BLIND MOTION DEBLURRING VIA INCEPTIONRESDENSENET BY USING GAN MODEL

[1]Ze-Ming Chen, [2]Long-Wen Chang

Department of Computer Science, National Tsing Hua University, Taiwan
[1]w103127@gmail.com, [2]lchang@cs.nthu.edu.tw

ABSTRACT

Deblurring from a motion blurred image has been studied for some times. Recently, convolution neural network (CNN) has been used widely and it can be used on finding the blur kernel or the latent sharp edge of a blurred image. In recent years, the generative adversarial network (GAN) performs well on style transformation. We consider that a deblurring problem as a style transformation problem. We focus on improving the DeblurGAN's generator, which is the state-of-the-art of the deblurring method and present a new kind of block which combined inception block, residual block and dense block to do deblurring from motion blur. By using the conception of DenseNet which can avoid overfitting. The improved DeblurGAN presents better in both structural similarity measure and by visual effect.

1. INTRODUCTION

Image deblurring has been studied in the field of image processing and computer vision. The problem can be divided into blind deblurring and non-blind deblurring. In general, we can use the degradation model of image deconvolution which is defined as:

$$b = k * l + n, \quad (1)$$

where b is the blurred image, l is the latent sharp image, k is the blur kernel, n is additive noise and $*$ denotes a convolution operator. If the blur kernel is known, then it is a non-blind deblurring problem; otherwise, it is a blind deblurring problem. When solving the non-blind deblurring problem, we usually assume that the blur kernel is uniform. That is, the whole image is blurred by the same blur kernel. By deconvolution, we can restore the blurred image to a sharp latent image. If the blur kernel is not uniformed, for example, the background and the objects both move in their own ways at the same time while we shot the picture. The blur is caused by different trajectories. It is called motion blur. We can separate the blur image into many image patches. In each blurred image patch, we estimate its blur kernel and restore the latent image patch by the deconvolution of the blurred image patch. By combining all the restored image patches together, we get the final latent image. Whether the result is good or bad is dependent on the patches we divide.

Deep learning has been widely studied recently. Many deblurring methods by using Convolutional Neural Network (CNN) have been proposed. Yan et al. [32] used CNN to estimate the blur kernel, by this kernel we can use traditional deconvolution method to get the restored image. Xu et al. [14] input the blur image and output the obvious edge for blur kernel estimation via deep learning. Some methods [5,7,10,13] present an end-to-end CNN architecture for directly recover the image from the blurred image without estimating the blur kernel. In this paper, we introduce a Generative Adversarial Network (GAN) proposed by Ian Goodfellow [15] to deal with this ill-posed problem. We also build a new block which called InceptionResDense (IRD) block to add in our GAN. The experimental results show that our method performs better than the ResNet [3]. The estimated latent image is very sharp with very high quality of detail structure in the recovered image.

2. RELATED WORK

In recent years, many deblurring methods [10,11,12,13,14] used convolutional neural network to replace priors based methods [8,9]. Chakrabarti [11] used a neural network to predict the Fourier coefficients of a filter by blurred image patches. He used these coefficients to generate a new input with less blur effect. Then he used the new input to do global kernel estimation and non-blind deconvolution to get estimated latent image. Xu et al. [14] proposed a deep CNN to extract sharp edges from a blurred image. The model consisted of two stages. The first is to suppress extraneous details and the second is to enhance sharp edges. Noroozi et al. [10] proposed DeblurNet which is a multiscale CNN architecture to predict a sharp image from a blurred image directly. In 2014, the generative adversarial network was introduced by Ian Goodfellow et al. [15]. His main idea is to build two competitive networks: a generator and a discriminator. The generator generates the image we want. The generator needs to generate images which can fool discriminator, and discriminator should not be fooled by the generated images. The game between the generator G and the discriminator D is the $minmax$ objective:

$$min_G max_D \, E_{x\sim P_r}[log(D(x))] + E_{\bar{x}\sim P_G}[log(1 - D(\bar{x}))], \quad (2)$$

where P_r is the distribution of real data, P_G is the distribution of the generated data $\bar{x} = G(z)$, which means that $\bar{x}$ is generated by the generator G with a random sample z from a simple noise distribution P_z.

In [16], it mentioned that the training of this model would suffer problems such as mode collapse and vanishing gradients. These problems may be caused by minimizing the value function in GAN, which is equal to minimize the

Jensen-Shannon (JS) divergence between the data and model distributions. Arjovsky et al. [17] proposed Wasserstein GAN (WGAN) which uses Earth-Mover distance to replace the JS divergence because Earth-Mover distance provides more usable gradients everywhere. The value function for WGAN is written as:

$$min_G max_D \; E_{x \sim P_r}[D(x)] - E_{\bar{x} \sim P_G}[D(\bar{x})], \quad (3)$$

where D is the set of 1-Lipschitz functions, The notation $x \sim P_r$ denotes the input x is a sample from real data distribution P_r and $\bar{x} \sim P_G$ denotes input $\bar{x}$ is a sample from the generated data distribution P_G. The discriminator D approximates the distance between x and $\bar{x}$. To enforce Lipschitz constraint in WGAN, Arjovsky et al. added weight clipping to $[-c, c]$. This technique may lead to optimization difficulties. Gulrajani et al. [18] proposed to add a gradient penalty term to the value function in equation (3) to enable stable training of wide variety of WGAN architectures:

$$\lambda \, E_{\tilde{x} \sim P_{penalty}}[(||\nabla_{\tilde{x}} D(\tilde{x})|| - 1)^2], \quad (4)$$

where $P_{penalty}$ is the penalty distribution between P_r and P_G, $\tilde{x} \sim P_{penalty}$ denotes sampling a point $\tilde{x}$ from $P_{penalty}$. $\nabla_{\tilde{x}} D(\tilde{x})$ is the gradient between input $\tilde{x}$ and output $D(\tilde{x})$.

In many papers, GANs have been used on different image-to-image translation problems, like style transfer [19], super resolution [20], semantic segmentation [21] and others. Isola et al. [22] present conditional GAN (cGAN) which is known as pix2pix to learn a mapping function from an observed image x and random noise vector z, to output the image y. That is,

$$G(x, z) \rightarrow y. \quad (5)$$

Some researchers also use cGAN's characteristic to build a generator to generate sharp images through blur images. Ramakrishnan et al. [6] proposed a novel deep filter based on GAN architecture and integrated with DenseNet architecture and global skip connection in order to deal with the problem of the relative motion between the camera and the object in 3D space induces a spatially varying blurring effect over the entire image. Kupyn et al. [1] presented DeblurGAN which is based on cGAN and ResNet architecture for motion deblurring. It achieves state-of-the-art in structure similarity (SSIM) and visual quality.

3. PROPOSED METHOD

We hope each layer of our generator can learn both sparse and non-sparse features of the blurred image. Also, we want our features can flow through the generator without changing the width and height. By the thoughts above, we can keep more detail information and use them to restore sharper images.

3.1. Network Architecture

Our GAN architecture includes one generator and one discriminator as shown in Figure 1. The generator generates a fake image to let the discriminator mistreat that the fake image is a realistic image. We mainly concentrate on the design of the generator of our GAN. In our generator shown in Figure 2, the width and the height of the feature maps will not be changed when flowing through the generator. In Figure 2 we input a blurred image of size 256 x 256 x 3 and we want the next convolutional layer to output 256 x 256 x 64 feature maps with 64 kernels of 7 x 7 x 3.

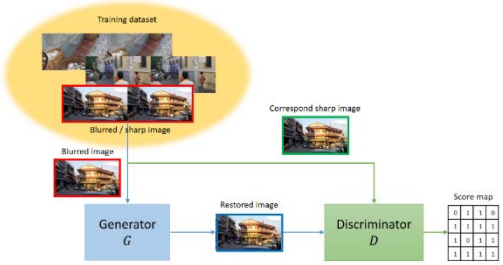


Figure 1. Our GAN architecture.

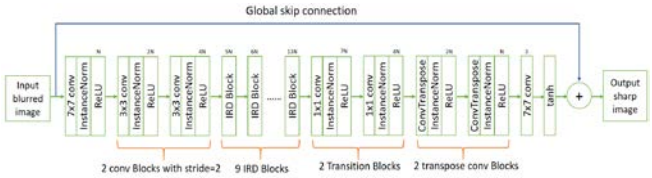


Figure 2. Our generator architecture.

Like Ramakrishnan et al. [6] we preserve the dimension of the information to prevent the network from generating checkerboard artifacts which usually be found in networks which use a pyramid structural architecture. We combined the inception-resnet concept of GoogLeNet [2, 23, 24 25] and densely connected networks proposed by Huang et al. [4] to build a new block. We called this block an IRD block. As shown in Figure 2 our GAN includes two convolution blocks with stride 2, nine IRD blocks, two transition blocks, two transposed convolution blocks and a global skip connection.

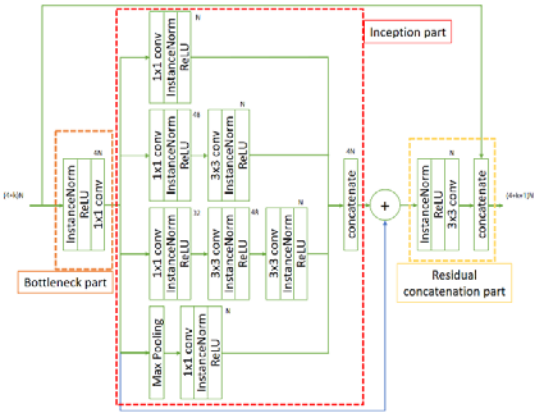


Figure 3. The proposed InceptionResDense (IRD) block.

We define a parameter N as the constant number of feature maps. The value of N is 64 in our experiment. Each convolution block consists of a 3 x 3 convolution layer, an instance normalization layer [26], and a ReLU [27] activation. We use two convolution blocks. The first one outputs 2N

(128) feature maps, and the second one outputs 4N (256) feature maps which is also the basic input of our first IRD block.

Our IRD block shown in Figure 3 can be divided into three parts: bottleneck, inception part and residual concatenation part. The bottleneck part has a normalization layer, a ReLU activation and an 1x1 convolution layer. Its main purpose is to choke the input feature maps of (4+k)N into 4N in order to protect parameters and data memories in deeper layers of the network. The inception part consists of four branches. The first branch contents of an 1x1 convolution layer. The second branch is made by one 1x1 convolution layer and one 3x3 convolution layer. The third branch is built by one 1x1 convolution layer and two 3x3 convolution layers. The fourth branch comprises of a max pooling and an 1x1 convolution layer. We add an instance normalization layer and a ReLU activation after each convolution layer in the inception part. Each branch inputs 4N feature maps and outputs N maps. The output of these four branches will be concatenated into 4N feature maps and allow us to get sparse and non-sparse features. Through these branches we can not only make our network wider, but also gain the network adaptability to deal with different kinds of motion kernels. We also add a residual connection which is the same concept as ResNet [3] to build directly information connectivity. Our experiments show that it performs well and converges faster when we add the residual connection. In the residual concatenation part, we use one 3x3 convolution layer to transfer the feature maps into N and concatenate them behind the input feature maps as the output feature maps.

The idea of the transition block is to transfer the output feature maps of the 9th IRD fro, 13N to 4N. We use two transition blocks to do this job. The first one reduces the feature maps from 13N to 7N. The second one reduces 7N into 4N. Each transition block is included an 1x1 convolution layer, an instance normalization layer and a ReLU activation. The transposed convolution is equal to deconvolution. We use these two transposed convolution blocks to do up-sampling from the features of 2nd transition block. The transposed convolution block is established by one convolution transpose layer, one instance normalization layer and one ReLU layer.

We introduce the global skip connection like many other methods do [1, 6, 12, 29]. Due to long-term memory effect, the global skip connection makes the generator easier to generate sharp boundaries at correct locations. It can be seen as:

$$I_{Sharp} = I_{Blurred} + G(I_{Blurred}), \qquad (6)$$

where $I_{Blurred}$ is the input blurred image, I_{Sharp} is the output image and $G(I_{Blurred})$ denotes the generator G generate the correction residual image to restore the input blurred image to the sharper one. The global skip connection makes training faster and generates a better latent image.

In this work, we defined our discriminator as DeblurGAN [1], which use PatchGAN [22, 33] as network architecture. The discriminator network is shallow that it memorizes the easier task of classification. All the convolution layers exclusive of the last layer are stacked with instance normalization layer and Leaky ReLU [28] with negative slope is 0.2. We input a generated image or a latent image and output a score map. Each pixel of the score map is between 0 and 1 means the probability of whether the corresponding image patch is real or it is artificially generated. We also introduce Wasserstein GAN [17] in equation (3) with Gradient Penalty [18] in equation (4) to train our network.

3.2. The Loss Function

Our loss function is combined with perceptual loss, WGAN loss and $l1$-loss to train our generator:

$$L = L_{WGAN} + \lambda_1 L_{per} + \lambda_2 L_{l1}, \qquad (7)$$

where L_{WGAN}, L_{per}, L_{l1} are explained below ,λ_1 is 100 and λ_2 is 170 in our experiments.

The WGAN value function in equation (3) describes the game between generator and discriminator. When training the generator, we can rewrite the equation (3) as equation (8) below:

$$L_{WGAN} = -D_{\theta_D}(G_{\theta_G}(I_{Blurred})), \qquad (8)$$

where D_{θ_D} is the discriminator D with its own parameters θ_D, G_{θ_G} is the generator G with its own parameter θ_G, and $I_{Blurred}$ is the blurred image. We input the blurred image $I_{Blurred}$ to the generator G_{θ_G} to generate a latent image. The discriminator evaluates the latent image to get a score map. We want each pixel of the score map to be as closer as possible to 1 by using the minimization function in equation (8).

We need some structural knowledge when training the network. Johnson et al. [30] proposed a perceptual loss function which can fit our need. The function is a kind of L2-loss which is the Euclidean difference between deep convolutional activations of the generated image by the generator G and the ground truth image. This loss term can be computed by VGG19 as:

$$L_{per} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}(I_{Sharp})_{x,y} - \phi_{i,j}\left(G_{\theta_G}(I_{Blurred})\right)_{x,y} \right)^2, \qquad (9)$$

where $W_{i,j}$, $H_{i,j}$ are the width and height of the feature maps, $\phi_{i,j}$ is the feature map obtained by j-th convolution before the i-th max-pooling layer forward pass through the pre-trained VGG19 network. The generated image $G_{\theta_G}(I_{Blurred})$ and I_{Sharp} are inputted to the VGG19 network individually and we calculate their Euclidean difference by their feature maps before the 14th convolution layer in the VGG19 network.

When image blurred more heavily, we cannot restore the latent image with true sharp edge well. For example, the license plate of the restored car image has difficulty to identify. It seems to generate some extra-lines to pretend the license plate numbers in our GAN architecture without this

loss term. We simply add a L1 loss by the input generated image and ground truth to constraint the wrong lines. The L1-loss function is:

$$L_{l1} = \left| I_{Sharp} - G_{\theta_G}(I_{Blurred}) \right|. \quad (10)$$

where I_{Sharp} is the ground truth, $G_{\theta_G}(I_{Blurred})$ is a sharp image generated by the generator G_{θ_G} from the given blured image $I_{Blurred}$.

3.3 Training Details

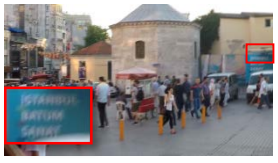
We use PyTorch to implement our GAN model. All the experiments were performed on a single GTX-1080-Ti GPU with GDDR5X 11G and i7-7700 processor with 16G RAM. The model was trained on a random crop of size 256x256 from GoPro dataset images and MS-COCO dataset images which are blurred by the method described in [1]. We totally use 3217 images for training and 1111 images from GoPro dataset for testing. For the optimization we follow the approach of [1, 14] to do 5 gradient descent steps on D_{θ_D}, then 1 step on G_{θ_G} by Adam gradient descent method [31]. The learning rate is set to 10^{-4} and linearly decay the rate to 0 after 150 epochs. The batch size is set to 1 that allows us to fit bigger image in the memory. We apply dropout = 0.2 after every IRD blocks by the idea of [1 ,22]. Our model needs 3 days for training.

4. EXPERIMENTAL RESULTS

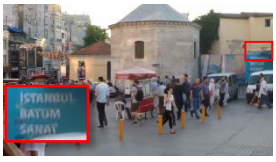
We evaluate our method on GoPro dataset [35]. We compare our experimental results with Kupyn et al. [1] and Ramakrishnan et al. [6] which are the state-of-the-art deblurring method by using GAN. We use our training dataset to re-train both [1,6]. In Table 1, we show the average peak signal to noise ratio (PSNR) and average structural similarity measure (SSIM) on the GoPro testing dataset of 1111 images for each method. Most of our results perform better than those of state-of-the-art methods [1,6]. In Figures 4 and 5, we show the comparison results of GoPro testing dataset of 1111 images. We can see that our restored image has less ringing artifact and can easily recognize the texts compared to [1,6].

Table 1. Comparisons of PSNRs and SSIMs with [1,6].

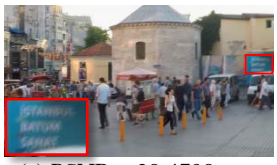
Method	PSNR	SSIM
Kupyn et al. [1]	27.1739	0.8338
Ramakrishnan et al. [6]	25.8340	0.7898
Our method	27.7883	0.8472



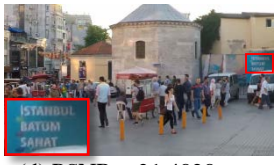
(a) Input blurred image



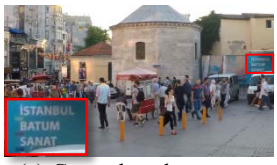
(b) PSNR = 30.3490, SSIM = 0.9077



(c) PSNR = 28.4798, SSIM = 0.8876

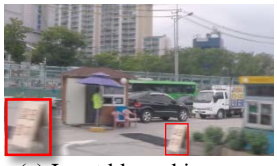


(d) PSNR = 31.4828, SSIM = 0.9217

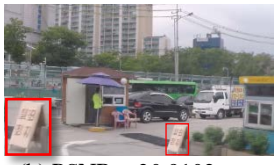


(e) Ground truth

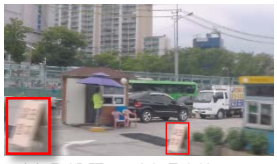
Figure 4. (a) Input blurred image. (b) Kupyn et al. [1]. (c) Ramakrishnan et al. [6] (d) Ours. (e) Ground truth.



(a) Input blurred image

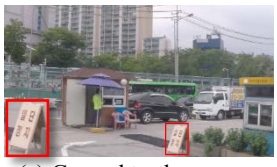


(b) PSNR = 30.9102, SSIM = 0.9014



(c) PSNR = 29.7662, SSIM = 0.8907

(d) PSNR = 31.1337, SSIM = 0.9074



(e) Ground truth

Figure 5. (a) Input blurred image. (b) Kupyn et al. [1]. (c) Ramakrishnan et al. [6] (d) Ours. (e) Ground truth.

5. CONCLUSIONS

In this paper, we use WGAN to train a generator to generate a latent image with a blurred one. We add the perceptual loss function and the L1-loss function into the total loss function. The proposed IRD block uses the concept of the GoogLeNet [2, 23, 24, 25] to get sparse and non-sparse features of the blurred image. Also, our WGAN combines the ResNet [3] to enhance the features and the DenseNet [4] to make sure the features can flow through the generator. Our experimental results out-perform other deblurring methods using GAN [1,6].

6. REFERENCES

[1] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, J. Matas. "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks," *arXiv:1711.07064v4*, 2018.

[2] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *arXiv: 1602.07261v2*, 2016.

[3] K. He, X. Zhang, S. Ren, J. Sun. "Deep Residual Learning for Image Recognition," *arXiv: 1512.00385*, 2015.

[4] G. Huang, Z. Liu, L. van der Maaten. "Densely Connected Convolutional Networks," *arXiv: 1608.06993*, Jan. 2018.

[5] L. Wang, Y. Li, S. Wang. "DeepDeblur: Fast one-step blurry face images restoration," *arXiv: 1711.09515v1*, 2017.

[6] S. Ramakrishnan, S. Pachori, A. Gangopadhyay, S. Raman. "Deep Generative Filter for Motion Deblurring," *arXiv: 1709.03481v1*, 2017.

[7] T. M. Nimisha, A. K. Singh, A. N. Rajagopalan. "Blur-Invariant Deep Learning for Blind-Deblurring," *IEEE International Conference on Computer Vision (ICCV)*, pp. 4762-4770, Oct 2017.

[8] D. Krishnan, T. Tay, R. Fergus. "Blind Deconvolution Using a Normalized Sparsity Measure," *In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 233-240, June 2011.

[9] J. Pan, Z. Hu, Z. Su, M. Yang. "Deblurring Text Images via L0-Regularized Intensity and Gradient Prior," *In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 2901-2908, June 2014.

[10] M. Noroozi, P. Chandramouli, P. Favaro. "Motion Deblurring in the Wild," *arXiv: 1701.01486v2*, Jan 2017.

[11] A. Chakrabarti. "A Neural Approach to Blind Motion Deblurring," *arXiv: 1603.04771v2*, Aug 2016.

[12] H. Son, S. Lee. "Fast Non-blind Deconvolution via Regularized Residual Networks with Long/Short Skip-Connections," *IEEE International Conference on Conference Photography (ICCP)*, pp. 1-10, May 2017.

[13] M. Hradiš, J. Kotera, P. Zemcík, F. Šroubek. "Convolutional Neural Networks for Direct Text Deblurring," *In Proceedings of the British Machine Vision Conference (BMVC)*, pp. 6.1-6.13, Sept 2015.

[14] X. Xu, J. Pan, Y. Zhang, M. Yang. "Motion Blur Kernel Estimation via Deep Learning," *IEEE Transaction on Image Processing*, pp. 194-205, Sept 2017.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. "Generative Adversarial Nets," *In Advances in Neural Information Proceeding Systems (NIPS)*, pp. 2672-2680, 2014.

[16] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen. "Improved Techniques for Training GANs," *arXiv: 1606.03498*, June 2016.

[17] M. Arjovsky, S. Chintala, L. Bottou. "Wasserstein GAN," *arXiv: 1701.07875*, Dec 2017.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, Aaron Courville. "Improved Training of Wasserstein GANs," *arXiv: 1704.00028*, Dec 2017.

[19] J. Zhu, T. Park, P. Isola, A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *arXiv: 1703.10593*, Feb 2018.

[20] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," *arXiv: 1609.04802*, May 2017.

[21] P. Luc, C. Couprie, S. Chintala, J. Verbeek. "Semantic Segmentation using Adversarial Networks," *arXiv: 1611.08408*, Nov 2016.

[22] P. Isola, J. Zhu, T. Zhou, A. A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks," *arXiv: 1611.07004*, Nov 2017.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. "Going deeper with convolutions," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1-9, June 2015.

[24] S. Ioffe, C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv: 1502.03167*, Mar 2015.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna. "Rethinking the Inception Architecture for Computer Vision," *arXiv: 1512.00567*, Dec 2015.

[26] D. Ulyanov, A. Vedaldi, V. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization," *arXiv: 1607.08022*, Nov 2017.

[27] V. Nair, G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines," *In Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

[28] A. L. Maas, A. Y. Hannun, A. Y. Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models," *In Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013.

[29] Z. Shen, W. Lai, T. Xu, J. Kautz, M. Yang. "Deep Semantic Face Deblurring," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] J. Johnson, A. Alahi, Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," *arXiv: 1603.08155*, Mar 2016.

[31] D. P. Kingma, J. L. Ba. "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," *Conference paper at the 3rd International Conference for Learning Representations (ICLR)*, 2015.

[32] R. Yan and L. Shao. "Blind Image Blur Estimation via Deep Learning," *IEEE Transaction on Image Processing*, vol. 25, no. 4, pp. 1910-1921, Apr 2016.

[33] C. Li and M. Wand. "Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks," *arXiv: 1604.04382v1*, Apr 2016.

[34] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, S. Harmeling. "Recording and playback of camera shake: benchmarking blind deconvolution with a real-world database," *In ECCV*, pp. 27-40, 2012.

[35] S. Nah, T. H. Kim, K. M. Lee. "Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.