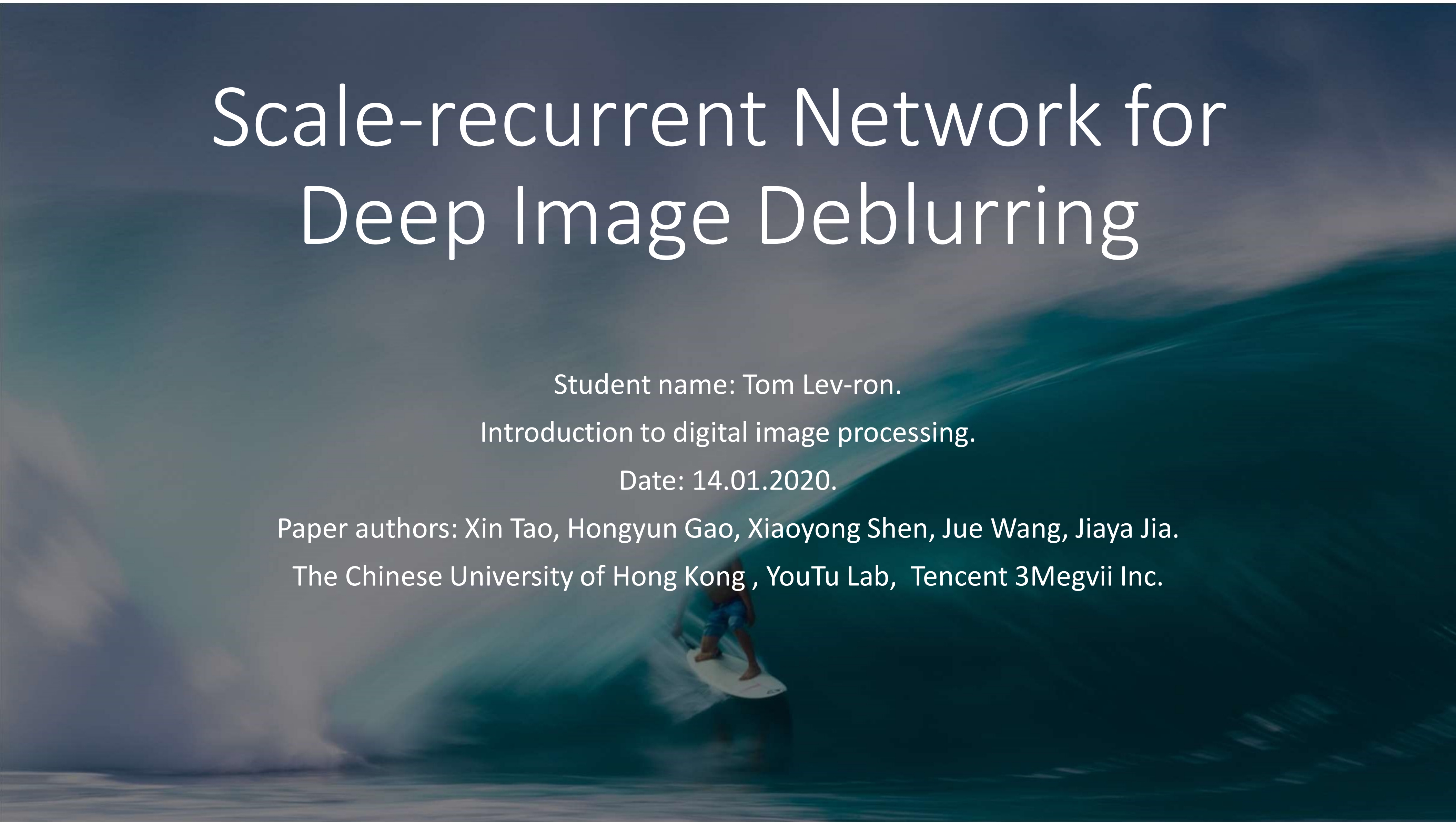


Scale-recurrent Network for Deep Image Deblurring

Student name: Tom Lev-ron.

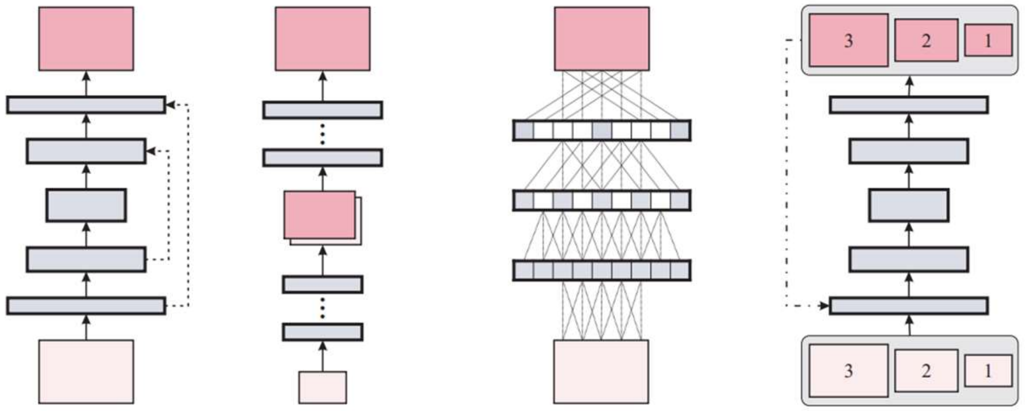
Introduction to digital image processing.

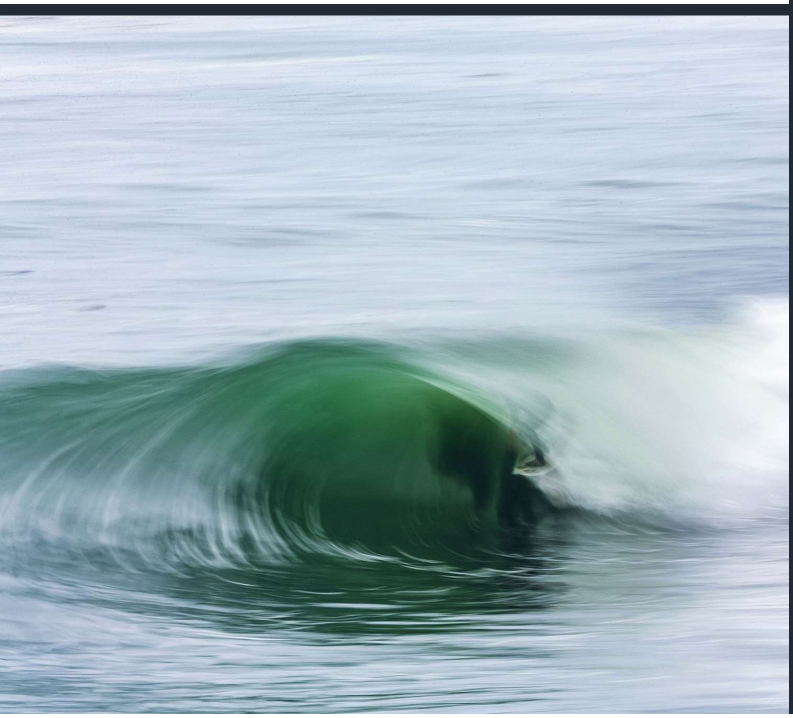
Date: 14.01.2020.

Paper authors: Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, Jiaya Jia.

The Chinese University of Hong Kong , YouTu Lab, Tencent 3Megvii Inc.

Opening notes





- This Presentation will fucus more on the article method and less about deblurring in general.
- Some of the basics principles of NN from the course won't be covered.
- High-level explanation of the suggested method.

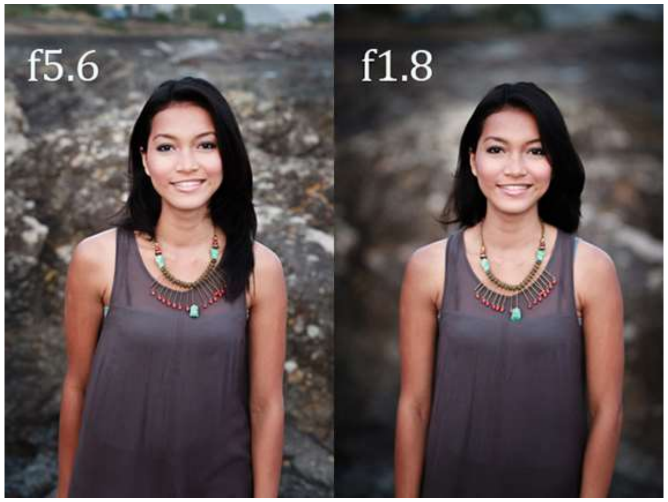
The Challenge –Image Deblurring

- Long exposure time causing camera shake (astronomy).
- Lens aberrations (wide apertures).
- Atmospheric turbulence (remote sensing).
- Fast moving object (sports, highway roads).
- Out of focus scene (small object photography).
- Combination of all the above with noise such as low light conditions.
- All the above are examples of inverse problem, meaning that both the “true” image and the noise are unknown: $y = k * x + n$.

K is the blur kernel, x original image and n is added noise.

Examples of blurry images

Narrow aperture vs wide aperture.

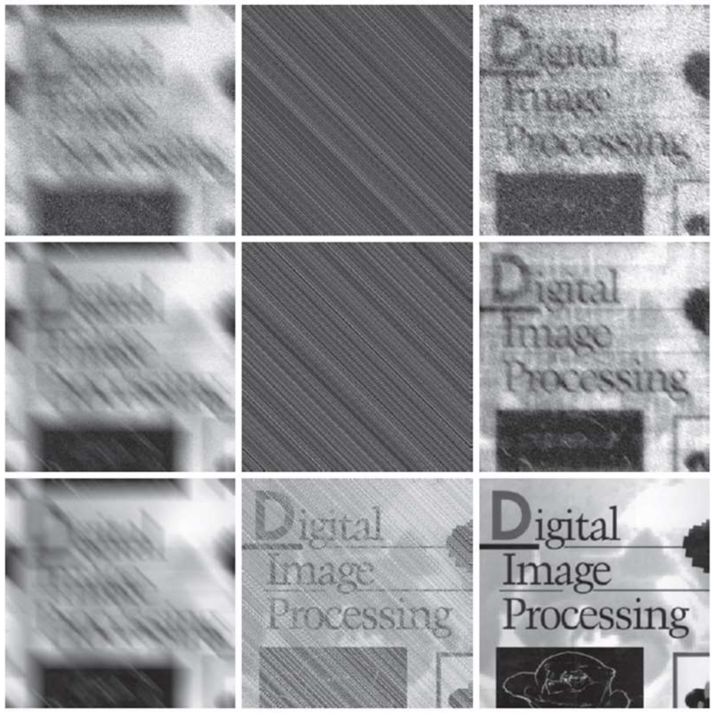


Long exposure

Motion blur

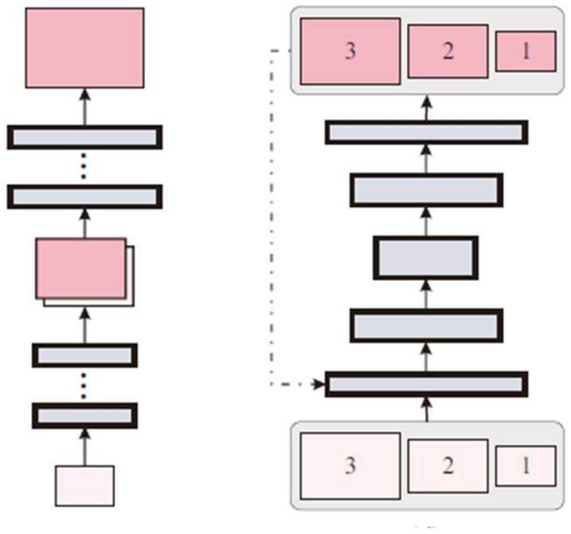
Methods of Deblurring

- Filtering (Inverse, Wiener, etc.).
- Blind deconvolution with NN as blur estimator (restricted by different types of blur).
- NN to inversely filter blurry images (learned inverse filter to deblurr text for license plats).
- Other learning-based approaches for blind deconvolution were used for deconvolution solution for a single image.
- neural network to learn a general procedure that is directly applicable to different images and blurs.

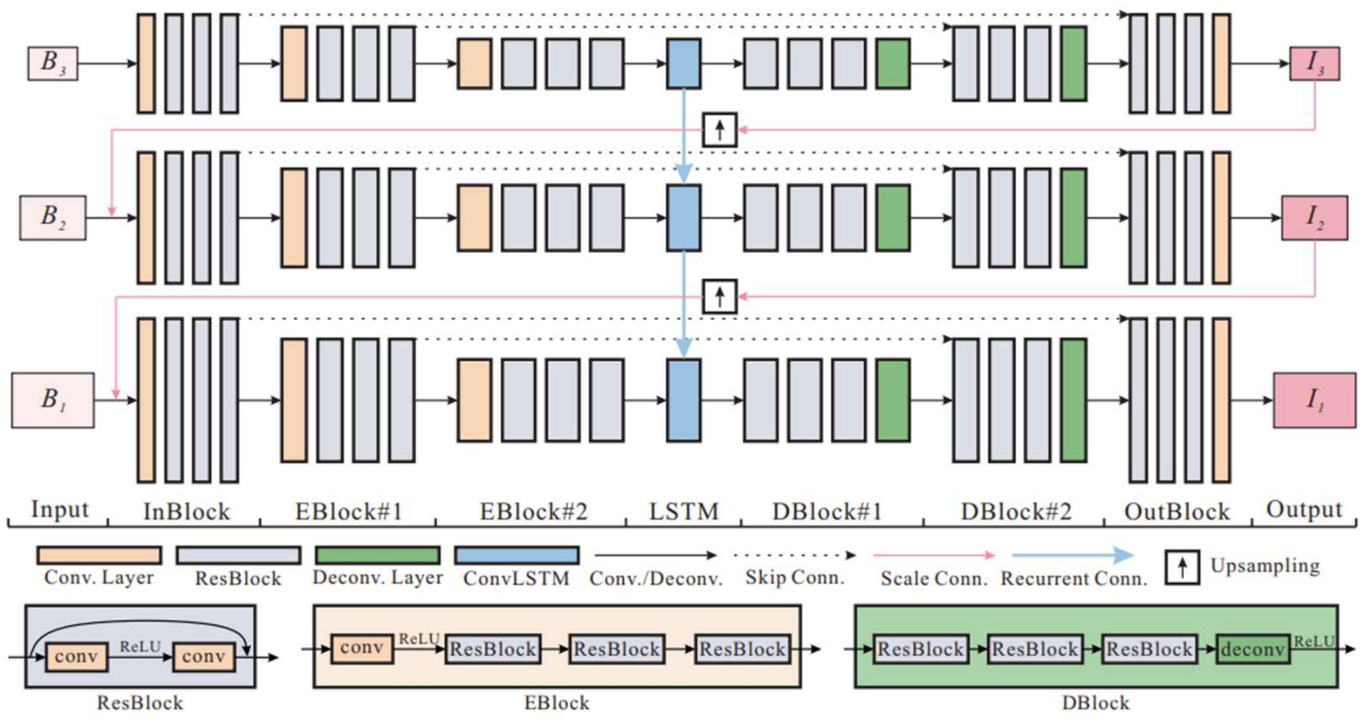


The Proposed Method

- A Coarse-to-Fine pipeline.
- What is multi-scale CNN?
- What is Scale-recurrent?
- Use of ResBlock (residual)
- Encoder-decoder with ResBlocks
- RNN with long-short term memory (LSTM).



The Network Outline



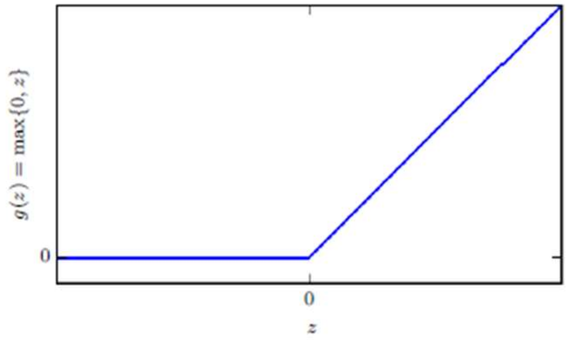
The NN input – output

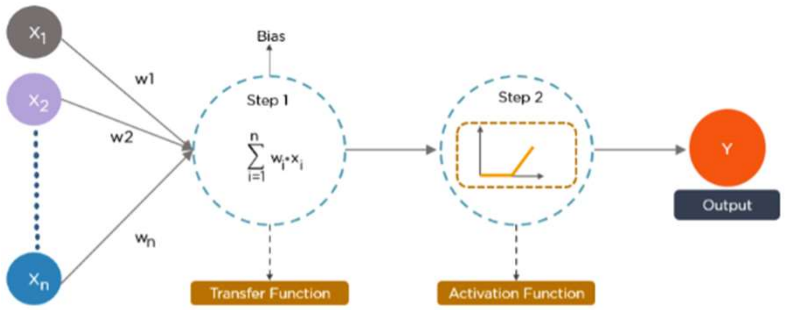
- The input of the network is a sequence of blurry randomly cropped images.
- The images are down-sampled from the original at different scales
- The output is a set of the corresponding inputs as sharp images at different scales
- The sharp full resolution image is the final output
- Blurry image are fed into the NN, the network outputs an initial up-sampled deblurred result from previous scale as input for estimated sharp image as:

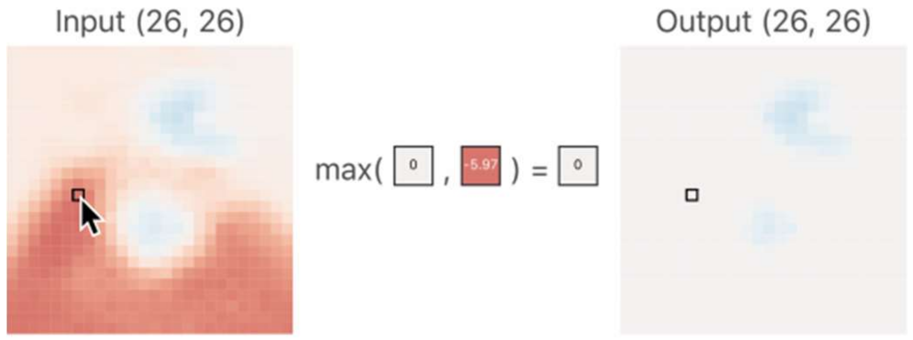
$$\mathrm{I}^{\mathrm{i}}, \mathrm{h}^{\mathrm{i}} = \mathrm{Net}_{\mathrm{SR}}\left(\mathrm{B}^{\mathrm{i}}, \mathrm{I}^{\mathrm{i+1\uparrow}}, \mathrm{h}^{\mathrm{i+1\uparrow}}; \theta_{SR}\right)$$

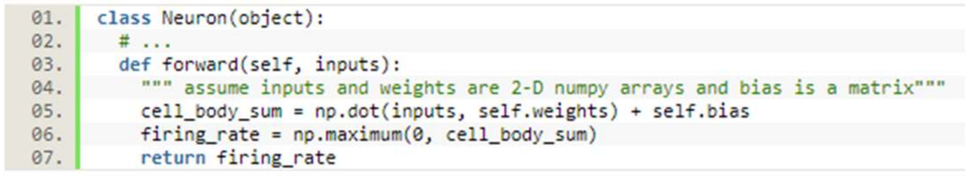
Activation function

- The network activation function is ReLU.









```
class Neuron(object):
  # ...
  def forward(self, inputs):
    """ assume inputs and weights are 2-D numpy arrays and bias is a matrix"""
    cell_body_sum = np.dot(inputs, self.weights) + self.bias
    firing_rate = np.maximum(0, cell_body_sum)
    return firing_rate
```

Parameter update – Solver

- The network training solver, updating the kernels parameters is Adam. The loss function is simple L_2-norm = $L_2 = \sum_{i=1}^{n} \frac{k_i}{N_i} \left\| I^i - I^i_{true} \right\|_2^2$
- Simplest form of a solver: x is a parameter vector, dx is the update gradient. So, $\mathrm{x} \mathrel{+}= -\mathrm{learning\_rate} * dx$.
- For this research, a second order method is used, called Adam:

$$\mathrm{x} \mathrel{+}= -\mathrm{learning\_rate} * \mathrm{m} / (\mathrm{np.sqrt(v)} + \mathrm{eps})$$

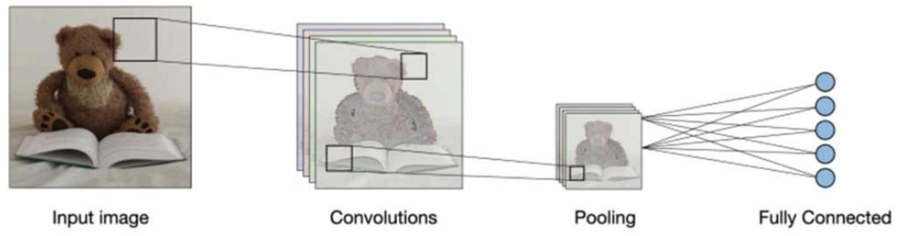
$$\mathrm{v} = \mathrm{beta2} * \mathrm{v} + (1 - \mathrm{beta2}) * (\mathrm{dx}^2)$$

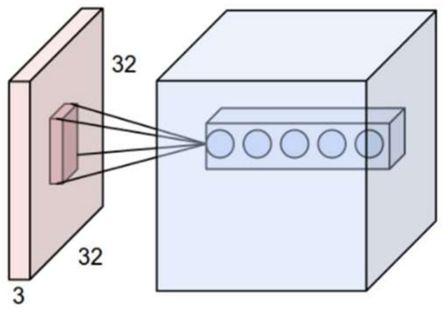
$$\mathrm{m} = \mathrm{beta1} * \mathrm{m} + (1 - \mathrm{beta1}) * \mathrm{dx}$$

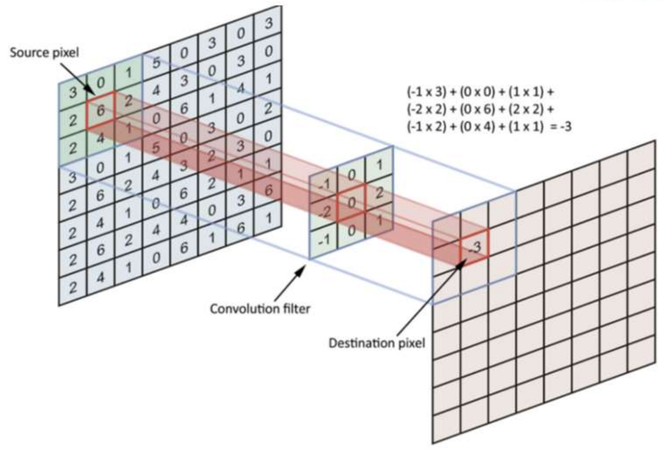
- While beta1=0.9, beta2=0.999 and smoothing factor $\epsilon = 10^{-8}$
- The learning rate is decaying as: $\mathrm{learning\_rate} = \alpha = \alpha_0 * e^{-kt}$, where k is 0.3 and t is the iteration number.
- In each iteration a batch of 16 blurry images are sampled. The number of epoch the network sees the data is 2000.

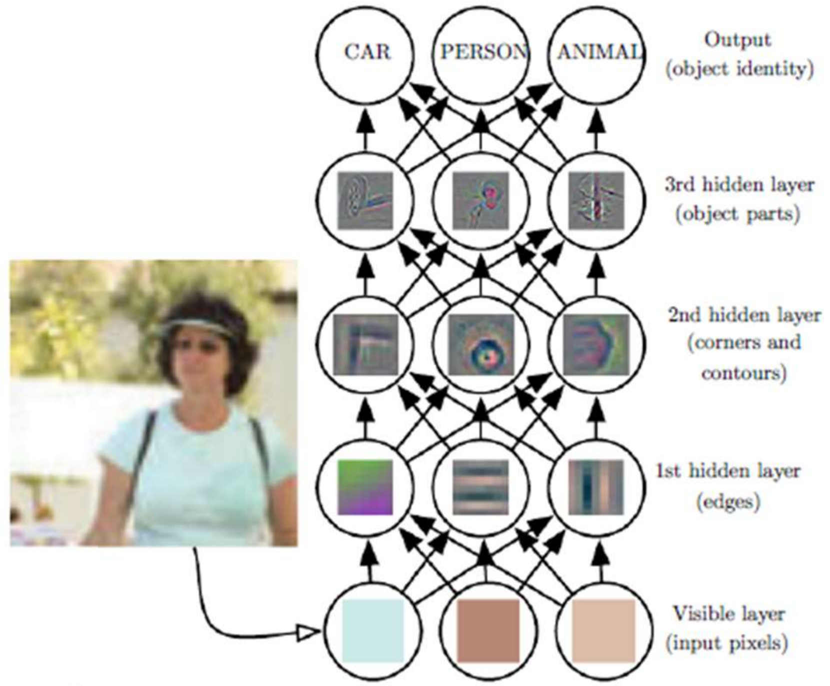
Basics of CNN, RNN, LSTM and ConvLSTM

- CNN



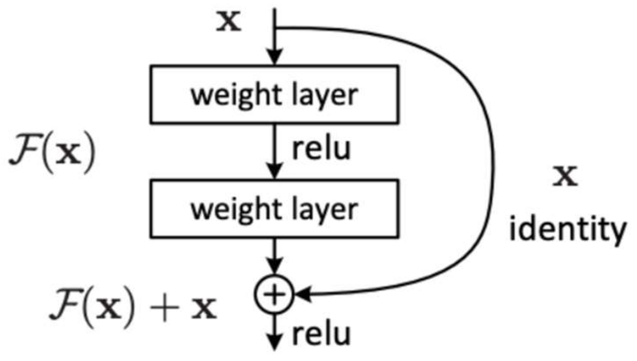


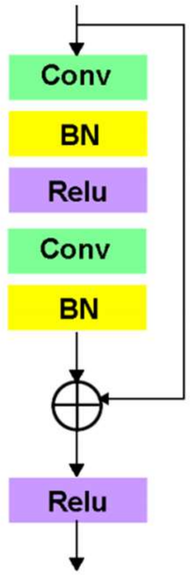

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

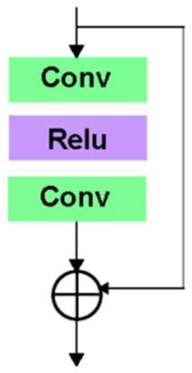

CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

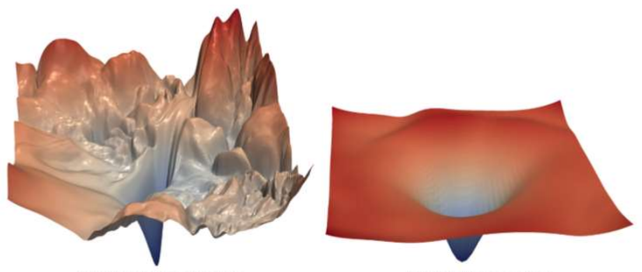
ResBLock and skip-connections

Output = x +Conv2(Conv1(x))







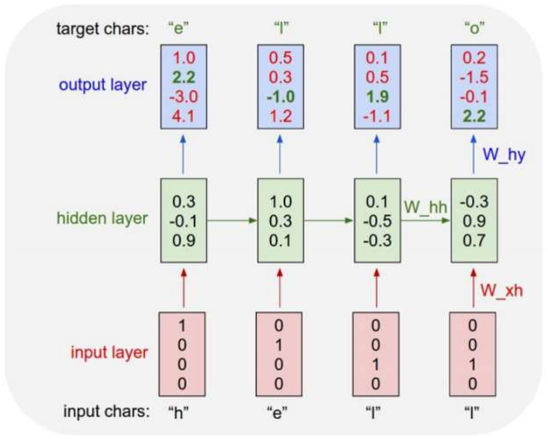

(a) without skip connections   (b) with skip connections

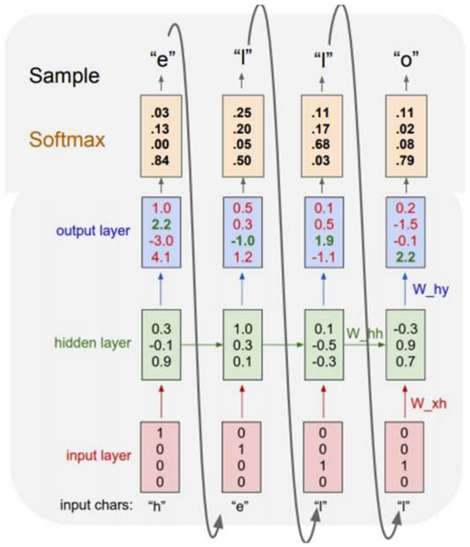
RNN

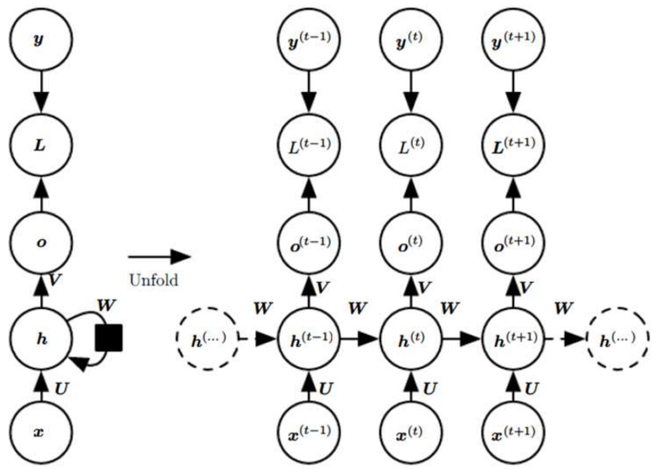
- Sequence to sequence translation is most often done with RNN (speech recognition, image sequence).
    - Make use of not only the input, but the last prediction is also an input.
    - U is input-to-hidden connection, V hidden-to-output and W hidden-to-hidden.

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}),$$

$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}$$

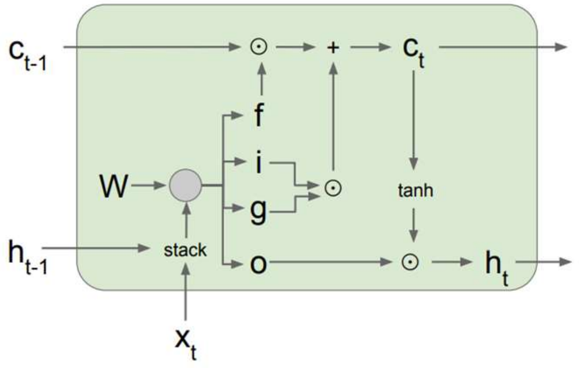


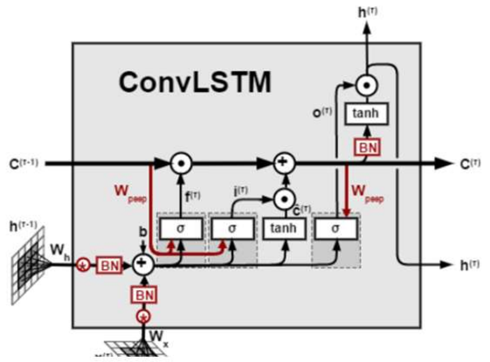


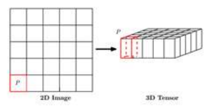

y
L
o
h
x
V
W
U
Unfold
$y^{(t-1)}$ $y^{(t)}$ $y^{(t+1)}$
$L^{(t-1)}$ $L^{(t)}$ $L^{(t+1)}$
$o^{(t-1)}$ $o^{(t)}$ $o^{(t+1)}$
$h^{(\dots)}$ $h^{(t-1)}$ $h^{(t)}$ $h^{(t+1)}$ $h^{(\dots)}$
$x^{(t-1)}$ $x^{(t)}$ $x^{(t+1)}$

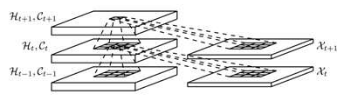
LSTM and ConvLSTM

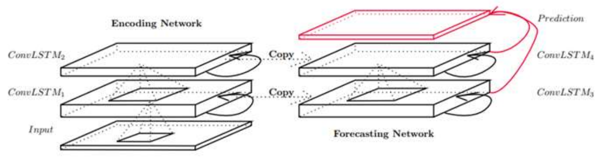
- The authors made use of the ConvLSTM.
- Originally design to solve the vanishing gradient problem.

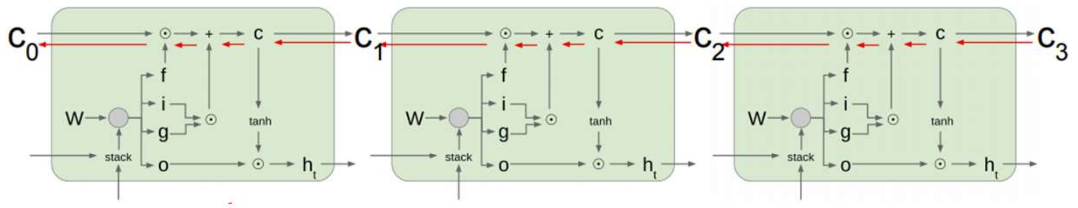


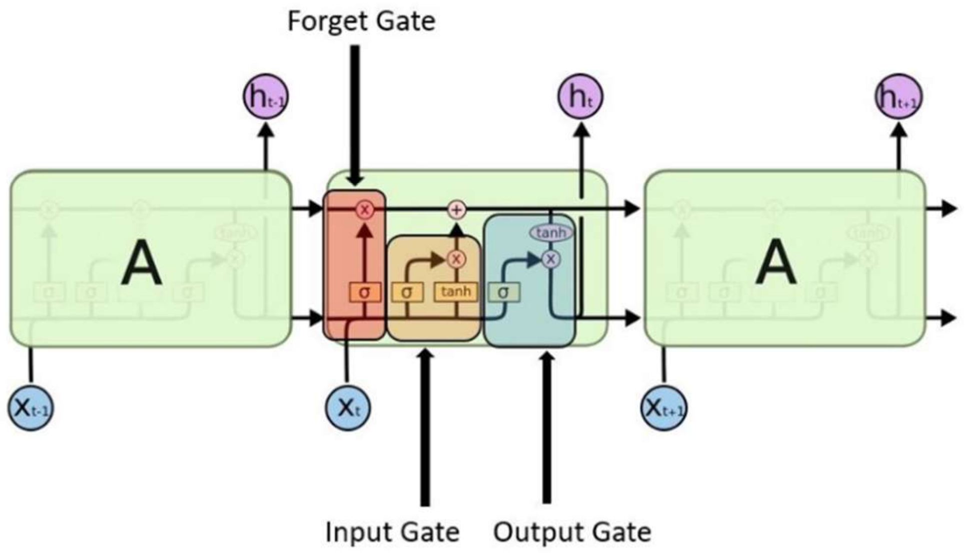










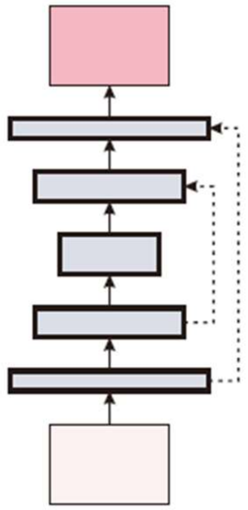


Deconvolution layer - (Transposed Convolutions)

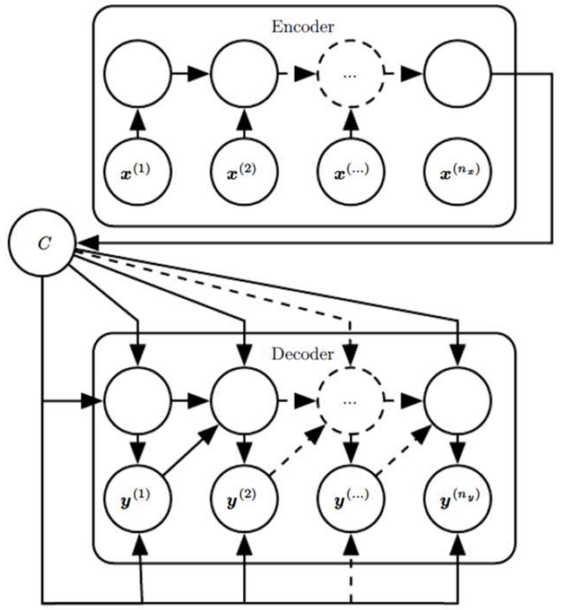
- Not an actual deconvolution, it’s a transposed convolution layer.
- So, this layer do not actually reverse the process, its merely reconstructs the spatial resolution from before and performs a convolution.
- This may not be the mathematical inverse, but for Encoder-Decoder architectures, it’s still very helpful. This way we can combine the upscaling of an image with a convolution, instead of doing two separate processes.

Encoder-Decoder

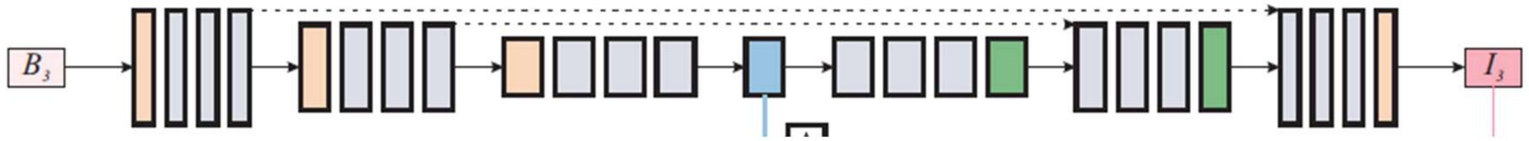
- Encoder-decoder as mentioned before, refers to a symmetric CNN structure that transform input data into feature maps with converging spatial size in the encoder and then transformed back to original size using the decoder.
- Encoder-decoder can accept a sequence of certain length and output another length.
- For this work task, a slight change is needed. For deblurring there is a need for a large receptive field to handle severe motions. This is usually accomplished by stacking more levels.
- This is not ideal in practice since it increase the number of parameters. Also, the feature maps are too small. Another reason is a slow convergences of the network.



Encoder-Decoder ResBlock



- This work Encoder-Decoder uses ResBlocks without batch normalization. The explanation relays solely on experimentation.
- The encoder contains one convolution layer followed by serval ResBlocks. The stride for the convolution is 2, so the feature map is down sampled by half and number of kernels is doubled.
- The decoder contains several ResBlocks followed by “deconvolution” layer that meant to double the spatial size of the feature maps and reduce the number of channels by half. Stride is also 1.



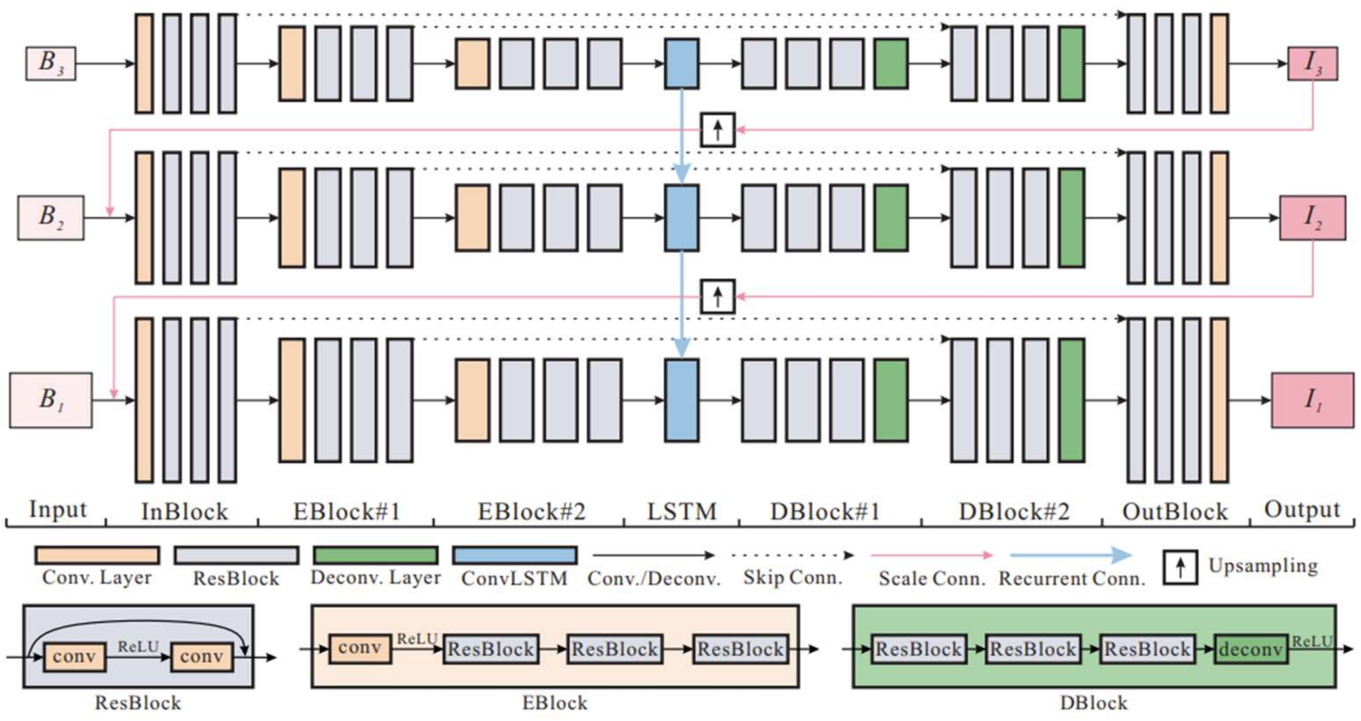
Scale connection

- The output of the last scale is now used as input (up-sampled) with the new scale input.
- A scale connection is used to connect between the bottle neck of each scale and the scale that follows.
- All convolution kernels are 5x5.
- The modified Encoder-decoder networks can be expressed as:

$$f^i = Net_E\left(B^i, I^{i+1\uparrow}; \theta_E\right)$$
$$h^i, g^i = ConvLSTM\left(h^{i+1\uparrow}, f^i; \theta_E\right)$$
$$I^i = Net_D\left(g^i; \theta_D\right)$$

All Together Now

$$f^i = Net_E(B^i, I^{i+1\uparrow}; \theta_E)$$
$$h^i, g^i = ConvLSTM(h^{i+1\uparrow}, f^i; \theta_E)$$
$$I^i = Net_D(g^i; \theta_D)$$



Data used

- Early learning-based methods used blurred images from convolution of images with blur kernel.
- The problem is that the real-world data is different, more complex.
- The dataset used in this work is made from averaging consecutive short-exposure frames from videos of high-speed camera.
- The camera used is GoPro Hero 4, averaging frames is similar to long-exposure. This can simulate complex camera movement and object motion. Which is more common in real world data.
- The resulting dataset contains 3214 blurry/clear image pairs. 2103 are used for training and the remaining 1111 are used for evaluation.

Model training

- As said before, the solver is Adam with beta1=0.9, beta2=0.999, epsilon=10^{-8}.
- Learning rate is exponentially decaying from 0.0001 to $1e^{-6}$ at 2000 epochs using 0.3 power.
- The batch size in each iteration is 16 blurry images that are randomly cropped to 256x256 pixels as training input, the same goes for the ground truth.
- All trainable parameters are initialized with Xavier method.

Quality measurement

- Peak signal-to-noise ratio (PSNR).

For a grey level image (8-bits), f is the “clean image” and g is the image to test:

$PSNR(f,g) = 10\log_{10}\left(255^2/MSE(f,g)\right)$ where $MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f_{ij} - g_{ij}\right)^2$

So, the higher the PSNR, the better the signal to noise ratio because the MSE is smaller (better image quality). Small PSNR can imply high numerical differences.

- structural similarity index measure (SSIM), good for measuring similarity between two images. Considered to be a good quality estimate for human visual system.

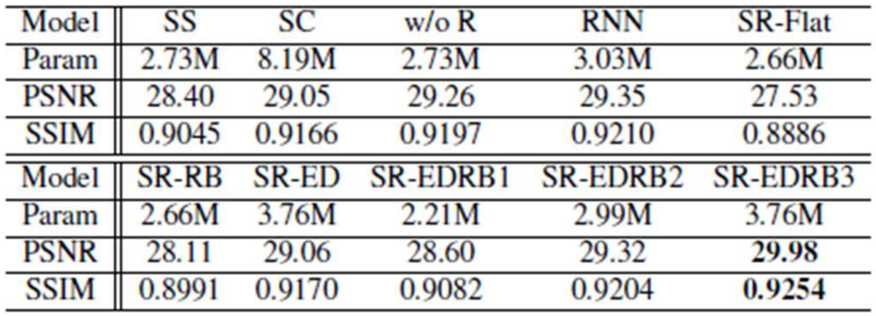
$$SSIM(f,g) = l(f,g)c(f,g)s(f,g)$$

$$l(f,g) = \frac{2\mu_f\mu_g + C_1}{\mu_f^2 + \mu_g^2 + C_1} \qquad c(f,g) = \frac{2\sigma_f\sigma_g + C_2}{\sigma_f^2 + \sigma_g^2 + C_2} \qquad s(f,g) = \frac{\sigma_{fg} + C_3}{\sigma_f\sigma_g + C_3}$$

Experiments

- The searchers compared the final NN architecture to others by means of PSNR and SSIM.
- A single scale (SS) NN was tested, the recurrent connection are replaced by one convolution layer.
- SC (single scale) network composed of the same SS-NN at three scales but independently.
- A Model without recurrent modules (w/oR) in bottleneck layer, therefor is a shared weight version of SC.
- Model RNN uses vanilla RNN rather then ConvLSTM.
- Models named SR-EDRB are models that use 1,2 and 3 ResBlock.
- Model SR-Flat replace encoder-decoder with a flat convolution of 43 layers.
- Model SR-RB replaces all Encoder and Decoder blocks with ResBlocks without any stride.
- Model SR-ED uses original encoder-decoder with ResBlocks replaced by 2 convolution layers.

Table 1. Quantitative results of the baseline models.



Model	SS	SC	w/o R	RNN	SR-Flat
Param	2.73M	8.19M	2.73M	3.03M	2.66M
PSNR	28.40	29.05	29.26	29.35	27.53
SSIM	0.9045	0.9166	0.9197	0.9210	0.8886
Model	**SR-RB**	**SR-ED**	**SR-EDRB1**	**SR-EDRB2**	**SR-EDRB3**
Param	2.66M	3.76M	2.21M	2.99M	3.76M
PSNR	28.11	29.06	28.60	29.32	**29.98**
SSIM	0.8991	0.9170	0.9082	0.9204	**0.9254**

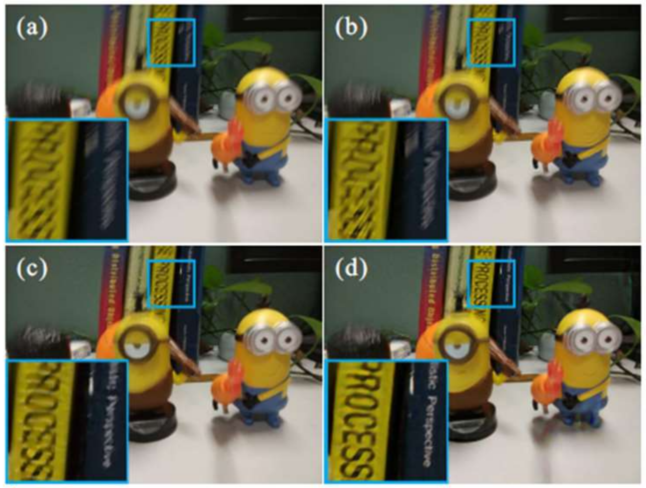
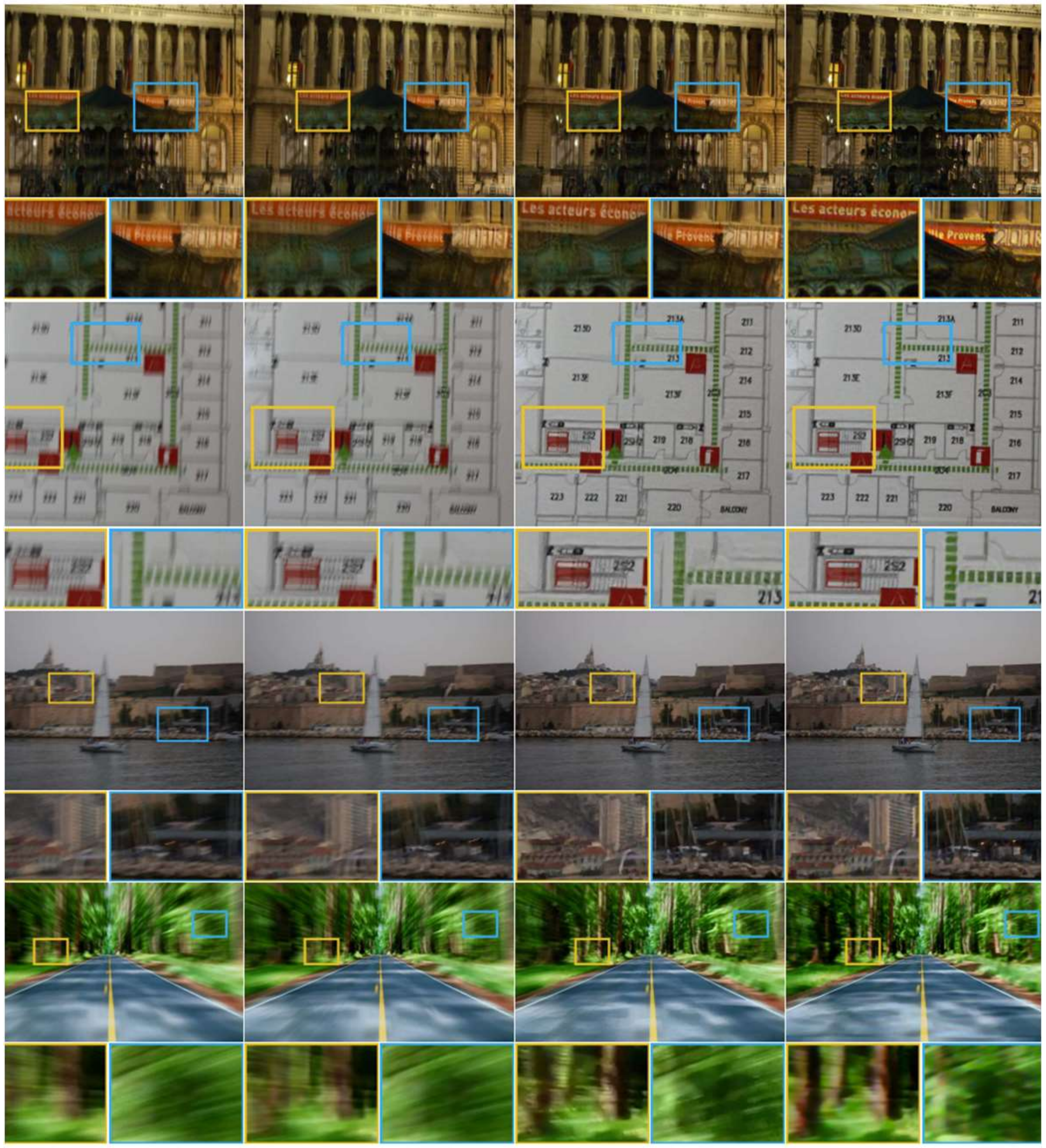
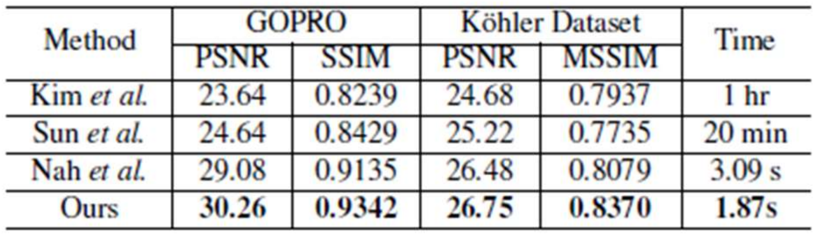


Figure 1. **One real example.** (a) Input blurred image. (b) Result of Sun *et al.* [34]. (c) Result of Nah *et al.* [25]. (d) Our result.


(a) Input (b) Sun et al. (c) Nah et al. (d) Ours

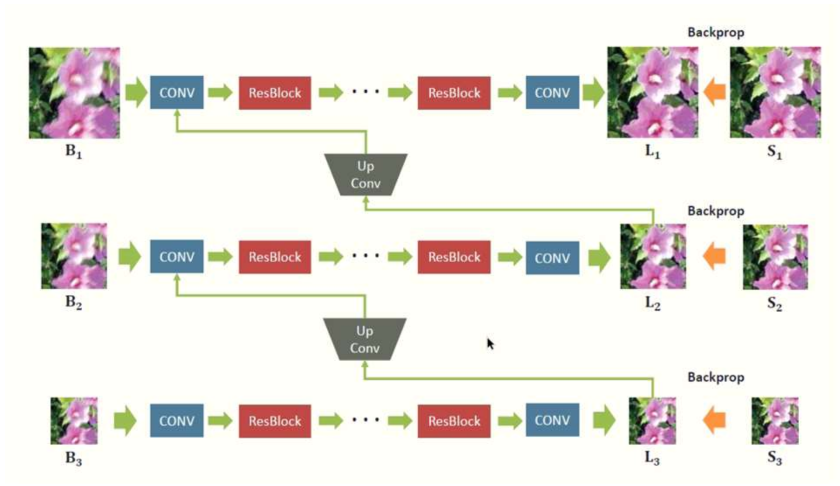
Real-world images tested

Comparison to models and other studies



Method	GOPRO		Köhler Dataset		Time
	PSNR	SSIM	PSNR	MSSIM	
Kim *et al.*	23.64	0.8239	24.68	0.7937	1 hr
Sun *et al.*	24.64	0.8429	25.22	0.7735	20 min
Nah *et al.*	29.08	0.9135	26.48	0.8079	3.09 s
Ours	**30.26**	**0.9342**	**26.75**	**0.8370**	**1.87s**

- Since the method deals with both general camera shake and object motion. The authors did not compare it to traditional uniform methods.
- Sun *et al* estimated blur kernels using CNN and used traditional deconvolution methods to recover sharp images.
- Nah *et al* used a Multi-scale network with ResBlocks.



Conclusion and recent progress

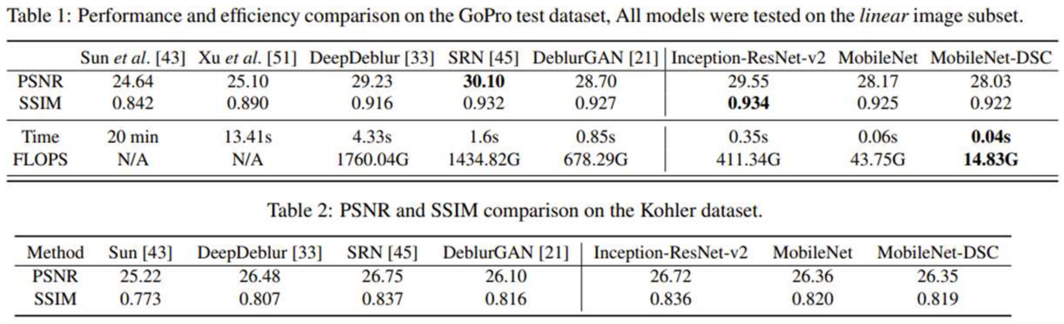


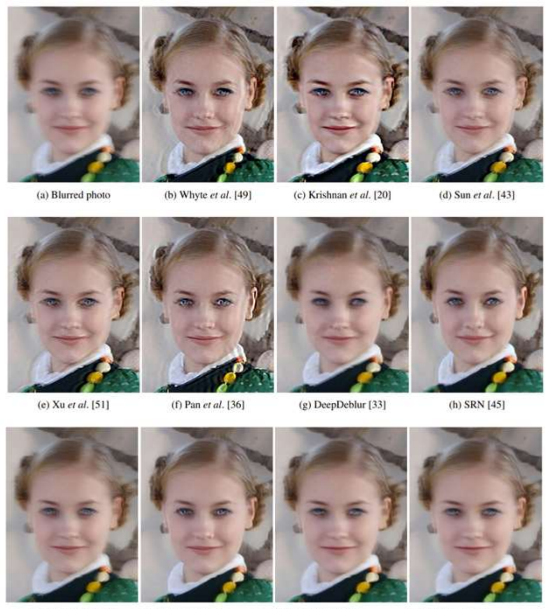
Table 1: Performance and efficiency comparison on the GoPro test dataset, All models were tested on the *linear* image subset.

	Sun *et al.* [43]	Xu *et al.* [51]	DeepDeblur [33]	SRN [45]	DeblurGAN [21]	Inception-ResNet-v2	MobileNet	MobileNet-DSC
PSNR	24.64	25.10	29.23	**30.10**	28.70	29.55	28.17	28.03
SSIM	0.842	0.890	0.916	0.932	0.927	**0.934**	0.925	0.922
Time	20 min	13.41s	4.33s	1.6s	0.85s	0.35s	0.06s	**0.04s**
FLOPS	N/A	N/A	1760.04G	1434.82G	678.29G	411.34G	43.75G	**14.83G**

Table 2: PSNR and SSIM comparison on the Kohler dataset.

Method	Sun [43]	DeepDeblur [33]	SRN [45]	DeblurGAN [21]	Inception-ResNet-v2	MobileNet	MobileNet-DSC
PSNR	25.22	26.48	26.75	26.10	26.72	26.36	26.35
SSIM	0.773	0.807	0.837	0.816	0.836	0.820	0.819

- The authors demonstrated a novel method which seems to work quite well on “real” world data as well.
- Articles from 2020 are deblurring with GANs and still comparing results to this study.
- Recent study claims to achieve better results out of this study using the concept of “Blur More To Deblur Better” (PSNR of 31.12 compared to 29.52).



(a) Blurred photo (b) Whyte *et al.* [49] (c) Krishnan *et al.* [20] (d) Sun *et al.* [43]

(e) Xu *et al.* [51] (f) Pan *et al.* [36] (g) DeepDeblur [33] (h) SRN [45]

(i) DeblurGAN [21] (j) DeblurGAN-v2 (Inception-ResNet-v2) **[Best visual quality]** (k) DeblurGAN-v2 (MobileNet) **[High efficiency]** (l) DeblurGAN-v2 (MobileNet-DSC) **[Highest efficiency]**

References

- Tao, Xin, et al. "Scale-recurrent network for deep image deblurring." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- Shi, Xingjian, et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." *Advances in neural information processing systems* 28 (2015): 802-810.
- S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. pages.2017 ,3891–3883
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE,2016.
- X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, pages 2802–2810,2016.
- Goodfellow, Ian, et al. Deep learning. Vol. 1. No. 2. Cambridge: MIT press, 2016.

Man in swan tent photographing swans