# **Readme**

## Purpose of our research

Generally, state-of-the-art deblurring methods could be categorized into two groups:

1. Priors based deblurring methods, e.g., l0-prior, dark channel prior, extreme channel prior, two-tone prior, hyper Laplacian prior, elastic net prior, bright channel prior, discriminative prior, low-rank prior.

Pros of these priors:

1. They are built on profound and reliable theories. Compared to CNN-based models, they are more explainable.
2. They are built on ingenuous observations on the clear images.
3. They are effective on large-scale motion blur deblurring if the clear images fit the priors exactly.
4. These methods are heuristic to the deblurring researches.

Limitations:

1. They are usually based on iterative optimization, which is slow especially when the optimization process contains Laplacian pyramids.
2. These methods rely on extensive manual parameter adjustment. It is even worse than CNNs because they need to adjust the parameters for every image while CNNs only require hyperparameter adjustment during training.
3. These methods could hardly be generalized to non-uniform blurs.
4. These methods rely on very exact observation of the clear images. If the clear image do not observe the priors exactly, the optimization might fail.
5. These methods are not robust enough even without noises. **They are very sensitive to noises.**
6. CNNs, e.g., DeepDeblur, SRN-Deblur, Semantic face deblurring, Gated Fusion Network, DeblurGAN/DeblurGAN v2.

Pros of these networks:

1. Excellent results, especially good at handling complex blurs.
2. Fast inference.

Drawback of these networks:

They are sensitive to noises.

Our work is more engineering than theoretical. We find that both two approaches are sensitive to noises, and we only find very few papers to handle images in deblurring and these methods are not based on deep learning. Therefore, we would like to tackle this problem with deep learning.

## Code

The code is tested with torch >= 1.0.0, torchvision, PIL, numpy, python >= 3.5 and Ubuntu.

Please run as:

“python3 main\_release.py [gpuid] [r/c]”

“[gpuid]” and “[n/c] “are command-line arguments that are parsed by sys.argv.

If you have 4 gpus and you want to run it on gpu2, please set the [gpuid] to 2. The default gpuid is 0.

As for “[n/c]”, if you want to run on natural images, please set it to ‘n’, if you want to run on CelebA/face images, set it to ‘c’ please. The default choice is ‘n’.

For example, if you want to test on GOPRO with gpu 3, please run:

“python3 main\_release.py 3 n”.

If you want to test on CelebA with gpu 2, please run:

“python3 main\_release.py 2 c”.

If you run “python3 main\_release”, it means you run it with GPU 0 and test it on GOPRO.

I use very general python grammars and packages to improve the compatibility. Therefore, I think the code could be adapted to different systems/python versions/package versions with slight modification. However, since torch 0.3.0, torch 0.4.0 and torch 1.x are different, I think torch 1.2.0 is the best choice to run this code.

## Model:

GOPRO\_joint.pth: model for natural images.

CelebA\_joint.pth: model for face images.

I find that deblurring networks are sensitive to the category of images, e.g., models for text deblurring could hardly be generalized to face or natural image deblurring. Therefore I have to train two models.

## Data:

1. train1.zip/test1.zip: training images and test images on the GOPRO dataset.
2. train.zip/test.zip: training images and test images on the CelebA dataset.
3. PSFs: corresponding PSFs for image 1-100. Image 101 is a blurry image in the real scenario.
4. test.dict/test\_face.dict: noisy level record for test images on GOPRO/CelebA respectively.

## Contact:

Email: [miaosi2018@sari.ac.cn](mailto:miaosi2018@sari.ac.cn) [2904661326@qq.com](mailto:2904661326@qq.com)

QQ: 2904661326

Welcome to contact me if:

1. You are interested in this work, or you want to collaborate with me.
2. You have trouble with running this code.
3. You have trouble downloading data.
4. You have trouble understanding this work.

## Copyright:

I do not care much on the copyright of this code. You may feel free to share the data and the code to others without informing me. When sharing, it is better to indicate the source.

## Acknowledgement:

This code is based on:

# Kupyn et al. “DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks”. We use its generator.

# Zhang et al. “[Gated Fusion Network for Joint Image Deblurring and Super-Resolution](http://www.researchgate.net/publication/326697054_Gated_Fusion_Network_for_Joint_Image_Deblurring_and_Super-Resolution?_sg=Ibpu-TDV_jKUyeeODA1R11tm7nceJYDG4cmmlbTxwcqhc4ZGxZiZ0tTmS8YGCAa2bsDzURwjSWgfcJJ88CrsbksgCPd_Dg" \t "http://xueshu.baidu.com/usercenter/paper/_blank)”. It proves that if would be better if we remove the InstanceNorm and Sigmoid from DeblurGAN. In addition, it has a considerable performance without a multi-scale structure. Its code is also easy to read.

1. Pan et al. “Deblurring face images with exemplars”. It is helpful for blur kernel estimation. Instead of using their exemplars, we use our estimated sharp images as exemplars.

Thanks to the authors for their reliable codes.

## Other demo examples:









## Failed Case/Limitation:

