Implementing image de-raining by ID-CGAN

Jintao Wu

515021910117

Ⅰ. Introduction

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other (thus the “adversarial”). It was introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio, in 2014. GANs’ potential is huge, because they can learn to mimic any distribution of data.

The GAN consists of two parts, the generator as well as the discriminator. One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data it reviews belongs to the actual training dataset or not.

As shown in Figure 1, there are mainly four steps for GAN to take. First, the generator takes in random numbers and returns an image. Second, this generated image is fed into the discriminator alongside a stream of images taken from the actual dataset. Third, the discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.[1]

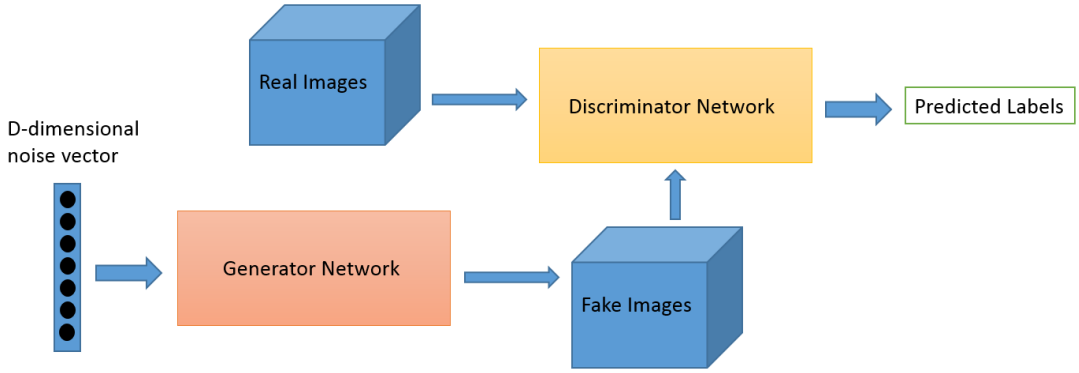

Figure 1. The procedure of GAN

The success of GANs in synthesizing realistic images has led to researchers exploring the GAN framework for numerous applications such as style transfer, image inpainting, text to image translation, image to image translation, texture synthesis and generating outdoor scenes from attributes [2]. In the paper “Image De-raining Using a Conditional Generative Adversarial Network”, the author proposed to realizing the image de-raining using a improved GAN, that is ID-CGAN (Image De-raining Conditional General Adversarial Network).

Ⅱ. Completed work

The project is based on the paper “Image De-raining Using a Conditional Generative Adversarial Network”. The aim of the project is to implementing the Image De-raining proposed in the paper. During the project, I have researched different structures of GAN and different types of loss function. The work completed is shown below.

(1) Implementing Image De-raining using torch7

(2) Training and testing ID-CGAN

(3) Performing evaluation experiment

The following report is organized as follows. Section Ⅲ describe the network ID-CGAN proposed by the paper. Section Ⅳ shows the implementation of ID-CGAN. Section Ⅴ display the simulation results of the network. Section Ⅵ summarize the final conclusion of the project.

Ⅲ. The proposed network

The proposed network ID-CGAN is composed of three important parts (generator, discriminator and perceptual loss function) that serve distinct purposes. Similar to traditional GANs, the author have two subnetworks: a generator sub-network G and a discriminator subnetwork D. The generator sub-network G is a symmetric deep CNN network with appropriate skip connections as shown in the top part in Figure 2. Its primary goal is to synthesize a de-rained image from an image that is degraded by rain (input rainy image). The discriminator sub-network D, as shown in the bottom part in Figure 2, serves to distinguish ”fake” de-rained image synthesized by the generators from corresponding ground truth ‘real’ image. It can also be viewed as a guidance for the generator G. Since GANs are known to be unstable to train which results in artifacts in the output image synthesized by G, the author define a refined perceptual loss functions to address this issue.

A. GAN objective function

In order to learn a good generator G so as to fool the learned discriminator D and to make the discriminator D good enough to distinguish synthesized de-rained image from real ground truth, the proposed method alternatively updates G and D following the structure proposed in [3], [4]. Given an input rainy image x and a random noise vector z, conditional GAN aims to learn a mapping function to generate output image y by solving the following optimization problem:

$$\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{data(\mathbf{x})},\mathbf{z}\sim p_{(\mathbf{z})}}[\log(1-D(\mathbf{x},G(\mathbf{x},\mathbf{z})))]+ \\ \mathbb{E}_{\mathbf{x}\sim p_{data(\mathbf{x},\mathbf{y})}}[\log D(\mathbf{x},\mathbf{y}))]. \tag{1}$$

B. Generator with symmetric structure

As the goal of single image de-raining is to generate pixel level de-rained image, the generator should be able to remove rain streaks as much as possible without loosing any detail information of the background image. So the key part lies in designing a good structure to generate de-rained image.

Existing methods, such as sparse coding based methods [5], [6], neural network-based methods [7] and CNN-based methods [8] have all adopted a symmetric structure. For example, sparse coding-based methods use a learned or pre-defined synthesis dictionaries to decode the input noisy image into sparse coefficient map. Then another set of analysis dictionaries are used to transfer the coefficients to desired clear output. Usually, the input rainy image is transferred to a specific domain for effective separation of background image and undesired component (rain-streak). After separation, the background image (in the new domain) has to be transferred back to the original domain which requires the use of a symmetric process. Therefore, the author also adopt a symmetric structure to form our generator subnetwork. Similar to traditional low-level vision CNN frameworks, the generator G directly learns an end-to-end mapping from input rainy image to its corresponding ground truth.

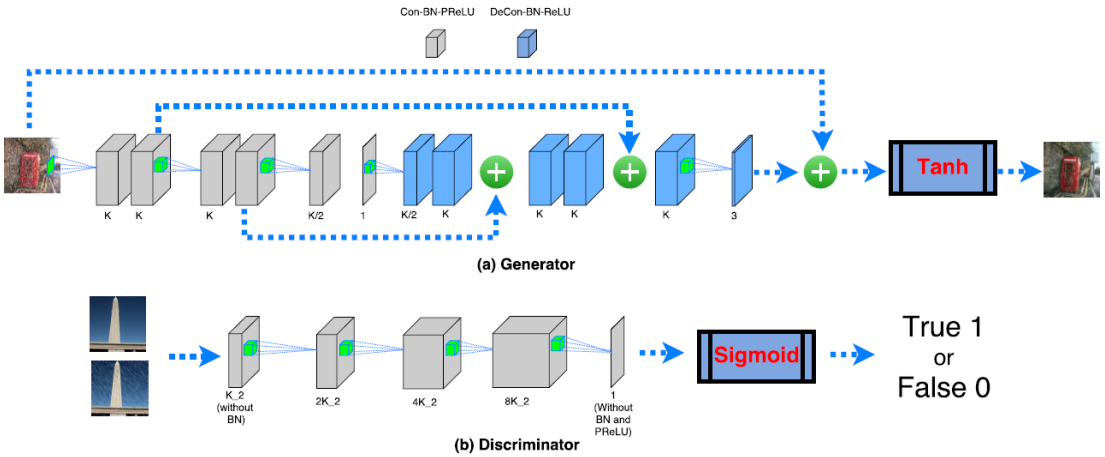

Figure 2. An overview of the proposed ID-CGAN method

The proposed generator G with a symmetric structure is shown in the top part of Figure 2. A set of convolutional layers (along with batch normalization and PReLU activation) are stacked in the front which act as a learned feature extractor or semantic attributes extractor. Then, three shrinking layers are stacked in the middle part serving for better computational efficiency. These three shrinking layers can be also regarded as performing linear combination within the learned features [9]. These are followed by a stack of deconvolutional layers (along with batch normalization and ReLU activation function). Note that the deconvolutional layers are a mirrored version of the forward convolutional layers. For all layers, the author use a stride of 1 and pad appropriate zeros to maintain the dimension of each feature map to be the same as that of the input. To make the network efficient in training and have better convergence performance, the author involve symmetric skip connections into the proposed generator sub-network, similar to [8]. The generator network is as follows:

$$CBP(K)\text{-}CBP(K)\text{-}CBP(K)\text{-}CBP(K)\text{-}CBP(K/2)\text{-}CBP(1)\text{-}$$
$$DBR(K/2)\text{-}DBR(K)\text{-}DBR(K)\text{-}DBR(K)\text{-}DBR(K)\text{-}DBR(3)\text{-}Tanh$$

where $CBP(K)$ is a set of K-channel convolutional layers followed by batch normalization and PReLU activation, $DBR(K)$ is a set of K-channel deconvolutional layers followed by batch normalization and ReLU activation. Skip connections are added via every two skips, as shown in Figure 2.

C. Discriminator

Following the structure that was proposed in [10], the author use convolutional layer with batch normalization and PReLU activation as a basis throughout the discriminator network. Once calculate the learned feature from a set of these Conv-BN-PReLU, a sigmoid function is stacked at the end to map the output to a probability score normalized to [0,1]. The proposed discriminator sub-network D is shown in the bottom part of Figure 2. The structure of the discriminator sub-network is as follows:

$$CB(K_2)\text{-}CBP(2K_2)\text{-}CBP(4K_2)\text{-}CBP(8K_2)\text{-}C(1)\text{-}Sigmoid$$

where, $CB(K2)$ is a set of $K2$ channel convolutional layers followed by batch normalization and $C(1)$ is a set of 1-channel convolutional layers.

D. Redefined perceptual loss

To ensure that the results have good visual and quantitative scores along with good discriminatory performance, the author propose a new refined loss function. Specifically, the author combine pixel-to-pixel Euclidean loss, perceptual loss [11] and adversarial loss together with appropriate weights to form our new refined loss function. The new loss function is then defined as follows:

$$L_{RP} = L_E + \lambda_a L_A + \lambda_p L_P, \qquad (2)$$

where L_A represents adversarial loss (loss from the discriminator D), L_P is perceptual loss and L_E is normal per-pixel loss function such as Euclidean loss. Here, λ_p and λ_a are pre-defined weights for perceptual loss and adversarial loss, respectively. If both λ_p and λ_a are set to be 0, then the network reduces to a normal CNN configuration, which aims to minimize only the Euclidean loss between output image and ground truth. If λ_p is set to 0, then the network reduces to a normal GAN. If λ_a set to 0, then the network reduces to the structure proposed in [11].

The three loss functions L_P , L_E and L_A are defined as follows. Given an image pair $\{x, y_b\}$ with C channels, width W and height H (i.e. $C \times W \times H$), where x is the input image and y_b is the corresponding ground truth, the per-pixel Euclidean loss is defined as:

$$L_E = \frac{1}{CWH} \sum_{c=1}^{C} \sum_{x=1}^{W} \sum_{y=1}^{H} \|\phi_E(\mathbf{x}^{c,w,h}) - (\mathbf{y}_b^{c,w,h})\|_2^2, \qquad (3)$$

where ϕ_E is the learned network G for generating the de-rained output. Suppose the outputs of certain high-level layer are with size $C_i \times W_i \times H_i$. Similarly, the perceptual loss is defined as:

$$L_P = \frac{1}{C_i W_i H_i} \sum_{c=1}^{C_i} \sum_{w=1}^{W_i} \sum_{h=1}^{H_i} \|V(\phi_E(\mathbf{x}^{c,w,h})) - V(\mathbf{y}_b^{c,w,h})\|_2^2, \qquad (4)$$

where V represents a non-linear CNN transformation. Similar to the idea proposed in [11], the author aim to minimize the distance between high-level features. In the proposed method, the author compute the feature loss at layer relu2_2 in VGG-16 model [12].

Given a set of N de-rained images generated from the generator $\{y_i\}_{i=1}^N$, the entropy loss from the discriminator to guide the generator is defined as:

$$L_A = -\frac{1}{N} \sum_{i=1}^{N} \log(D(\mathbf{y}_i)). \qquad (5)$$

Ⅳ. The implementation of ID-CGAN

The proposed network is implemented by torch7 using cuda8.0 and cudnn5.0. The parameters of training ID-CGAN is shown in Table 1.

Generator Filters in the First Conv Layer	64
Discriminator Filters in the First Conv Layer	48
Iteration	2000
Initial Learning Rate for Adam	0.0002
Momentum Term of Adam	0.5

Batch Size	7
Weight on Perceptual Loss Term in Objective	150
weight on MSE Term in Objective	150

Table 1. Parameters for training

Ⅴ. Experiment Results

A. Datasets

The training set consists of a total of 500 images, where 300 images are randomly chosen from the first 800 images in the UCID dataset [13] and 200 images are randomly chosen from the BSD-500's training set [14]. The test set consists of a total of 100 images, where 50 images are randomly chosen from the last 500 images in the UCID dataset [13] and 50 images are randomly chosen from the test-set of the BSD-500 dataset [14]. After the train and test sets are created, rain-streaks are added to these images by following the guidelines mentioned in [15] using Photoshop4. The example is shown in Figure 3. It is ensured that rain pixels of different intensities and orientations are added to generate a diverse training and test set. Note that the images with rain form the set of observed images and the corresponding clean images form the set of ground truth images. All the training and test samples are resized to 256×256.

Figure 3. The original image and image added rain

In order to demonstrate the effectiveness of the proposed method on real-world data, a dataset of 50 rainy images was downloaded from the Internet. The images are chosen carefully to ensure that the images collected were diverse in terms of content as well as intensity and orientation of the rain pixels.

B. Results

Some of the tested images are chosen to show the effect of ID-CGAN, as shown in Figure 4 and Figure 5.

Figure 4. The original image and image after de-raining

Figure 4. The original image and image after de-raining

From the example, we can see that most of the rain streak is removed from the image. However, some large rain streaks are still remained in the image. It shows that ID-CGAN is more capable of removing scattered and small rain streaks. There are mainly two reasons causing this drawback. First, the dataset is limited, so it does not contain various types and sizes of rain streaks. Second, the loss function remained to be improved to take large rain streaks into consideration. Therefore, future research would focused on the improvement of loss function as well as the structure of network.

Ⅵ. Conclusions

In the project, the proposed ID-CGAN is implemented. For improved stability in training and reducing artifacts introduced by GANs in the output images, the use of a new refined loss function is introduced in the GAN optimization framework. By performing the experiment of image de-raining, the results shows that ID-CGAN is capable of removing most of rain streaks, while some large streaks are remained, which is the focus of future research.

During this project, I have the chance to have a deeper understanding of GAN, such as its

different structures, different loss functions and their effects on the results. Meanwhile, I have learned the use of torch and other deep learning platforms, which is essential for deep learning researches. Thanks the professors and teaching assistants for providing such an opportunity.

References

[1] https://deeplearning4j.org/generative-adversarial-network

[2] He Zhang, Vishwanath Sindagi, Vishal M. Patel, “Image De-raining Using a Conditional Generative Adversarial Network”

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” arxiv, 2016.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[5] J.-L. Starck, M. Elad, and D. L. Donoho, “Image decomposition via the combination of sparse representations and a variational approach,” Image Processing, IEEE Transactions on, vol. 14, no. 10, pp. 1570– 1582, 2005.

[6] G. Peyr´e, J. Fadili, and J.-L. Starck, “Learning the morphological diversity,” SIAM Journal on Imaging Sciences, vol. 3, no. 3, pp. 646– 669, 2010.

[7] J. Xie, L. Xu, and E. Chen, “Image denoising and inpainting with deep neural networks,” in Advances in Neural Information Processing Systems, 2012, pp. 341–349.

[8] X.-J. Mao, C. Shen, and Y.-B. Yang, “Image denoising using very deep fully convolutional encoder-decoder networks with symmetric skip connections,” arXiv preprint arXiv:1603.09056, 2016.

[9] C. Dong, C. C. Loy, and X. Tang, “Accelerating the super-resolution convolutional neural network,” in European Conference on Computer Vision. Springer, 2016, pp. 391–407.

[10] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015

[11] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” arXiv preprint arXiv:1603.08155, 2016.

[12] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[13] G. Schaefer and M. Stich, “Ucid: an uncompressed color image database,” in Electronic Imaging 2004. International Society for Optics and Photonics, 2003, pp. 472–480.

[14] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 5, pp. 898–916, 2011.

[15] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, “Clearing the Skies: A deep network architecture for single-image rain removal,” ArXiv eprints, Sep. 2016.