \( x \) in the dataset to \( y \). The objective of conditional GAN can be expressed as

\[
\mathcal{L}_{cGAN} = \mathbb{E}_{(x,y) \sim p_{data}(x,y)}[\log D(x,y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D(x,G(x)))]
\]  

(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

\[
L1 = \begin{cases} 
\alpha \|G(x_i) - y_i\|, & \text{if } y_i > \eta \\
0, & 0 < y_i \leq \eta \\
\beta \|G(x_i) - y_i\|, & \text{otherwise}
\end{cases}
\]  

(3)

If the corresponding pixel value \( y_i \) in the label is greater than \( \eta \), it is set to 0.5 in our program. Then one hyper parameter \( \alpha \) 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 \( \beta \) is applied to reduce the negative log-likelihood error. Inspired by RCF [23] and HED [28], we compute the hyper parameter \( \alpha \) based on the number of positive and negative class labels. Below shows the equations of these parameters.

\[
\alpha = 1 + \frac{Y^-}{Y^+ + Y^-} \\
\beta = \frac{Y^+}{Y^+ + Y^-}
\]  

(4)

Our final objective loss function can be expressed as

\[
\mathcal{L}_{obj} = \arg \min_G \max_D [\mathcal{L}_{cGAN} + \lambda L1]
\]  

(5)

Here, we use weight \( \lambda \), which is set to 100.0, to enlarge L1 distance [17].
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.

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 called 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 \( \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \).

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 traditional 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

<table>
<thead>
<tr>
<th>Method</th>
<th>ODS</th>
<th>OIS</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canny [7]</td>
<td>0.611</td>
<td>0.676</td>
<td>28</td>
</tr>
<tr>
<td>EGB [12]</td>
<td>0.614</td>
<td>0.658</td>
<td>10</td>
</tr>
<tr>
<td>MShift [9]</td>
<td>0.598</td>
<td>0.645</td>
<td>1/5</td>
</tr>
<tr>
<td>gPb-UCM [1]</td>
<td>0.729</td>
<td>0.755</td>
<td>1/20</td>
</tr>
<tr>
<td>Sketch Tokens [21]</td>
<td>0.727</td>
<td>0.746</td>
<td>1</td>
</tr>
<tr>
<td>MCG [3]</td>
<td>0.744</td>
<td>0.777</td>
<td>1/18</td>
</tr>
<tr>
<td>SE [11]</td>
<td>0.773</td>
<td>0.763</td>
<td>2.5</td>
</tr>
<tr>
<td>OEF [15]</td>
<td>0.746</td>
<td>0.770</td>
<td>2/3</td>
</tr>
<tr>
<td>DeepContour [8]</td>
<td>0.757</td>
<td>0.776</td>
<td>1/30</td>
</tr>
<tr>
<td>DeepEdge [5]</td>
<td>0.753</td>
<td>0.772</td>
<td>1/1001†</td>
</tr>
<tr>
<td>HFL [6]</td>
<td>0.767</td>
<td>0.788</td>
<td>5/6†</td>
</tr>
<tr>
<td>N^4-Fields [13]</td>
<td>0.753</td>
<td>0.769</td>
<td>1/6†</td>
</tr>
<tr>
<td>HED [28]</td>
<td>0.788</td>
<td>0.808</td>
<td>30†</td>
</tr>
<tr>
<td>RDS [22]</td>
<td>0.792</td>
<td>0.810</td>
<td>30†</td>
</tr>
<tr>
<td>CEDN [29]</td>
<td>0.788</td>
<td>0.804</td>
<td>10†</td>
</tr>
<tr>
<td>MIL*G-DSN+MS+NCuts [19]</td>
<td>0.813</td>
<td>0.831</td>
<td>1</td>
</tr>
<tr>
<td>RCF [23]</td>
<td>0.806</td>
<td>0.823</td>
<td>30†</td>
</tr>
<tr>
<td>RCF-MS [23]</td>
<td>0.813</td>
<td>0.830</td>
<td>8†</td>
</tr>
<tr>
<td>GAN-EDGE (Ours)</td>
<td>0.749</td>
<td>0.777</td>
<td>26†</td>
</tr>
<tr>
<td>GAN-EDGE-MS (Ours)</td>
<td>0.772</td>
<td>0.797</td>
<td>1/18†</td>
</tr>
</tbody>
</table>

### 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


[31] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In ICLR, 2016. 3