IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Single Image Deblurring Algorithm for Non-uniform Motion using GAN with KERAS

BY

Shuraiq Rahman Khan(18554)

Gaddam Satwik Reddy(18520)

Dongsarwar Nikitha(18517)

Under the supervision of

Dr.Talagondapati Veeraiah

ABSTRACT

Taking perfect photos under dim/improper lighting conditions using a hand-held camera is a challenging task. When the camera is set to a longer exposure time, the image might be blurred due to camera shake. On the other hand, Handheld cameras often have insufficient light and require excessive exposure times , which cause blurring . When the light is too dim or too bright the image may suffer from motion blur artifacts. By combining information extracted from both blurred and noisy images, however, we show in this paper how to produce a high quality image by reducing the blur.

Our approach is image deblurring with the help of Deblur GAN, an end-to-end learned method for motion deblurring. The learning is based on a conditional GAN an end-to-end method for non-uniform motion deblurring and the content loss . Deblur GAN achieves state-of-the art performance both in the structural similarity measure and visual appearance. The quality of the deblur-ring model is also evaluated in a novel way on a real-world problem – object detection on (de-)blurred images.We expect that the proposed method can achieve sufficiently good deblurring of a single non-uniform blur image.

1.Introduction:

This work is on non-uniform motion image deblurring of a single photograph. Significant advancement has been recently achieved in related areas of image super-resolution by applying generative adversarial networks(GANs) .

With the popularization of cheap camera devices deblurring can now be integrated in nonessential desktop or mobile devices for recovering personal movies, photographs or audio recordings.

GANs are known for the ability to conserve texture details in images, generate solutions that are almost identical to the real time data .GANs can learn the internal representations of any data. Also GANs can learn chaotic and complicated distributions of data .Stimulated by the recent work on image super-resolution and image-to-image translation by generative adversarial net-works(GAN's) , we treat deblurring as a special case of such image-to-image translation. We present Deblur GAN – an approach based on conditional generative adversarial net-works and a multi-component loss function. In our work we extract two losses Wasserstein loss and perceptual loss . This encourages solutions which are perceptually hard to distinguish from real sharp images and allows to restore finer texture details than if using traditional MSE or MAE as an optimization target.

The common formulation of non-uniform blur model is the following:

$$B=k(M)*I+N$$

where B is a blurred image,k(M)are unknown blur kernels determined by motion field M.I is the sharp latent image ,$*$ denotes the convolution ,N is an additive noise .The family of deblurring problems is divided into two types :blind and non-blind deblurring.

In nonblind image deconvolution, the PSF is already known or computed therefore, the problem focuses on how to recover a blurred image by using a known M. TheWiener filter and Richardson-Lucy deconvolution are well-known non-blind deconvolution algorithms that are effective at image deblurring for cases in which the M is not complicated. The blind
problem entails recovering the unblurred image and estimating the M simultaneously .Image pair approaches have been proposed for image deblurring. Leveraging additional
images makes the blind deblurring problem more tractable.

We make three contributions. First, we propose a lossand architecture which obtain state-of-the art results in mo-tion deblurring, while being 5x faster than the fastest com-petitor. Second, we present a method based on randomtrajectories for generating a dataset for motion deblurringtraining in an automated fashion from the set of sharp im-age. We show that combining it with an existing datasetfor motion deblurring learning improves

results comparedto training on real-world images only. Finally, we present anovel dataset and method for evaluation of deblurring algo-rithms based on how they improve object detection results.

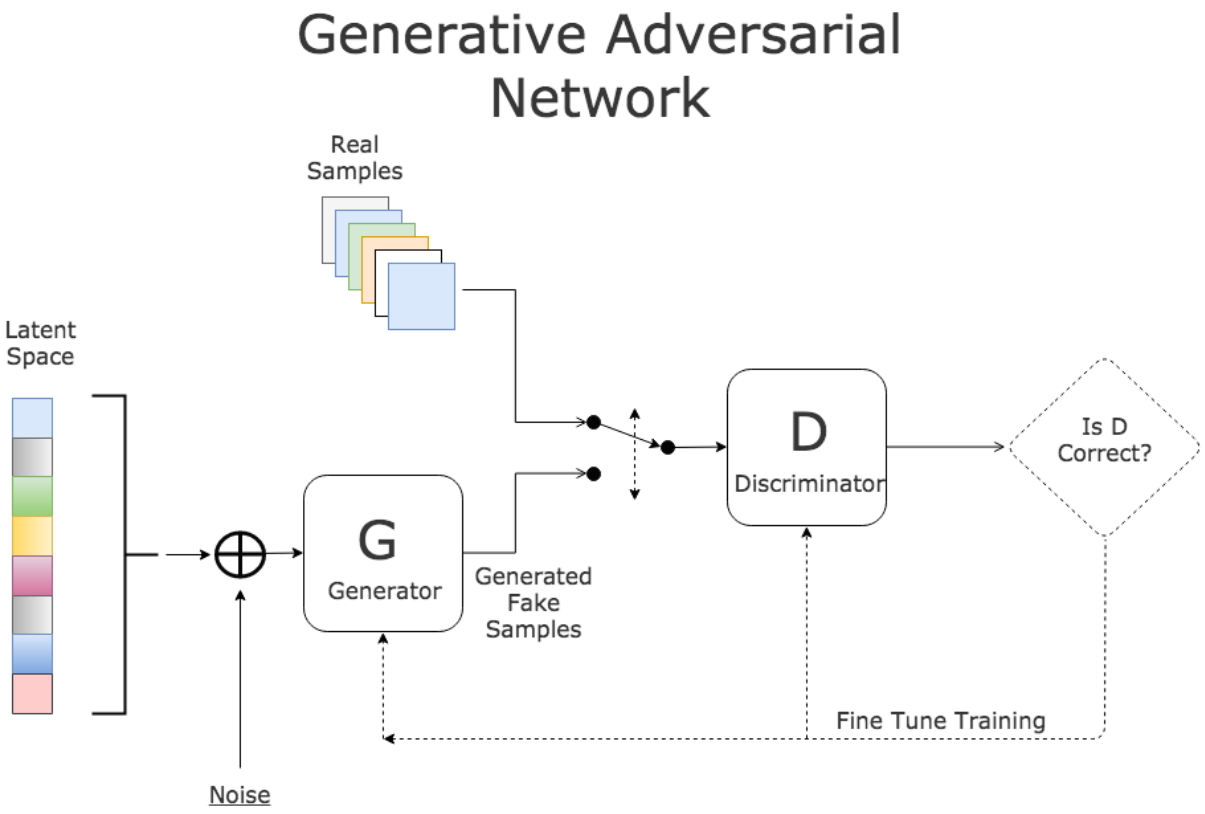


figure 1.1

2.Related works:

for non-uniform blind deblurring based on a parametrized geometric model of the blurring process in terms of the rotational velocity of the camera during exposure. Similarly Gupta et al. made an assumption that the blur is caused only by 3D camera movement. With the success of deep learning, over the last few years, there appeared some approaches based on convolutional neural networks(CNNs). Sun et al. use CNN to estimate blur kernel, Chakrabarti predicts complex Fourier coefficients of motion kernel to perform non-blind deblurring in Fourier space whereas Gong use fully convolutional network to move for motion flow estimation. All of these approaches use CNN to estimate the unknown blur function .

Recently ,a kernel-free end-to-end approaches by Nooroz i and Nah that uses multi-scale CNN to directly deblur theimage. Ramakrishnan et al. use the combination ofpix2pix framework and densely connected convolutional networks to perform blind kernel-free image deblurring. Such methods are able to deal with different sources of the blur.

2.1 Generative adversarial networks:

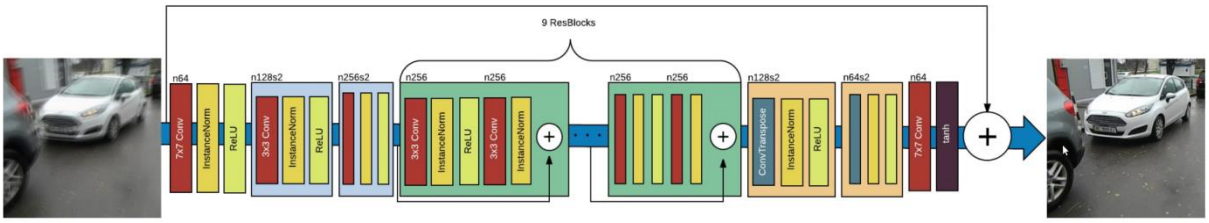


figure 1.2

GANs generate data that looks almost similar to original data. When you give GAN an image then a new version of the image is generated which looks almost similar to the given original image. GANs go into internal details of data and can easily interpret into various versions so it is helpful in doing machine learning work.

Good fellow et al introduced the idea of generative adversarial networks, GAN is an architecture to define a game between two competing networks: the discriminator and the generator .The generator receives noise as an input and generates a sample. A discriminator receives a real and generated sample and is trying to distinguish between them. The goal of the generator is to fool the discriminator by generating perceptually convincing samples that cannot be distinguished from the real one.

The relation between the generator G and discriminator D is the minimax objective :

$$\min_G \max_D \mathop{\mathbb{E}}_{x \backsim \mathbb{P}_r} [\log(D(x))] + \mathop{\mathbb{E}}_{\tilde{x} \backsim \mathbb{P}_g} [\log(1 - D(\tilde{x}))]$$

where Pr is the data distribution and Pg is the model distribution, defined by˜

x= G(z),z~P(z), the input z is a sample from a simple noise distribution. GANs are known for its ability to generate samples of good perceptual quality, however, training of vanilla version suffer from many problems such as mode collapse, vanishing gradients etc, as described in . Minimizing the value function in GAN is equal to minimizing the Jensen-Shannon divergence between the data and model distributions on x. Arjovsky et al. discuss the difficulties in GAN training caused by JS divergence approximation and propose to use the Earth-Mover (also called Wasserstein-1) distance W(q,p). The value function for WGAN is constructed using Kantorovich-Rubinstein duality :

$$\min_G \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{x \backsim \mathbb{P}_r} [D(x)] - \mathop{\mathbb{E}}_{\tilde{x} \backsim \mathbb{P}_g} [D(\tilde{x})]$$

where D is the set of1−Lipschitz functions and Pg is once again the model distribution The idea here is that critic value approximates K·W(Pr,Pθ), where K is a Lipschitz constant and W(Pr,Pθ) is a Wasserstein distance. In this set-ting, a discriminator network is called critic and it approximates the distance between the samples. To enforce Lipschitz constraint in WGAN Arjovsky et al. add weight clip-ping to[−c,c]. Gulrajani et al. propose to add a gradient penalty term instead :

$$\lambda \mathop{\mathbb{E}}_{\tilde{x} \backsim \mathbb{P}_{\tilde{x}}} [(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1)^2]$$

the value function as an alternative way to enforce the Lipschitz constraint. This approach is robust to the choice of generator architecture and requires almost no hyperparameter tuning. This is crucial for image deblurring as it al-lows to use novel lightweight neural network architectures in contrast to standard Deep ResNet architectures, previously used for image deblurring .

Generator Model

```
def res_block(input, filters, kernel_size=(3,3), strides=(1,1), use_dropout=False):
    x = ReflectionPadding2D((1,1))(input)
    x = Conv2D(filters=filters,
               kernel_size=kernel_size,
               strides=strides,)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    if use_dropout:
        x = Dropout(0.5)(x)

    x = ReflectionPadding2D((1,1))(x)
    x = Conv2D(filters=filters,
               kernel_size=kernel_size,
               strides=strides,)(x)
    x = BatchNormalization()(x)

    # Two convolution layers followed by a direct connection between input and output
    merged = Add()([input, x])
    return merged
```

```
def generator_model():
    """Build generator architecture."""
    # Current version : ResNet block
    inputs = Input(shape=image_shape)

    x = ReflectionPadding2D((3, 3))(inputs)
    x = Conv2D(filters=ngf, kernel_size=(7,7), padding='valid')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # Increase filter number
    n_downsampling = 2
    for i in range(n_downsampling):
        mult = 2**i
        x = Conv2D(filters=ngf*mult*2, kernel_size=(3,3), strides=2, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

    # Apply 9 ResNet blocks
    mult = 2**n_downsampling
    for i in range(n_blocks_gen):
        x = res_block(x, ngf*mult, use_dropout=True)

    # Decrease filter number to 3 (RGB)
    for i in range(n_downsampling):
        mult = 2**(n_downsampling - i)
        x = Conv2DTranspose(filters=int(ngf * mult / 2), kernel_size=(3,3), strides=2, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

    x = ReflectionPadding2D((3,3))(x)
    x = Conv2D(filters=output_nc, kernel_size=(7,7), padding='valid')(x)
    x = Activation('tanh')(x)

    # Add direct connection from input to output and recenter to [-1, 1]
    outputs = Add()([x, inputs])
    outputs = Lambda(lambda z: z/2)(outputs)

    model = Model(inputs=inputs, outputs=outputs, name='Generator')
    return model
```

2.2 Conditional adversarial networks :

Generative Adversarial Networks have been applied to different image-to-image translation problems, such as super resolution , style transfer , product photo generation and others. Isola et al. provides a detailed overview of those approaches and present conditional GAN architecture also known as pix2pix. Unlike vanilla GAN, c GAN learns a mapping from observed image x and random noise vector z, toy: G:x,z→y. Isola et al .also put a condition on the discriminator and use U-net architecture for generator and Markovian discriminator which allows achieving perceptually superior results on many tasks, including synthesizing photos from label maps ,reconstructing objects from edge maps, and colorizing images.

the conditional version of generative adversarial nets, which can be constructed by simply feeding the data, y, we wish to condition on to both the generator and discriminator. We show that this model can generate MNIST digits conditioned on class labels. We also illustrate how this model could be used to learn a multi-modal model, and provide preliminary examples of an application to image tagging in which we demonstrate how this approach can generate descriptive tags which are not part of training labels.

2.3 ResNet GAN:

In its simplest form, a ResNet network is a network with residual layers. A residual layer is a layer in which the layer input is added to the layer output. This connection from the layer input to the layer output is called a residual connection.

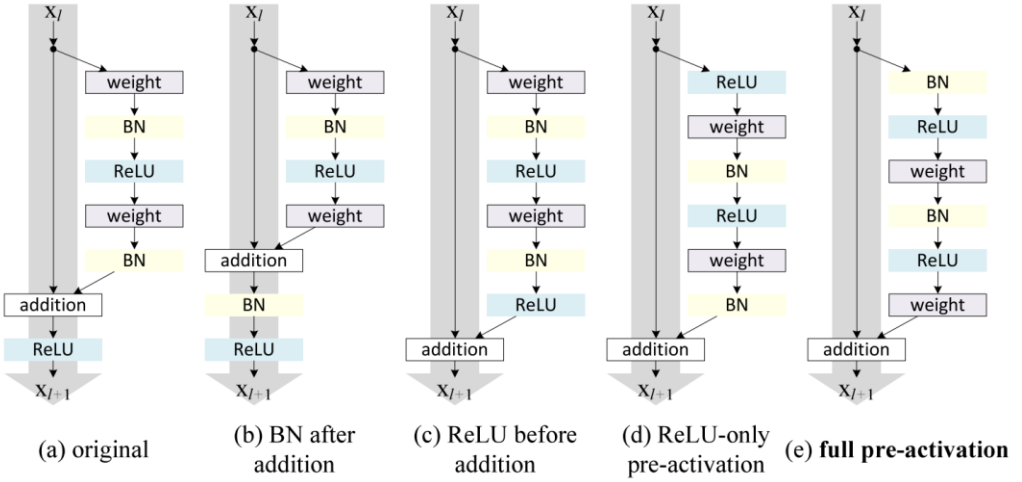


figure 2.1

3.Proposed Method:

The Aim is to retrieve sharp image IS where the input given input is only a blurred image IB, so no information about the blur kernel K is provided. Debluring is done by the trained CNN GθG, to which we refer as the Generator. For each IB (Blur image) it estimates corresponding IS(sharp image) image. In addition, during the training phase, we introduce critic the network DθD and train both networks in an adversarial manner.

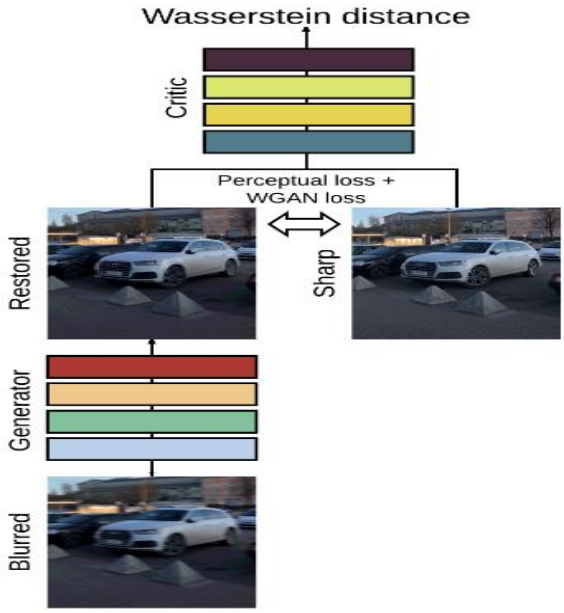
3.1 Loss Function:

The total loss here is defined as a combination of two different losses i.e., content loss and adverserial loss and it is formulated as:

$$\mathcal{L} = \underbrace{\underbrace{\mathcal{L}_{GAN}}_{adv\ loss} + \underbrace{\lambda \cdot \mathcal{L}_{X}}_{content\ loss}}_{total\ loss}$$

where in all experiments the λ equals to 100 . Unlike Isola et al. we do not condition the discriminator as we do not need to penalize mismatch between the input and output .Adversarial loss Most of the papers related to conditional GANs, use vanilla GAN objective as the loss function. Recently provides an alternative way of using least aquare GAN which is more stable and generates higher quality results. The critic function we use

is WGAN-GP , which is known to be robust to the choice of generator architecture . Our introductory experiments with different architectures confirmed that findings and we are able to use architecture much lighter than ResNet152 . The loss is calculated as the following



$$\mathcal{L}_{GAN} = \sum_{n=1}^{N} -D_{\theta_D}(G_{\theta_G}(I^B))$$

figure 3.1

Deblur GAN trained without GAN component converges ,but produces smooth and blurry images.Content loss. Two classical choices for ”content” loss function areL1 pr MAE loss,L2 or MSE loss on raw pixels. Using those functions as sole optimization target leads to the blurry artifacts on generated images due to the pixel-wise average of possible solutions in the pixel space .Instead, we adopted recently proposed Perceptual loss .Perceptual loss is a simple L2-loss, but based on the difference of the generated and target image CNN feature maps. It is defined as following:

DeblurGAN training. The generator network takes the blurred image as input and produces the estimate of the sharp image. The critic network takes the restored and sharp images and outputs a distance between them. The total loss consists of the WGAN loss from critic and the perceptual loss . The perceptual loss is the difference between the VGG-19 [34]conv3.3feature maps of the sharp and restored images. At test time, only the generator is kept

$$\mathcal{L}_X = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^S)_{x,y} - \phi_{i,j}(G_{\theta_G}(I^B))_{x,y})^2$$

where φi,jis the feature map obtained by the j-th convolution (after activation) before the i-th maxpooling layer within the VGG19 network, pretrained on ImageNet ,Wi,j and Hi,j are the dimensions of the feature maps. In our work we use activations from V GG3,3convolutionallayer. The activations of the deeper layers represents the features of a higher abstraction . The perceptual loss focuses on restoring general content while adversarial loss focuses on restoring texture details. Deblur-GAN trained without Perceptual loss or with simple MSE on pixels instead doesn’t converge to meaningful state.

Losses

```
from tensorflow.keras.applications.vgg16 import VGG16

image_shape = (256, 256, 3)

def perceptual_loss(y_true, y_pred):
    vgg = VGG16(include_top=False, weights='imagenet', input_shape=image_shape)
    loss_model = Model(inputs=vgg.input, outputs=vgg.get_layer('block3_conv3').output)
    loss_model.trainable = False
    return K.mean(K.square(loss_model(y_true) - loss_model(y_pred)))


def wasserstein_loss(y_true, y_pred):
    return K.mean(y_true*y_pred)
```

figure 3.2

figure 3.3

3.2 Network architecture:

Generator CNN architecture similar to one proposed by Johnsonet al. It contains two strided convolution blocks with stride12, nine residual blocks (ResBlocks) and two transposed convolution blocks. Each ResBlock consists of a convolution layer, instance normalization layer ,and ReLU activation .Dropout regularization with a probability of 0.5 is added after the first convolution layer in each ResBlock. In addition, we introduce the global skip connection which we refer to as ResOut. CNN learns a residual correction IR to the blurred image IB, so IS=IB+IR. We find that such formulation makes training faster and resulting model generalizes better. During the training phase, we define a critic network DθD, which isWasserstein GAN with gradient penalty , to which we refer as WGAN-GP. The architecture of critic network is identical to Patch GAN . All the convolutional layers except the last are followed by Instance Norm layer. and Leaky ReLU with α= 0.2.

Discriminator Model

```
ndf = 64
output_nc = 3
input_shape_discriminator = (256, 256, output_nc)


def discriminator_model():
    """Build discriminator architecture."""
    n_layers, use_sigmoid = 3, False
    inputs = Input(shape=input_shape_discriminator)

    x = Conv2D(filters=ndf, kernel_size=(4,4), strides=2, padding='same')(inputs)
    x = LeakyReLU(0.2)(x)

    nf_mult, nf_mult_prev = 1, 1
    for n in range(n_layers):
        nf_mult_prev, nf_mult = nf_mult, min(2**n, 8)
        x = Conv2D(filters=ndf*nf_mult, kernel_size=(4,4), strides=2, padding='same')(x)
        x = BatchNormalization()(x)
        x = LeakyReLU(0.2)(x)

    nf_mult_prev, nf_mult = nf_mult, min(2**n_layers, 8)
    x = Conv2D(filters=ndf*nf_mult, kernel_size=(4,4), strides=1, padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(0.2)(x)

    x = Conv2D(filters=1, kernel_size=(4,4), strides=1, padding='same')(x)
    if use_sigmoid:
        x = Activation('sigmoid')(x)

    x = Flatten()(x)
    x = Dense(1024, activation='tanh')(x)
    x = Dense(1, activation='sigmoid')(x)

    model = Model(inputs=inputs, outputs=x, name='Discriminator')
    return model
```

4.Training Details:

We implemented all of our models using Keras deep learning framework. The training was performed on a single Maxwell GTX Titan-X GPU using three different datasets .The first model to which we refer as Deblur GAN WILD was trained on a random crops of size 256x256 from 1000 GoPro training dataset images downscaled by a factor of two. The second one Deblur GAN Synth was trained on 256x256 patches from MSCOCO dataset blurred by method, presented in previous section. We also trained Deblur GAN Comb on a combination of synthetically blurred images and images taken in the wild, where the ratio of synthetically generated images to the images taken by a high frame-rate camera is 2:1. As the models are fully convolutional and are trained on image patches they can be applied to images of arbitrary size. For optimization we follow the approach of and perform 5 gradient descent steps on DθD , then one step on GθG, using Adam as a solver. The learning rate is set initially to10−4for both generator and critic. After the first 150 epochs we linearly decay the rate to zero over the next150 epochs. At inference time we follow the idea of and apply both dropout and instance normalization. All the models were trained with a batch size = 1, which show edempirically better results on validation. The training phase took 6 days for training one Deblur GAN network.

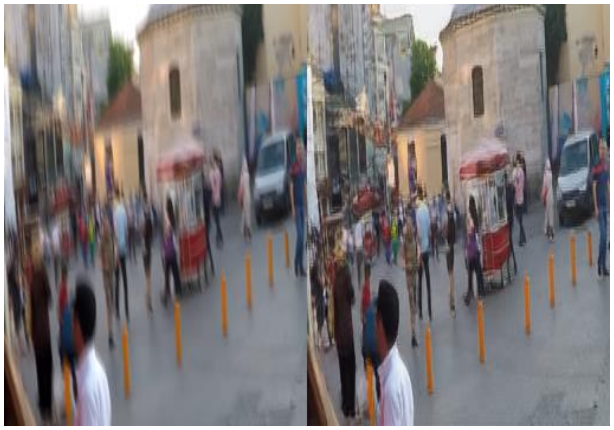


figure 4.1

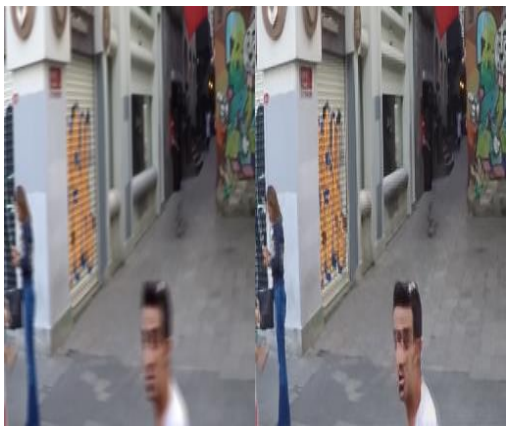


figure 4.2

Discriminator on Generator Model

```
[ ]  def generator_containing_discriminator_multiple_outputs(generator, discriminator):
         inputs = Input(shape=image_shape)
         generated_images = generator(inputs)
         outputs = discriminator(generated_images)
         model = Model(inputs=inputs, outputs=[generated_images, outputs])
         return model
```

4.1 Algorithm:

Algorithm 1 :

Motion blur kernel generation.Parameters:

M= 2000 – number of iterations,

L_{max}= 60 – max length of the movement ,

p_s= 0.001 – probability of impulsive shake,

I– inertia term, uniform from (0,0.7),

p_b– probability of big shake,

uniform from (0,0.2),

p_g– probability of gaussian shake,

uniform from (0,0.7),

φ– initial angle,

uniform from (0,2π),

x– trajectory vector.

1:procedureBLUR(Img,M,L_{max},p_s)

2 :$v_0 \leftarrow \cos(\varphi) + \sin(\varphi) * i$

3: $v \leftarrow v_o * L_{max}/(M-1)$

4 :x=zeros(M,1)

5: for t= 1toM−1do

6: if randn < $p_b * p_s$ then

7: $nextDir \leftarrow 2 \cdot v \cdot e^{i*(\pi+(randn-0.5))})$

8: else:

9 :nextDir←0

10: dv←nextDir+p_s∗(p_g∗(randn+i∗randn)∗I∗x[t]∗(L_{max}/(M−1))

11 :v←v+dv

12: v←(v/abs(v))∗L_{max}/(M−1)

13: x[t+ 1]←x[t] +v

14: Kernel←sub pixel interpolation(x)

15: Blurred image←conv(Kernel,Img)

16: returnBlurred image

5.Experimental Results:

GoPro dataset consists of 2103 pairs of blurred and sharp images in 720p quality, taken from various scenes. We compare the results of our models with state of the art models on standard metrics and also show the Table : Peak signal-to-noise ratio and the structural similarity measure, mean over the GoPro test dataset of 1111images. All models were tested on the linear image subset. State-of-art results(∗)by Nah et al. obtained on the gamma subset.

Metric	Sun et al. [36]	Nah et al. [25]	Xuet al. [44]	DeblurGAN WILD	DeblurGAN Synth	DeblurGAN Comb
PSNR	24.6	28.3/29.1*	25.1	27.2	23.6	28.7
SSIM	0.842	0.916	0.89	0.954	0.884	0.958
Time	20 min	4.33s	13.41s			0.85s

running time of each algorithm on a single GPU. Result sare in Table1. DeblurGAN shows superior results in terms of structured self-similarity, is close to state-of-the-art in peak signal-to-noise-ratio and provides better looking results by visual inspection. In contrast to other neural models, our network does not use L2 distance in pixel space soit is not directly optimized for PSNR metric. It can handle blur caused by camera shake and object movement, does not suffer from usual artifacts in kernel estimation methods and at the same time has more than 6x fewer parameters comparing to Multi-scale CNN , which heavily speeds up the inference.

figure 5.1

figure 5.2

figure 5.3

Deblured images from test on GoPro data set are shown in Figure

6.Conclusion:

Finally, we successively train the discriminator and the generator, based on both losses. We generate fake inputs with the generator. We train the discriminator to distinguish fake from real inputs, and we train the whole model.We described a kernel-free blind motion deblurring learning approach and introduced DeblurGAN which is a Conditional Adversarial Network that is optimized using a multi-component loss function. In addition to this, we implemented a new method for creating a realistic synthetic motion blur able to model different blur sources. We introduce a new benchmark and evaluation protocol based on results of object detection and show that DeblurGAN significantly helps detection on blurred images.

In the past years, deconvolution proved to be a resolvable problem that can aid in many domains, from medical imaging to space photography. The theoretical problems that made deconvolution be overlooked for everyday photography until the third millennium, like ill conditioned systems, high ratios of noise amplification because of inversion of small values in the blur kernel, artifacts originating from real values truncation and others, found their solutions with practical approaches in a very short time . This Research has proven that is now ready to be used in everyday applications, like introducing special camera aperture or coded camera exposures that can aid the software editing of blur in photographs, modifying the medical instruments so that they incorporate these algorithms in order to give clearer results

References:

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN.ArXiv e-prints, Jan. 2017. 1, 3, 4, 5, 7

[2] S. D. Babacan, R. Molina, M. N. Do, and A. K. Katsaggelos.Bayesian blind deconvolution with general sparse image pri-ors. InEuropean Conference on Computer Vision (ECCV),Firenze, Italy, October 2012. Springer. 2

[3] G. Boracchi and A. Foi. Modeling the performance of im-age restoration from motion blur.Image Processing, IEEETransactions on, 21(8):3502 –3517, aug. 2012. 5

[4] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Kr-ishnan. Unsupervised Pixel-Level Domain Adaptation withGenerative Adversarial Networks.ArXiv e-prints, Dec. 2016.3

[5] A. Chakrabarti. A neural approach to blind motion deblur-ring. InLecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and LectureNotes in Bioinformatics), 2016. 3, 5

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.ImageNet: A Large-Scale Hierarchical Image Database. InCVPR09, 2009. 4

[7] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T.Freeman. Removing camera shake from a single photograph.ACM Trans. Graph., 25(3):787–794, July 2006. 2, 5

[8] D. Gong, J. Yang, L. Liu, Y. Zhang, I. Reid, C. Shen, A. VanDen Hengel, and Q. Shi. From Motion Blur to Motion Flow:a Deep Learning Solution for Removing Heterogeneous Mo-tion Blur. 2016. 3

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu,D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Gen-erative Adversarial Networks. June 2014. 1, 3

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, andA. Courville.Improved Training of Wasserstein GANs.ArXiv e-prints, Mar. 2017. 1, 3, 4, 5

[11] A. Gupta, N. Joshi, C. L. Zitnick, M. Cohen, and B. Curless.Single image deblurring using motion density functions. InProceedings of the 11th European Conference on ComputerVision: Part I, ECCV'10, pages 171–184, Berlin, Heidel-berg, 2010. Springer-Verlag. 3

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learn-ing for image recognition.arXiv preprint arXiv:1512.03385,2015. 3, 5[14] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Scholkopf.Fast removal of non-uniform camera shake. InProceedingsof the 2011 International Conference on Computer Vision,ICCV '11, pages 463–470, Washington, DC, USA, 2011.IEEE Computer Society. 8[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Wein-berger. Densely connected convolutional networks.2017IEEE Conference on Computer Vision and Pattern recognition.arXiv preprint arXiv:1512.03385,2015.