Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition

Ilya Novikov [1] **Vlad Cheremnykh** [1] **Grigory Vyaznikov** [1] **Oleg Nesterenkov** [1] **Ksenia Kosmynina** [1]

Abstract

This research focuses on comparing, testing and improving deblurring algorithms. We have implemented the following approaches: 1) Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition, 2) DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. In our project, we have conducted several experiments to deblur images from Gopro and Kohler datasets. This article presents an algorithm and obtained results, as well as our conclusions about the effectiveness of each approach and some suggestions for further improvements.

Github repo: Github
Presentation file: Presentation

1. Introduction

Motion blur estimation and restoration are fundamental problems in image processing and computer vision. Motion blur is typically produced by camera or object movement, inaccurate focusing, low resolution, out-of focus and so on. Blurring causes significant degradation in image quality.

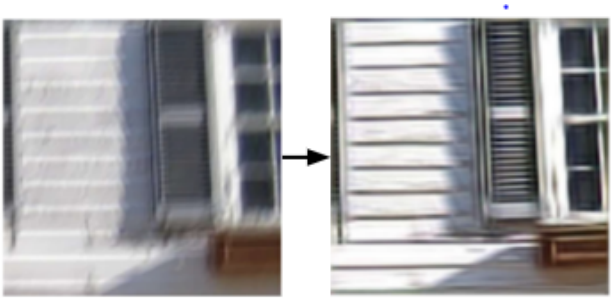

Figure 1. An example of blureed and deblurred images.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Ksenia Kosmynina <ksenia.kosmynina@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

Consequences such as aesthetics and performance degradation of subsequent computer vision tasks: tracking, detection, classification may occur. In order to avoid them and get a clearer picture we can either reshoot the same photo or use our knowledge of Machine Learning and reproduce a deblurred image.

In this work we focus on the realistic non-uniform motion blur estimation. Major goal of our study is to provide dense, accurate estimates of non-uniform motion fields via local kernel estimation. Estimate kernels at the pixel level. Perform non-blind image deblurring.

The literature on motion kernel estimation is extremely vast. We limit the current analysis to spatially varying kernel estimation methods from a single image. Early methods attempting to estimate non-uniform motion blur kernels are limited to the case of camera egomotion.

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

$$I_B = k(M) * I_S + N$$

where I_B is a blurred image, $k(M)$ are unknown blur kernels determined by motion field M. I_S is the sharp latent image, * denotes the convolution, N is an additive noise.

Among all known digital image processing methods, in this study we will examine the following:

- Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition;
- DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks.

1.1. Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition

The detailed explanation of this method is described in [1]. Shortly, given a blurry image, a neural network predicts a set of kernel basis and corresponding pixel-wise mixing coefficients allowing to deblur the corresponding sharp image by first convolving it with each basis kernel and performing a weighted sum using the mixing coefficients.

Fig.2 demonstrates a principle of operation of an adaptive method, we can observe that the model calculates mixing coefficients and different kernels for different objects in a given image. We want to emphasize that this model works well for digital images in which some objects are moving, some are stable.

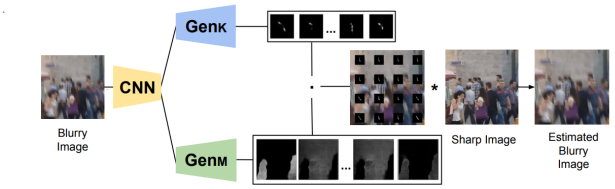

Figure 2. Overview of an Adaptive method.

Fig.3 shows examples of generated kernel basis and corresponding mixing coefficients predicted from the blurry images shown on the left. The adaptation to the input is more notorious for the elements that have significant weights.

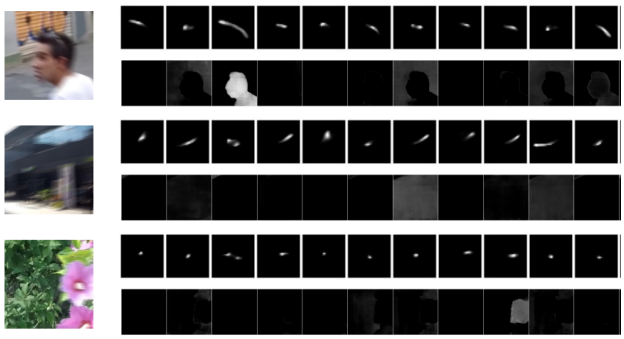

Figure 3. Examples of generated kernels and corresponding mixing coefficients.

The proposed network is composed by an encoder and 2 decoders. The encoder takes as an input a blurry image. Decoders output the motion kernel basis and corresponding mixing coefficients. Fig.4 demonstrates an architecture of the network.

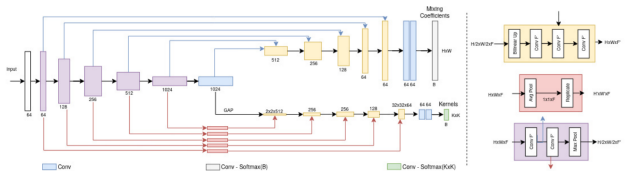

Figure 4. Architecture Details.

1.2. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks

DeblurGAN – an approach based on conditional generative adversarial networks and a multi-component loss function. For this method, Wasserstein GAN with the gradient penalty and perceptual loss are used. This solution allows to restore finer texture details than if using traditional MSE or MAE as an optimization target. A detailed explanation of this method is presented in a paper [2].

Fig.5 demonstrates a DeblurGAN generator architecture. DeblurGAN contains two strided convolution blocks with stride 12, nine residual blocks and two transposed convolution blocks. Each ResBlock consists of a convolution layer, instance normalization layer, and ReLU activation.

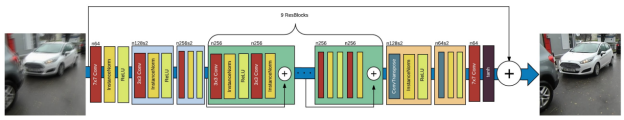

Figure 5. DeblurGAN generator architecture.

1.3. Paper Organization

In this study, we were asked to solve the following tasks:

- Use pre-trained Non-uniform model and compute PSNR and SSIM metrics for both Kohler and Gopro datasets. Compare results with the reference.
- Use pre-trained DeblurGAN model and follow the same steps. In case if results are worse, we can try to retrain a model.
- Write a code to train a Non-uniform model. Plot training and validation metrics depending on number of epoch. Compare results.
- In case of failure in task 3, try to reproduce tables 1 and 2 from the DeblurGAN paper and train the networks from scratch.

In this report, we have described all the results obtained during our work in sections 3 and 4. We have conducted several experiments on both DeblurGAN and Adaptive models with different datasets.

The main contributions of this report are as follows: We have reproduced all the described steps and validated 2 papers [1] and [2], and it was proven that authors of [1] falsified their results.

2. Related Work

Recent years, a huge amount of work was conducted on the topic of deblurring of non-uniform images. Lots of approaches were developed and described in a literature.

For instance, in paper [3] authors proposed a parameterized geometric model of the blurring process in terms of the rotational motion of the camera during exposure.

Also, in study [4] authors have developed a model to handle a non-uniform blurring. Their algorithm has two main components: A new method for recovering the unknown blur-field directly from the blurry image, and a method for deblurring the image given the recovered nonuniform blur-field.

One of the latest works [5] proposes a solution to transform spatially variant blurry images into the photo-realistic sharp manifold. In this paper, authors investigate an attention network for image deblurring. Comprehensive experimental results prove that this method outperforms many other state-of-the-art methods.

Despite the fact that lots of research was done in this field, still image deblurring task is extremely challenging in computer vision. We have checked lots of different approaches and architectures for deblurring non-uniform digital images to have a clearer picture of the problem. However, in our project we had to conduct a research and try to reproduce results from [1] and [2].

3. Algorithms and Models

Since we had pre-processed Kohler and Gopro datasets which are available in a Github repository, and our goal was to reproduce results of given models, we started the process of validation as follows. Firstly, we decided to measure the deblur performance of an adaptive model. For this goal, we used a Kohler dataset, which described in [8].

Kohler is a dataset that was created using static objects (printed photo posters) and a monopod located on a moving platform and having 6 axes of movement (3 rotating and 3 translating). It would not be entirely correct to compare the original images with those obtained due to some algorithm, since different resolutions and lighting would contribute to the variability of the metrics. In this way, reliable images along the trajectory were generated by the mobility of the platform, reproducing the camera trajectory step by step and taking a picture. Using this strategy, a dataset was formed of 12 camera shakes, 4 images and 200 ground truth images along the trajectory. The resulting kernels (1,3,5,7,9,11) are non-uniform, (2,4,6,8,10,12) are approximately uniform.

The Gopro dataset for deblurring consists of 3,214 blurred images with the size of 1,280×720 that are divided into 2,103 training images and 1,111 test images. The dataset consists of pairs of a realistic blurry image and the corresponding ground truth sharp image that are obtained by a high-speed camera.

To compare similarity between 2 images A and B (represented as vectors), we first estimate the optimal scaling a and translation T such that L2 norm between A and B becomes minimal, after that we calculate PSNR as:

$$PSNR(A,B) = 10log_{10} * (m^2/||A - T(aB)||^2)$$

where m is a maximal possible intensity value, i.e. m =255 as we work with 8bit encoding. Given a sequence of ground truth images U*, along the trajectory, we define the PSNR similarity between an estimated image U and the ground truth as the maximum PSNR between U and any of the images along the trajectory.

$$SIM = maxPSNR(U, U*)$$

Gopro is a dataset created by cutting video into frames. It is important to note that most of the photos were rendered with motion blur due to camera movement, but there are also frames with objects moving in different directions, it is difficult to assess the ratio of uniform and non-uniform. Also, there were no intermediate ground truth images, so standard formulas of MSE and PSNR were used.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$PSNR = 10 * log_{10}(\frac{MAX_I}{MSE})$$

$$PSNR = 20 * log_{10}(\frac{MAX_I^2}{\sqrt{MSE}})$$

$$PSNR = 20 * log_{10}(MAX_I) - 10 * log_{10}(MSE)$$

SSIM was calculated using the following formula:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

In our project, we were not asked to develop our own algorithms, instead we had to reconstruct and test Adaptive and DeblurGAN algorithms, which are available in our Repository.

4. Experiments and Results

We have conducted several experiments with two datasets and models.

According to the results obtained, it can be seen in Fig.6 that on Kohler, the Adaptive non-uniform dataset showed better results than DeblurGan in both metrics. But the result obtained by us still did not coincide with that indicated in the article.

To validate the implemented algorithm for calculating PSNR and SSIM, we took ready-made Images From Hirch Deblurred Dataset and Blurry image From Kohler and calculated metrics for them. The result coincided with the expected, so we tried to run the Adaptive model with a different gamma parameter (in the article it was indicated that gamma = 1.0 on the Kohler dataset, the standard value for the model is 2.20), this also did not allow us to achieve the result indicated by the authors. Received values picture by picture

PSNR Expected	27.58	33.16	NA	35.19
PSNR Calculated	27.58	33.16	26.98	34.47

Figure 6. Results of deblurring.

Fig.7 demonstrates cropped deblurred digital images by both Adaptive and DeblurGAN methods. We can observe than not only by calculations, but also visually that Adaptive algorithm gives better results.

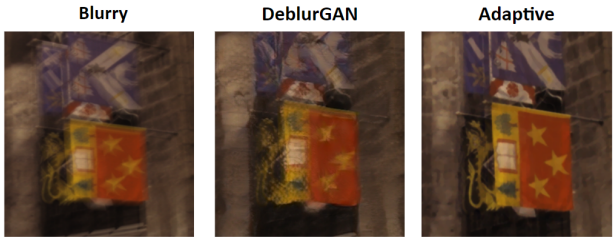

Figure 7. Cropped images.

On the Gopro dataset, the adaptive model showed a worse result than the DeblurGAN, this can be explained by the fact that there is a smaller percentage of images with non-uniform blur than in the above-mentioned Kohler.

Table 1. Mean Values of PSNR and SSIM metrics.

Method	KOHLER		GOPRO	
	PSNR	SSIM	PSNR	SSIM
DeblurGAN	**25.97**	**0.75**	**32.12**	**0.98**
Adaptive	**26.86**	**0.80**	**31.21**	**0.91**
Reference Value	**28.39**	**0.82**	**NA**	

Also, we have calculated mean PSNR and SSIM metrics for each method with each dataset. It is seen that Adaptive performs better again. The results are presented in Table 1.

In the research task, we were asked to write a code for a training process of a Non-uniform Blur Kernel Estimation Neural Network, since authors of a corresponding work have not provided an initial training code for an open access. The task can be divided into two parts:

- Creating a non-uniform blurred dataset for training;
- Reproduction of the training model itself.

Due to there are no available datasets which satisfy our requirements of non-uniform blur, it is an important task to create such dataset for deblur models training. It can be done both with high frame-rate camera and generated synthetically. Our dataset is based on the training set of the ADE20K semantic segmentation dataset [6]. We chose a subset of sharp images, contains at least one moving object (plane, person, car) and convolved it with random generated kernels. Each kernel corresponds to the one object. Kernels were generated with random trajectories approach [7]. A pseudo-code of the non-uniform dataset generation is presented Algorithm 1. Due to there are some non-moving objects in images, we assumed to use them as a background.

Algorithm 1 Non-uniform dataset generation

```
procedure DatasetGeneration(ImgSet)
    DataSet = {}
    KernelSet = {}
    for Image in ImgSet do
        for SegObject in Image do
            kernel := RandomKernel()
            SegObject := SegObject * kernel
            KernelSet.append(kernel)
        end for
        DataSet.append(Image)
    end for
    return DataSet, KernelSet
end procedure
```

Since persons and vehicle registration plates are blurry in the currently available ADE20K dataset version, for your implementation we suggest to use another semantic segmentation dataset.

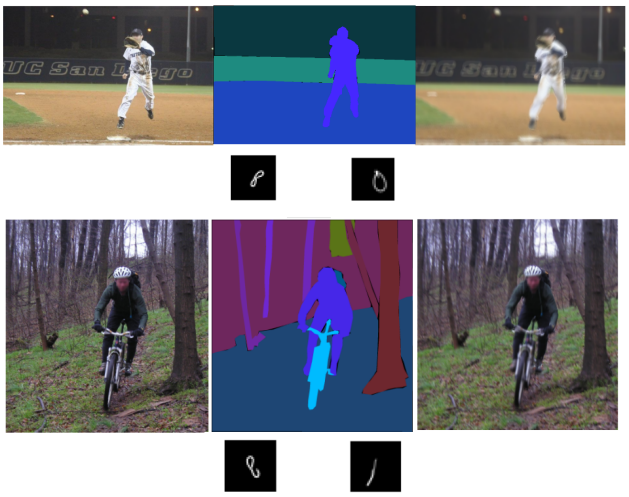

Figure 8. From left to right: sharp image, segmented objects, blured image. Kernels which were used are presented in a second row.

The training process of a neural network takes place as follows: a synthetically blurred image, corresponds sharp image and blurring kernels are fed to the input. The model is processing the blurred image and estimates the set of baseline blurring kernels and mixing coefficients, linear combination of which reproduces the set of per-pixel blurring kernels of the image. After that, reblur and kernel loss functions are calculated. For the repaired model, these two functions were repaired using PyTorch.

Figure 9. DeblurGAN merged uniform dataset.

The reproduction of the non-uniform training model and the generation of a training dataset was too long, so we decided to train the DeblurGAN Neural Network in parallel. It was easier to do so, because the initial code was available in the developers' Github repository. To perform a training model, the training dataset was generated. The program provided by developers takes at an input a blurred image and a corresponding sharp image, after that merges them together. The resulting image will be used for the training. 2103 training pairs of images with 1280 x 720 resolution with value of gamma function = 2.2 of the Gopro training dataset were taken as the basis. Due to the structure of the Gopro dataset, which consists of many sub-folders with images of different nature, the dataset making program provided by the developers was modified to work with the specified structure. As the result, we have obtained the dataset above (Fig.9). The

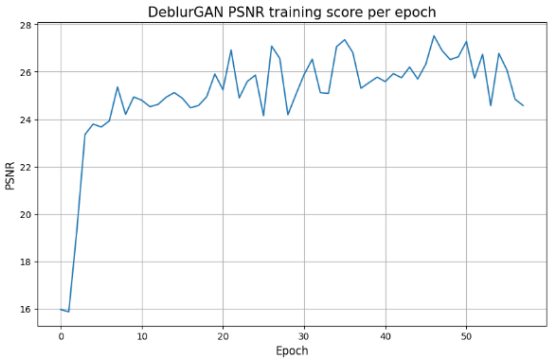

Figure 10. The model training process.

resulting dataset was loaded to the training model, which was also modified for a multi-fold structure. The training lasts for 300 epochs, during which various model quality metrics were captured, including PSNR. We have concluded that the model quality increases during the learning process. Fig.10 demonstrates the training during 58 epochs.

Figure 11. An example of obtained results for the 1st epoch.

Figure 12. An example of obtained results for the 58th epoch.

Fig.11 and Fig.12 demonstrate an example of blurred, re-

stored and sharp image during one epoch. We took to samples do demonstrate the difference in the model accuracy during its learning. We trained our model during 58 epochs due to a too long time of training (about 10 hours for 58 epochs). We can visually estimate an accuracy of a developed model and observe a difference between blurred and restored images.

5. Conclusion

In this work, we have conducted a research on 2 methods of image deblurring - Adaptive basis decomposition and DeblurGAN.

We have figured out that our calculations are accurate for the Hirsch method, which proves that we have defined the correct metrics. However, we encountered the following problem: the results for the Adaptive method differ from the reference value of PSNR by 1.5 dB in average. It can be assumed that authors falsified the results, since they did not make the training model publicly available and their article was not published.

Our suggestions how to improve a working process include: 1) Creation of a python code instead of Matlab for PSNR calculation of Kohler dataset; 2) Validation of authors' results, since they do not seem to be reproducible at some point.

References

[1] Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition // Guillermo Carbajal, Patricia Vitoria, Mauricio Delbracio, Pablo Musé, José Lezama // 1 Feb 2021.

[2] DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks // Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, Jiri Matas // Nov 2017.

[3] Non-uniform Deblurring for Shaken Images // Whyte, O., Sivic, J., Zisserman, A. // Int J Comput Vis 98, 168–186 // 2012.

[4] Non-Uniform Blind Deblurring by Reblurring // Yuval Bahat, Netalee Efrat, Michal Irani // Proceedings of the IEEE International Conference on Computer Vision (ICCV) // 2017.

[5] Attention Network for Non-Uniform Deblurring // QING QI, JICHANG GUO, WEIPEI JIN // June 8, 2020.

[6] Semantic understanding of scenes through the ade20k dataset // Zhou, Bolei and Zhao, Hang and Puig, Xavier and Xiao, Tete and Fidler, Sanja and Barriuso, Adela and Torralba, Antonio // International Journal of Computer Vision // 2019

[7] Modeling the performance of image restoration from motion blur // G. Boracchi and A. Foi // Image Processing, IEEE Transactions // August 2012

[8] R. Kohler, M. Hirsch, B. Mohler, B. Sch ¨ olkopf, and S. Harmeling // Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database // Proceedings of the 12th European Conference on Computer Vision - Volume Part VII, ECCV'12, pages 27–40 // Berlin, Heidelberg // 2012.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Ilya Novikov (20% of work)

- Project coordination
- Coding the main algorithm
- DeblurGAN training and metrics recording

Vlad Cheremnykh (20% of work)

- Preparing the Github Repo
- Preparation of DeblurGAN (Kohler) and Non-uniform adaptive model (Kohler) datasets
- PSNR, SSIM metrics calculation

Grigory Vyaznikov (20% of work)

- Reconstruction of a DeblurGAN algorithm
- Preparing Kohler and Gopro datasets
- Calculating mean values of metrics

Oleg Nesterenkov (20% of work)

- Writing a code for a non-uniform adaptive model
- Plotting metrics depending on number of epoch
- Synthetic dataset for a non-uniform model investigation and generation

Ksenia Kosmynina (20% of work)

- Preparing a presentation
- Reviewing literate on the topic (3 papers)
- Preparing this report

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: 90-95 percent was used and 5-10 percent we wrote on our own. We used code: Adapt DeblurGAN GoPro

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: We have provide formulas to calculate losses, PSNR and SSIM. Also, we have described models we used.

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: Github

4. A complete description of the data collection process, including sample size, is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: It was described in a report

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: We have all the links in our Github repository

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   - [ ] Yes.
   - [ ] No.
   - [x] Not applicable.

   Students' comment: We used a pre-processed dataset

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [x] Not applicable.

   Students' comment: We used a pre-processed dataset, which was devided for train-test in advance

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: We have added a list of hyper-parameters (also check repository: hyper-parameters)

9. The exact number of evaluation runs is included.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: We mentioned in our report

10. A description of how experiments have been conducted is included.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: It was described

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: It was described in our report

12. Clearly defined error bars are included in the report.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: It was described in our report

13. A description of the computing infrastructure used is included in the report.

    - [x] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: It was described in our report