

# CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE RESTORATION

Giacomo Boracchi,

Politecnico di Milano, DEIB.

<https://boracchi.faculty.polimi.it/>

February 7<sup>th</sup>, 2022

Advanced Deep Learning Models and Methods  
PhD course

# **CNN FOR IMAGE DENOISING**

Thanks Edoardo Peretti (our thesis student) for helping out in the literature survey

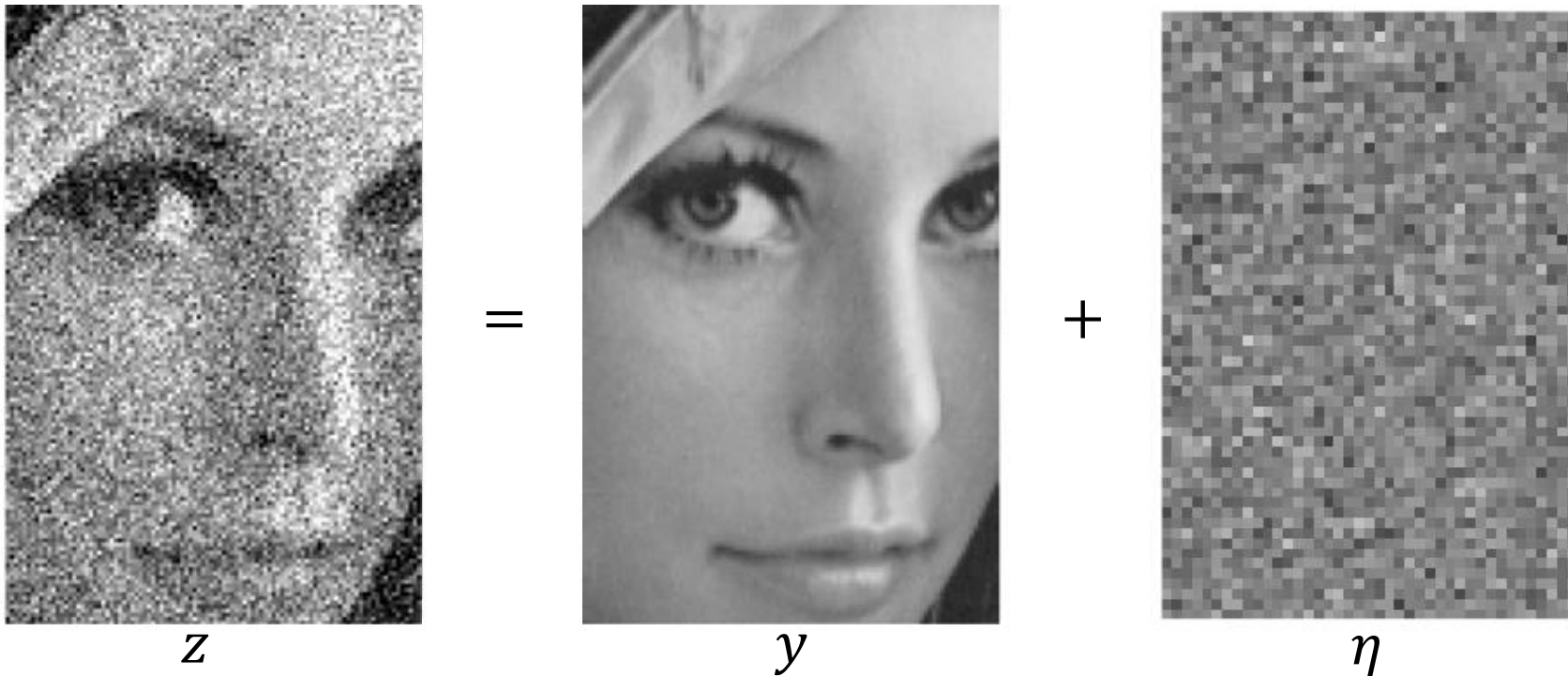
## OBSERVATION MODEL

$$z(x) = y(x) + \eta(x), x \in X$$

$z : X \rightarrow \mathbb{R}$  observed noisy image

$y : X \rightarrow \mathbb{R}$  unknown original image (grayscale)

$\eta : X \rightarrow \mathbb{R}$  i.i.d. Gaussian white noise,  $\eta \sim N(0, \sigma^2)$

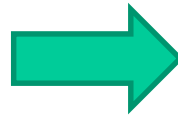


## GOAL OF IMAGE DENOISING

The purpose of any **denoising** algorithm is to provide  $\hat{y}$ , an estimate of the original image  $y$ .



$z$



$\hat{y}$

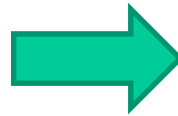


## GOAL OF IMAGE DENOISING

And the same works for RGB images



$z$



$\hat{y}$

# IMAGE DENOISING

Denoising plays a crucial role in many stages of imaging pipelines:

- **Preprocessing:** to enhance output quality and improve the effectiveness of subsequent algorithms
- **Post-processing:** to remove compression artifacts (e.g. blocks and ringing)
- **Plug-and-play filter:** as an implicit regularization prior in various imaging applications

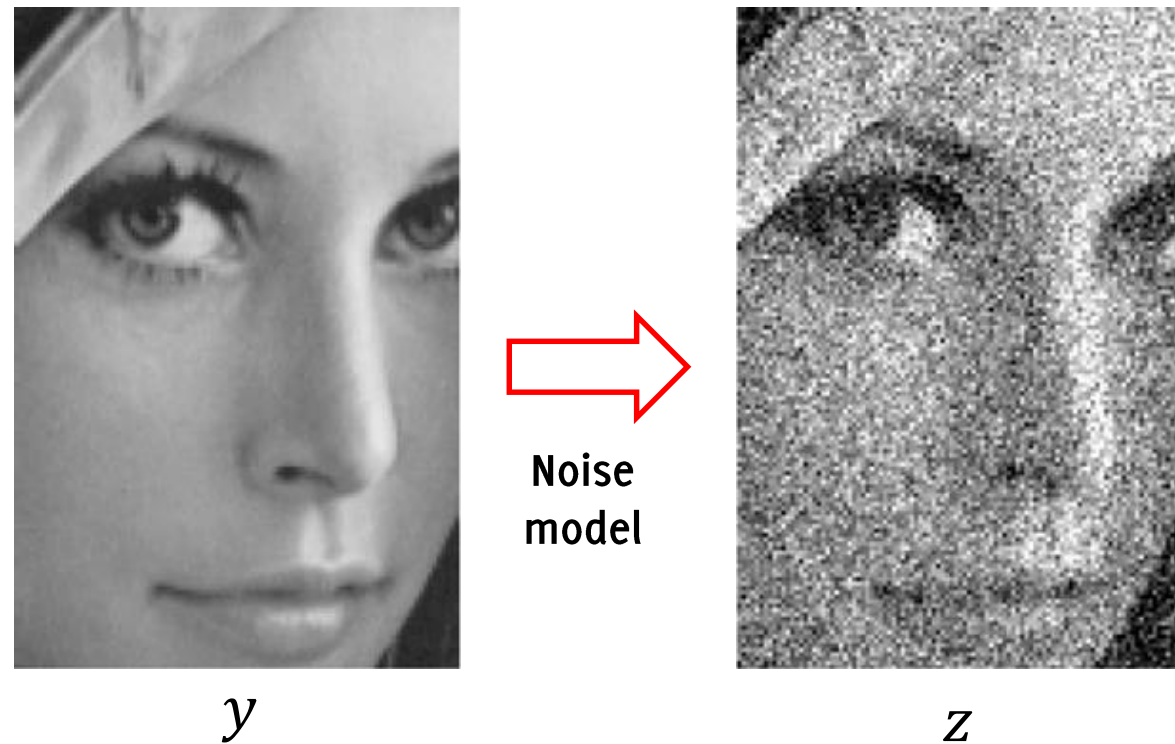
## STRAIGHTFORWARD SOLUTION

**NN architecture:** a CNN having the same input and output size, like those for semantic segmentation.

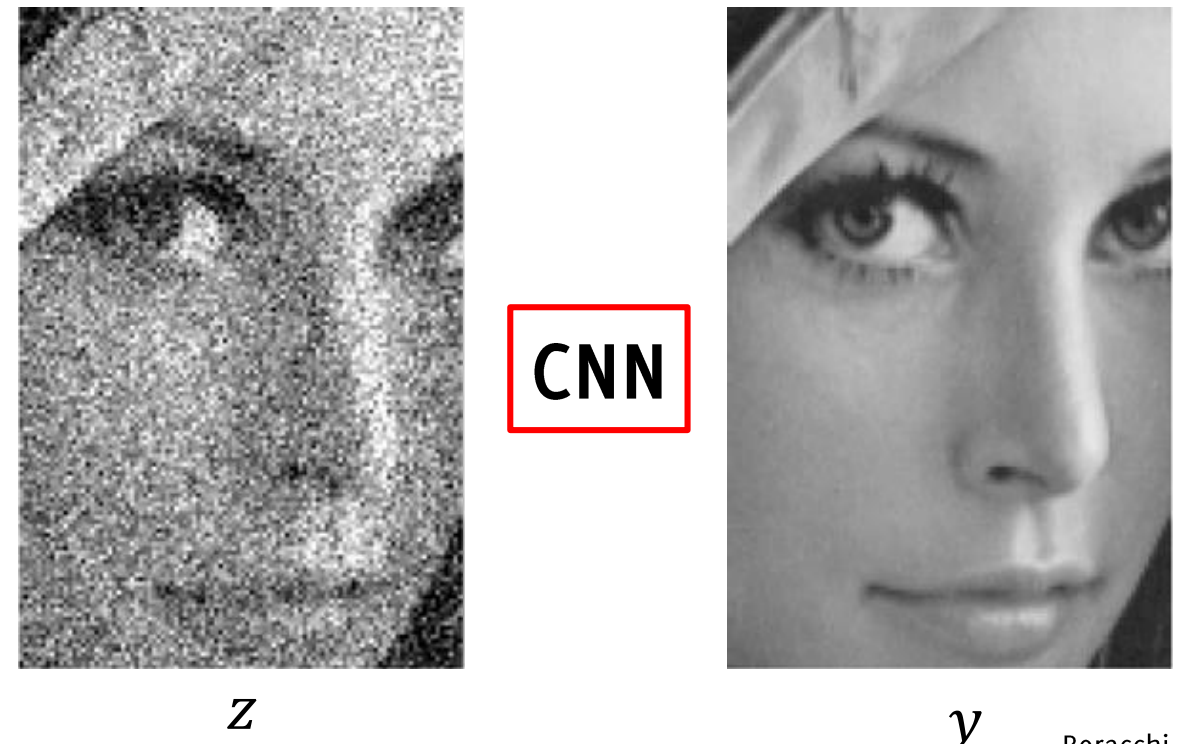
**Training data:**  $TR = \{(y, z), \text{ being } y \text{ a natural image}\}$

Easy to gather large  $TR$ , given the forward model to generate noisy images.

Forward pass, ruled by physical laws / sensor



Inverse pass, ill-posed, unknown





## STRAIGHTFORWARD SOLUTION

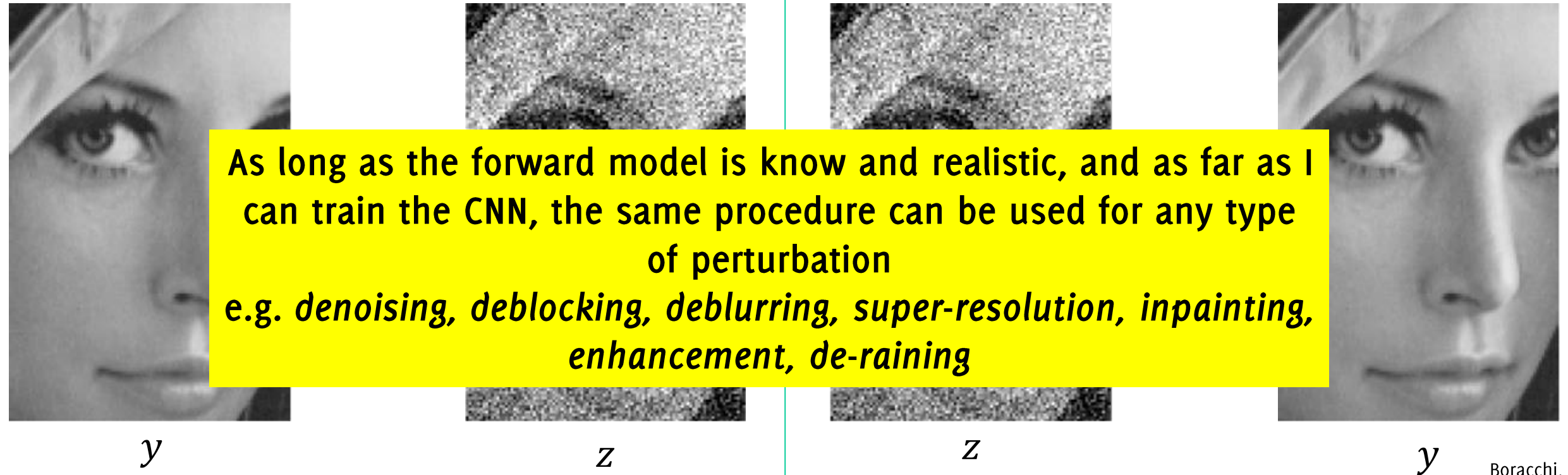
**NN architecture:** a CNN having the same input and output size, like those for semantic segmentation.

**Training data:**  $TR = \{(y, z), \text{ being } y \text{ a natural image}\}$

Easy to gather large  $TR$ , given the forward model to generate noisy images.

Forward pass, ruled by physical laws / sensor

Inverse pass, ill-posed, unknown



## IMAGE DENOISING BY CNNs

We will consider this a reference problem.

› *Is denoising by deep neural network a Supervised or Unsupervised Learning problem?*

- It is an unsupervised problem since no label / annotation is required for training.
- The loss function however is typical of supervised learning, say the MSE

$$\|\hat{y} - y\|_2$$

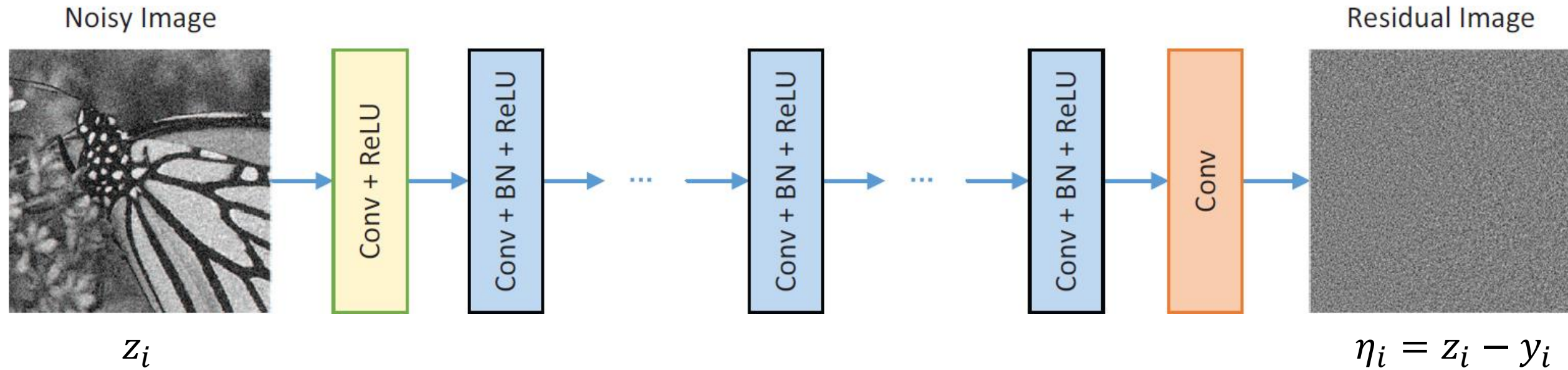
**Weight optimization** by reducing a loss function has made a substantial difference w.r.t. existing denoising algorithm, which were primarily designed over statistical modeling of the input and the noise.

# **SIMPLE DEEP NNS FOR IMAGE DENOISING**

# Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising

Kai Zhang, Wangmeng Zuo, *Senior Member, IEEE*, Yunjin Chen, Deyu Meng,  
and Lei Zhang, *Senior Member, IEEE*

## DN-CNN: THE ARCHITECTURE (FCNN)





## RESIDUAL LEARNING

The network is trained to predict the noise, rather than the noise-free image

$$TR = \{(z, \eta), \quad z = y + \eta\}$$

Thus the output will be

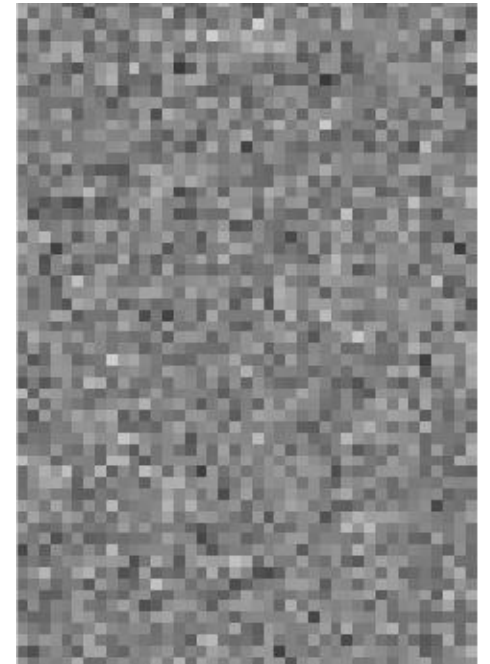
$$\hat{y} = z - \hat{\eta} = z - CNN(z)$$

It easier to optimize the loss function over the noise  $\|\eta - CNN(z)\|_2$  rather than on the image



$z$

**CNN**



$\eta$

# DNCNN ARCHITECTURE

## Network architecture:

- Conv+ReLU : 64 filters of size  $3 \times 3 \times c$  (layer 1)..
- Conv+BN+ReLU: 64 filters of size  $3 \times 3 \times 64$  (layer  $2 - d - 1$ ) + Batch Normalization
- Conv:  $c$  filters of size  $3 \times 3 \times 64$  are used to reconstruct the input

## Border artifacts:

zero padding outside the image before the processing

## DNCNN DETAILS

Loss to minimize

$$\ell(\Theta) = \frac{1}{2N} \sum_{i=1}^N \|CNN(z_i, \Theta) - (z_i - y_i)\|_F^2$$

- Assessing on how good the network can **estimate the noise realizations** in images
- **Training samples** are **easy to generate**: add synthetic corruption
- To ease the training process, the **loss is assessed on patches** cropped from noisy images.
- Since the network is fully convolutional, it can be applied to images of arbitrary size

## DNCNN DETAILS

### Training:

400 images of 180 x 180 pixels (larger training sets do not improve performance substantially)

**Network training is performed patch-wise**

- 40x40 patches for Gaussian denoising
- 50x50 patches for other denoising problems

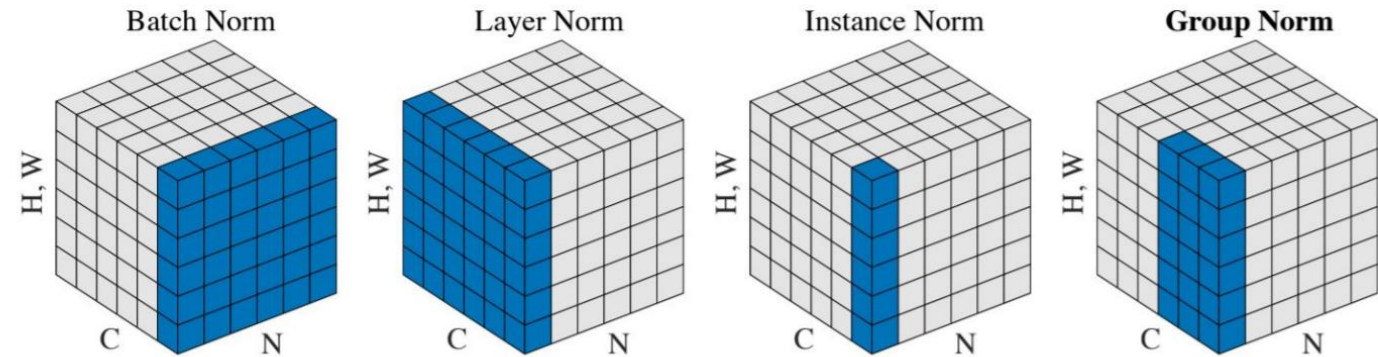
Repeat training when corrupting training images by different noise levels  $\sigma$

# BATCH NORMALIZATION

Consider a batch of activations  $\{x_i\}$ , the following transformation bring these to unit variance and zero mean

$$x'_i = \frac{x_i - E[x_i]}{\sqrt{\text{var}[x_i]}}$$

Where  $E[x_i]$  and  $\sqrt{\text{var}[x_i]}$  are computed from each batch and separately for each channel!



Wu and He, "Group Normalization", ECCV 2018

**Can we get more flexibility than zero-mean, unit variance?**

## BATCH NORMALIZATION

Batch normalization adds after standard normalization

$$x'_i = \frac{x_i - E[x_i]}{\sqrt{\text{var}[x_i]}}$$

a further a parametric transformation

$$y_{i,j} = \gamma_j x'_i + \beta_j$$

Where parameters  $\gamma$  and  $\beta$  are learnable scale and shift parameters.

**Rmk:** estimates  $E[x_i]$  and  $\sqrt{\text{var}[x_i]}$  are computed on each minibatch, need to be fixed after training. **After training, these are replaced by (running) averages of values seen during training.**

# BATCH NORMALIZATION

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots x_m\}$ ;

Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

**Algorithm 1:** Batch Normalizing Transform, applied to activation  $x$  over a mini-batch.

# BATCH NORMALIZATION

During testing batch normalization becomes a linear operator! Can be fused with the previous fully-connected or conv layer

In practice networks that use Batch Normalization are significantly more robust to bad initialization

Typically Batch Normalization is used in between FC layers of deep CNN, but sometimes also between Conv Layers



# BATCH NORMALIZATION

## Pros:

- Makes deep networks much easier to train!
- Improves gradient flow
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Zero overhead at test-time: can be fused with conv!

## Watch out:

- Behaves differently during training and testing: this is a very common source of bugs!

# DNCNN DETAILS

## Denoising task:

- Denoising additive white Gaussian noise
  - with a known variance
  - With unknown variance
- Denoising other artifacts:
  - bicubic upsampling (Super-resolution)
  - Jpeg compression

# **THE RECEPTIVE FIELD**

## **A VERY IMPORTANT ASPECT IN CNNs**

# THE RECEPTIVE FIELD

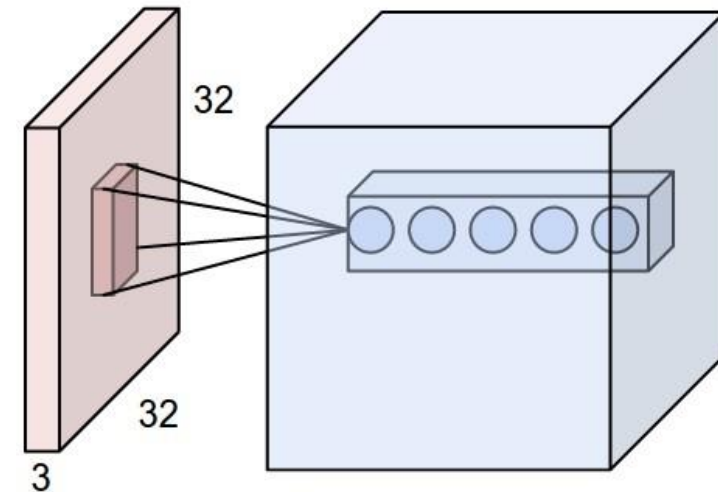
One of the basic concepts in deep CNNs.

**Due to sparse connectivity**, unlike in FC networks where the value of each output depends on the entire input, in **CNN an output only depends on a region of the input**.

**This region in the input is the receptive field for that output**

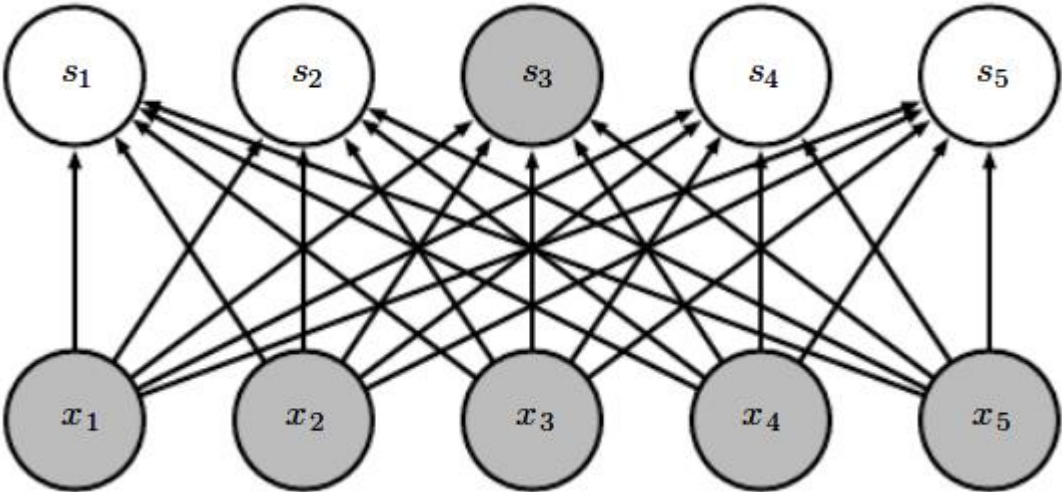
**The deeper you go, the wider the receptive field is:** maxpooling, convolutions and stride  $> 1$  increase the receptive field

**Usually, the receptive field refers to the final output unit of the network in relation to the network input, but the same definition holds for intermediate volumes**

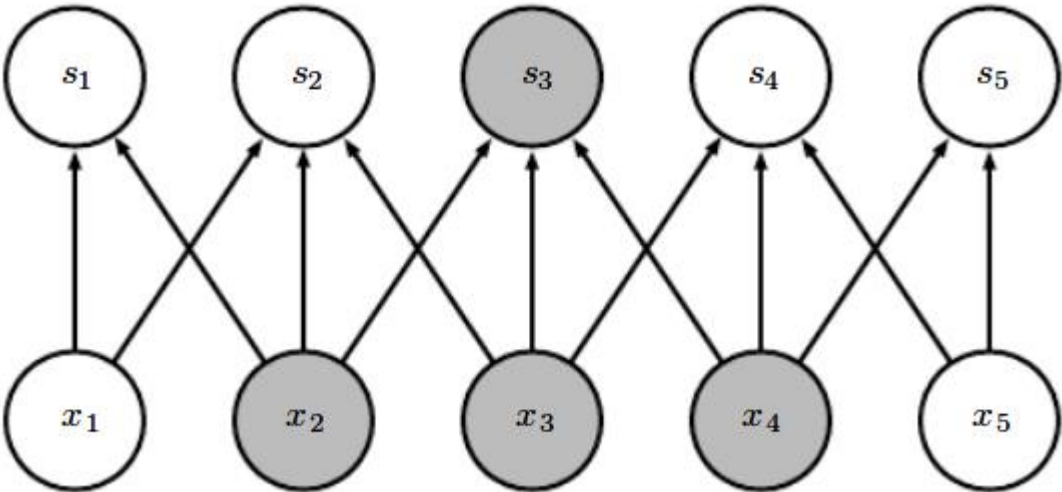


# RECEPTIVE FIELDS

Fully connected



3x1 convolutional

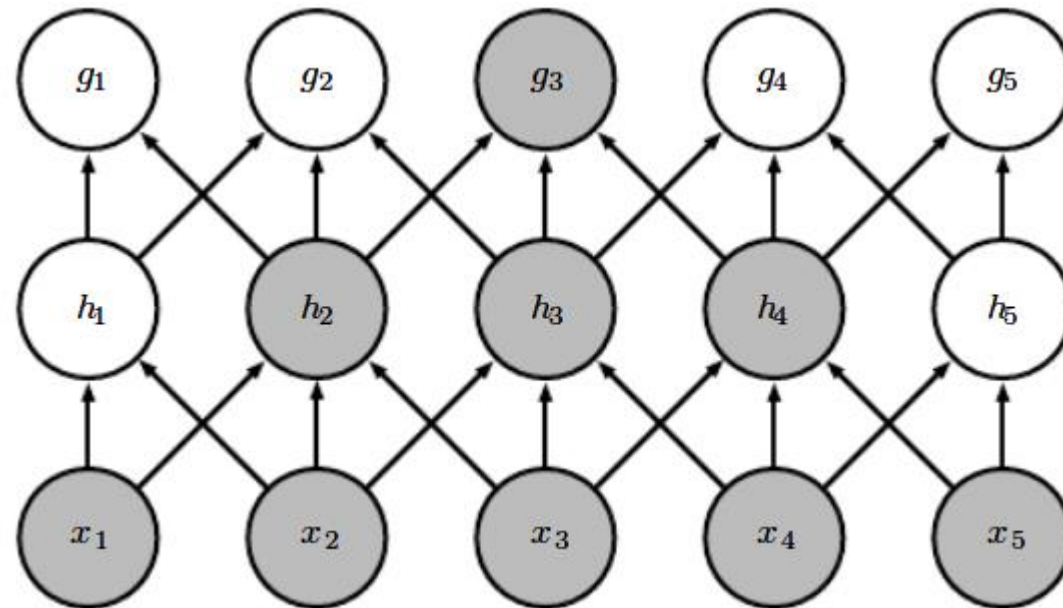


## RECEPTIVE FIELDS

Deeper neurons depend on wider patches of the input (convolution is enough to increase receptive field, no need of maxpooling)

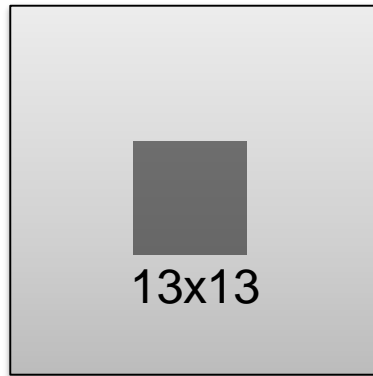
3x1 convolutional

3x1 convolutional



# RECEPTIVE FIELDS

Input



Conv 3x3

Conv 3x3

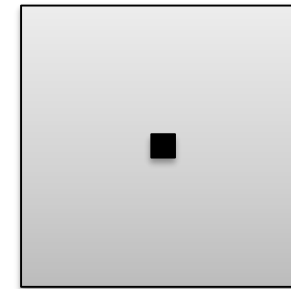
Conv 3x3

Conv 3x3

Conv 3x3

Conv 3x3

map



How large is the receptive field of the black neuron?



Conv 3x3

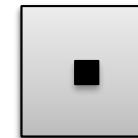
MP 2x2

Conv 3x3

MP 2x2

Conv 3x3

MP 2x2



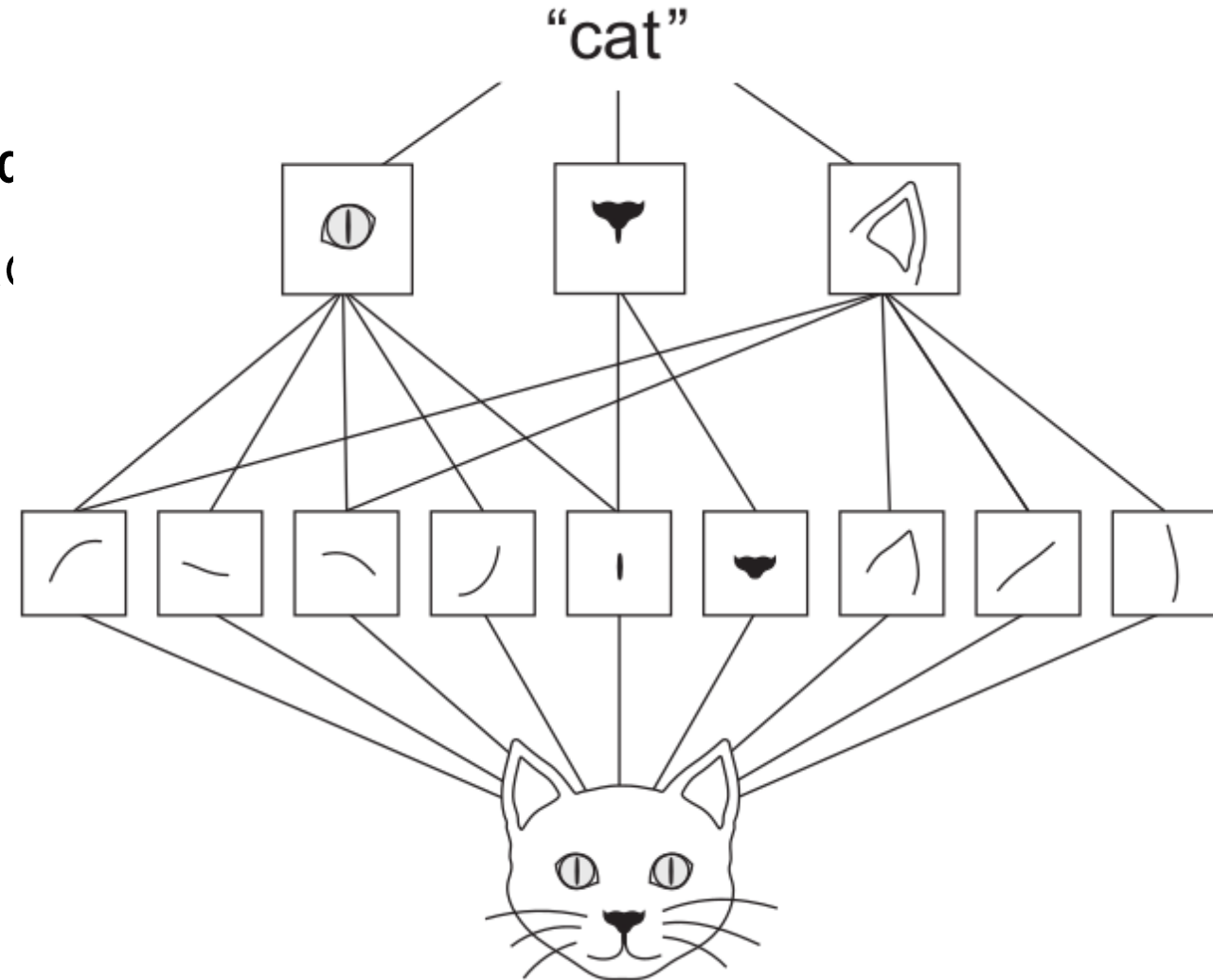
## AS WE MOVE DEEPER...

As we move to deeper layers:

- spatial resolution is reduced
- the number of maps increases

We search for **higher-level patterns**, and c

There are more high-level patterns than l





# DN-CNN DEPTH AND RECEPTIVE FIELD

## Network depth:

To be sized depending on the "receptive field", to make that comparable to receptive fields in image denoising algorithms.

Having only  $3 \times 3$  convolutions, the receptive field of a  $d$ -layered network is  $(2d + 1) \times (2d + 1)$

- $d = 17$  for Gaussian denoising (known variance)
- $d = 20$  for Gaussian denoising (unknown variance) and artefact removal (superresolution / jpeg)

TABLE I  
THE EFFECTIVE PATCH SIZES OF DIFFERENT METHODS WITH NOISE LEVEL  $\sigma = 25$ .

Methods	BM3D [2]	WNNM [13]	EPLL [33]	MLP [24]	CSF [14]	TNRD [16]
Effective Patch Size	$49 \times 49$	$361 \times 361$	$36 \times 36$	$47 \times 47$	$61 \times 61$	$61 \times 61$

## DNCNN LIMITATIONS

DnCNN learns a mapping function

$$z \rightarrow \mathcal{F}(\Theta_{\sigma}, z)$$

Where the network parameters  $\Theta_{\sigma}$  depend on  $\sigma$ , the noise standard deviation

DnCNN is hard to be deployed to images corrupted by:

- different noise levels.
- different noise models
- spatially variant / signal dependent noise.

# **LOSS FUNCTIONS FOR TRAINING A DENOISING CNN**

# Loss Functions for Image Restoration With Neural Networks

Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz

**Abstract**—Neural networks are becoming central in several areas of computer vision and image processing and different architectures have been proposed to solve specific problems. The impact of the loss layer of neural networks, however, has not received much attention in the context of image processing: the default and virtually only choice is  $\ell_2$ . In this paper, we bring attention to alternative choices for image restoration. In particular, we show the importance of perceptually-motivated losses when the resulting image is to be evaluated by a human observer. We compare the performance of several losses, and propose a novel, differentiable error function. We show that the quality of the results improves significantly with better loss functions, even when the network architecture is left unchanged.

**Index Terms**—Image processing, image restoration, neural networks, loss functions.

trying to fool networks with specific inputs [6]. Other advances were made on the techniques to improve the network's convergence [7].

The loss layer, despite being the effective driver of the network's learning, has attracted little attention within the image processing research community: the choice of the cost function generally defaults to the squared  $\ell_2$  norm of the error [3], [8]–[10]. This is understandable, given the many desirable properties this norm possesses. There is also a less well-founded, but just as relevant reason for the continued popularity of  $\ell_2$ : standard neural networks packages, such as Caffe [11], only offer the implementation for this metric.

However,  $\ell_2$  suffers from well-known limitations. For in-

# LOSS FUNCTIONS FOR RESTORATION

General loss for a patch:

$$\mathcal{L}^{\mathcal{E}}(P) = \frac{1}{N} \sum_{p \in P} \mathcal{E}(p)$$

The choice of the cost function generally defaults to the squared  $\ell_2$  norm of the error.

But,  $\ell_2$  **correlates poorly with image quality as perceived by a human observer.**

- E.g.  $\ell_2$  assumes that the impact of noise is independent of the local characteristics of the image.

Typically,  $\ell_2$  works under the assumption of white Gaussian noise, which is not valid in general.

## LOSSES: $\ell_1$

It does not over-penalize larger errors, and, consequently, it may have different convergence properties wrt  $\ell_2$ .

$$\mathcal{L}^{\ell_1}(P) = \frac{1}{N} \sum_{p \in P} |x(p) - y(p)|$$

$$\partial \mathcal{L}^{\ell_1}(p) / \partial q = 0, \forall q \neq p.$$

$$\partial \mathcal{L}^{\ell_1}(P) / \partial x(p) = \text{sign}(x(p) - y(p))$$

## LOSSES: SSIM

The loss is the SSIM index in all the pixel of the patch

$$\mathcal{L}^{\text{SSIM}}(P) = \frac{1}{N} \sum_{p \in P} 1 - \text{SSIM}(p)$$

Where

$$\begin{aligned} \text{SSIM}(p) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \cdot \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \\ &= l(p) \cdot cs(p) \end{aligned}$$

The means and standard deviations are computed with a Gaussian filter with standard deviation  $\sigma_G$ .

Since the network is convolutional, it is possible to compute the patch loss as the SSIM at the central pixel of the patch  $\tilde{p}$

$$\mathcal{L}^{\text{SSIM}}(P) = 1 - \text{SSIM}(\tilde{p})$$

## LOSSES: SSIM

Limitations of SSIM used for training CNN

- Small  $\sigma_G$  the network loses the ability to preserve the local structure introduce artifacts in flat regions;
- Large  $\sigma_G$  we observe that the network tends to preserve noise in the proximity of edges.

For color images, SSIM-based losses are still an approximation since were designed for grayscale images



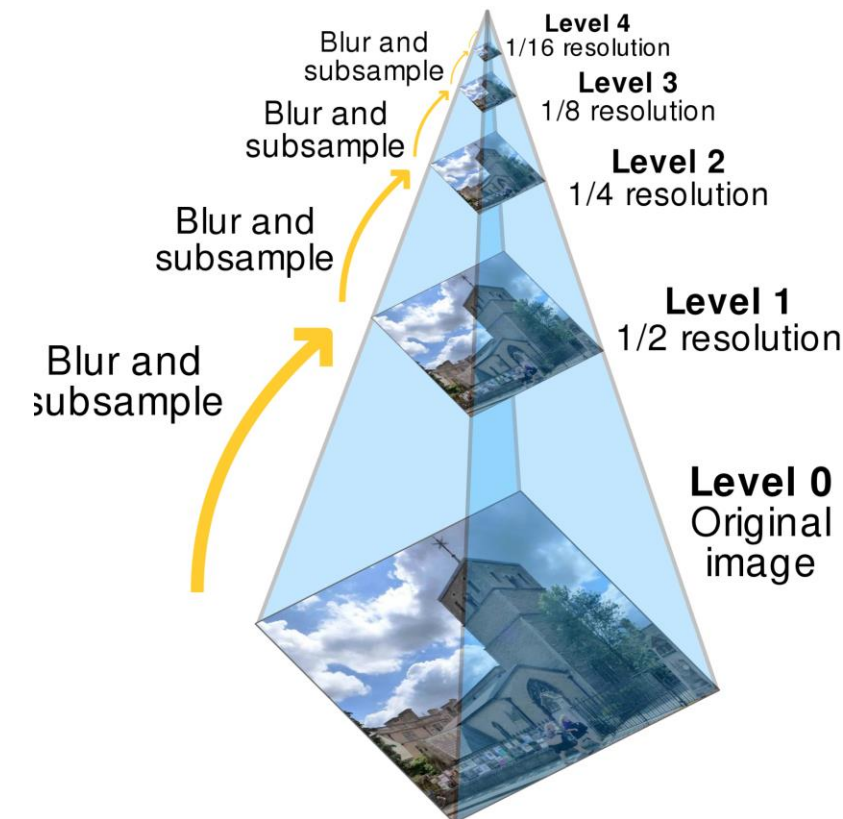
## LOSSES: MS-SSIM

Instead of fine tuning  $\sigma_G$  it is possible to adopt a multiscale version of the SSIM

$$\mathcal{L}^{\text{MS-SSIM}}(P) = 1 - \text{MS-SSIM}(\tilde{p})$$

The MS-SSIM should be computed over an image pyramid, which is computationally demanding

However, it can be well by the SSIM using different values of  $\sigma_G$



## LOSSES: MIXED $\ell^1$ AND MS-SSIM

By design, both MS-SSIM and SSIM are not particularly sensitive to uniform biases, resulting in changes of brightness or shifts of colors. MS-SSIM preserves the contrast in high-frequency regions better than  $\ell^2$  and  $\ell^1$ .

$\ell^1$  preserves colors and luminance—an error is weighed equally regardless of the local structure— but does not produce quite the same contrast as MS-SSIM.

To capture the best characteristics of both error functions

$$\mathcal{L}^{\text{Mix}} = \alpha \cdot \mathcal{L}^{\text{MS-SSIM}} + (1 - \alpha) \cdot G_{\sigma_G^M} \cdot \mathcal{L}^{\ell_1}$$

Where  $G_{\sigma_G^M}$  is a Gaussian kernel point-wise multiplied to make the  $\ell^1$  comparable with the MS-SSIM (which adopts Gaussian decay)

# LOSSES: EXPERIMENTS

The network performance improves with better loss functions, even when the network architecture is left unchanged.

Tested on different restoration tasks

Denoising + demosaicking			Training cost function					
Image quality metric	Noisy	<i>BM3D</i>	$\ell_2$	$\ell_1$	SSIM <sub>5</sub>	SSIM <sub>9</sub>	MS-SSIM	Mix
$1000 \cdot \ell_2$	1.65	0.45	0.56	0.43	0.58	0.61	0.55	<b>0.41</b>
PSNR	28.24	34.05	33.18	34.42	33.15	32.98	33.29	<b>34.61</b>
$1000 \cdot \ell_1$	27.36	14.14	15.90	13.47	15.90	16.33	15.99	<b>13.19</b>
SSIM	0.8075	0.9479	0.9346	0.9535	0.9500	0.9495	0.9536	<b>0.9564</b>
MS-SSIM	0.8965	0.9719	0.9636	0.9745	0.9721	0.9718	0.9741	<b>0.9757</b>
IW-SSIM	0.8673	0.9597	0.9473	0.9619	0.9587	0.9582	0.9617	<b>0.9636</b>
GMSD	0.1229	0.0441	0.0490	0.0434	0.0452	0.0467	0.0437	<b>0.0401</b>
FSIM	0.9439	0.9744	0.9716	0.9775	0.9764	0.9759	0.9782	<b>0.9795</b>
FSIM <sub>c</sub>	0.9381	0.9737	0.9706	0.9767	0.9752	0.9746	0.9769	<b>0.9788</b>

Super-resolution		Training cost function			
Image quality metric	Bilinear	$\ell_2$	$\ell_1$	MS-SSIM	Mix
$1000 \cdot \ell_2$	2.5697	1.2407	1.1062	1.3223	<b>1.0990</b>
PSNR	27.16	30.66	31.26	30.11	<b>31.34</b>
$1000 \cdot \ell_1$	28.7764	20.4730	19.0643	22.3968	<b>18.8983</b>
SSIM	0.8632	0.9274	0.9322	0.9290	<b>0.9334</b>
MS-SSIM	0.9603	0.9816	0.9826	0.9817	<b>0.9829</b>
IW-SSIM	0.9532	0.9868	0.9879	0.9866	<b>0.9881</b>
GMSD	0.0714	0.0298	0.0259	0.0316	<b>0.0255</b>
FSIM	0.9070	0.9600	0.9671	0.9601	<b>0.9680</b>
FSIM <sub>c</sub>	0.9064	0.9596	0.9667	0.9597	<b>0.9677</b>

JPEG de-blocking		Training cost function			
Image quality metric	Original JPEG	$\ell_2$	$\ell_1$	MS-SSIM	Mix
$1000 \cdot \ell_2$	0.6463	0.6511	0.6027	1.9262	<b>0.5580</b>
PSNR	32.60	32.73	32.96	27.66	<b>33.25</b>
$1000 \cdot \ell_1$	16.5129	16.2633	16.0687	33.6134	<b>15.5489</b>
SSIM	0.9410	0.9427	0.9467	0.9364	<b>0.9501</b>
MS-SSIM	0.9672	0.9692	0.9714	0.9674	<b>0.9734</b>
IW-SSIM	0.9527	0.9562	0.9591	0.9550	<b>0.9625</b>
GMSD	0.0467	0.0427	0.0413	0.0468	<b>0.0402</b>
FSIM	0.9805	0.9803	0.9825	0.9789	<b>0.9830</b>
FSIM <sub>c</sub>	0.9791	0.9790	0.9809	0.9705	<b>0.9815</b>

TABLE I: Average value of different image quality metrics on the testing dataset for the different cost functions. For SSIM, MS-SSIM, IW-SSIM, GMSD and FSIM the value reported here has been obtained as an average of the three color channels. Best results are shown in bold.(Lower is better for  $\ell_1$ ,  $\ell_2$ , and GMSD, higher is better for the others.)

## LOSSES: $\ell_1$ VS $\ell_2$

Smoothness and convexity of  $\ell_2$  could lead to a local minima.

Therefore: change the network loss after convergence and run a few more iterations

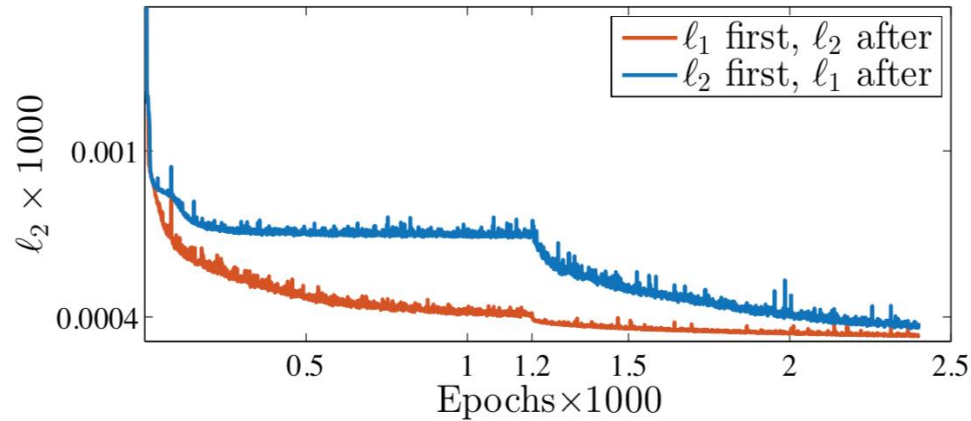


Fig. 7:  $\ell_2$  loss on the testing set for the two networks that switch loss functions during training.

Image quality metric	Training cost function	
	$\ell_2$ first, $\ell_1$ after	$\ell_1$ first, $\ell_2$ after
$1000 \cdot \ell_2$	0.3939	<b>0.3896</b>
PSNR	34.76	<b>34.77</b>
$1000 \cdot \ell_1$	<b>12.7932</b>	12.8919
SSIM	0.9544	0.9540
MS-SSIM	0.9753	0.9748
IW-SSIM	0.9634	0.9624
GMSD	0.0432	0.0405
FSIM	0.9777	0.9781
FSIM <sub>c</sub>	0.9770	0.9774

TABLE II: Average value of different image quality metrics for the networks trained for denoising + demosaicking with alternating loss functions. Bold indicates that the network achieves a better score than any of the networks in Table I, see Section V-A.

They still do not outperform the MIX loss on any perceptual metrics

# **HANDLING NOISE MODEL**

# FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising

Kai Zhang, Wangmeng Zuo<sup>ID</sup>, *Senior Member, IEEE*, and Lei Zhang<sup>ID</sup>, *Fellow, IEEE*

# FFDNET

FFD-Net learns a mapping function

$$z \rightarrow \mathcal{F}(\boldsymbol{\theta}, z, M)$$

the network parameters  $\boldsymbol{\theta}$  does not depend on noise parameters (e.g. the standard deviation),

**Network inputs for an image  $W \times H \times C$**

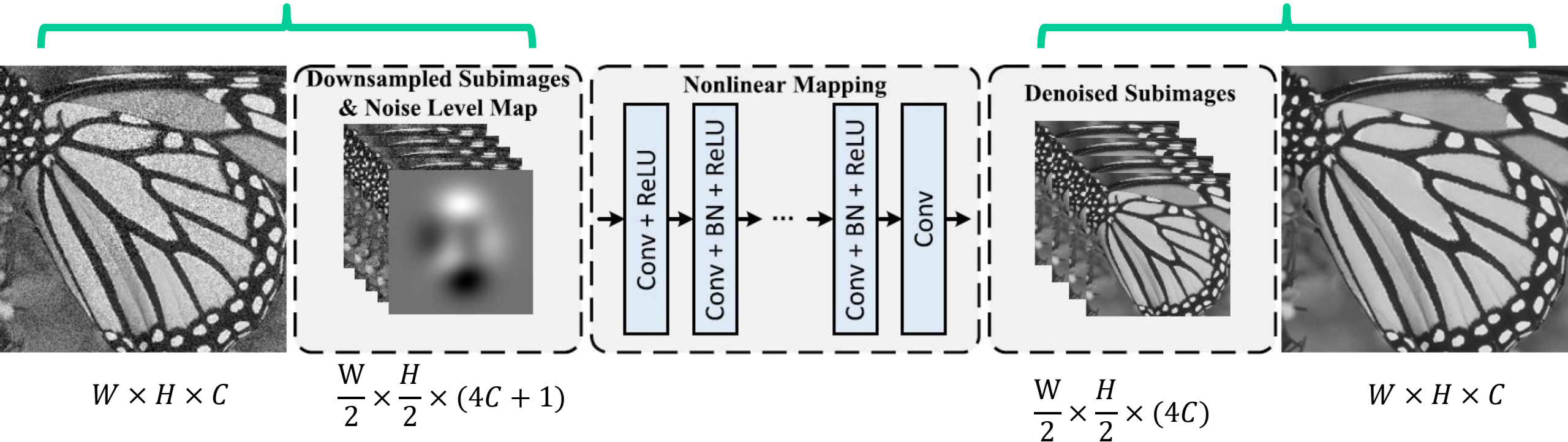
- a noise level-map  $M$  of the noise standard deviation over  $z$
- 4 downsampled and cropped images having size  $\frac{W}{2} \times \frac{H}{2} \times C$ , stacked in different channels

Thus the input has size  $\frac{W}{2} \times \frac{H}{2} \times (4C + 1)$

# FFDNET ARCHITECTURE

Input preparation,  
image downsampling

upsampling





# FFDNET ARCHITECTURE

All the convolutions are 3x3, as DnCNN

## Grayscale images:

- depth level:  $d = 15$ ,
- Number of filters per layer: 64

## Color images:

- Depth level  $d = 12$ ,
- Number of filters per layer: 96

Design driven by performance and «heuristic criteria»: use lower values of  $d$  in color images to better exploit correlation among R,G,B channels

## FFDNET: THE INPUT

Resized images (pixel shuffle):

- Resize is meant to reduce spatial extent of the input, thus the computational complexity in all the layers after the first one.
- Resizing also increases receptive field.
- “Downsampling” is indeed shuffling of the image among channels

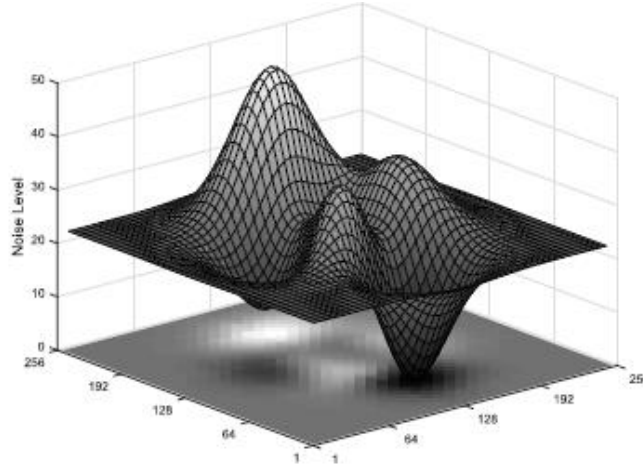
Noise level-map  $M$ :

- An image reporting in each pixel the noise standard deviation
- It enables using a single model for different values of  $\sigma$  without need to retrain
- In case of Additive White Gaussian Noise (AWGN),  $M$  corresponds to a uniform image equal to the noise std  $\sigma$
- In case of spatial-variant noise,  $M$  is a non-uniform image defined by the noisy image  $z$  and the sensor characteristics

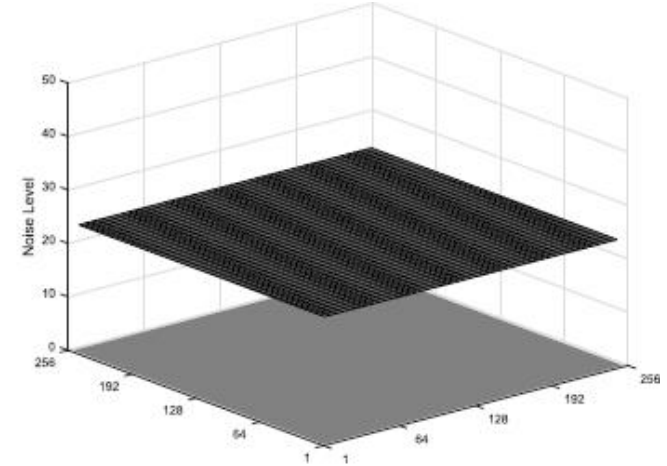
## FFDNET: NOISE LEVEL MAPS



(a)



(b)



(c)

K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," IEEE Transactions on Image Processing, vol. 26, no. 7, pp. 3142–3155, July 2017.

# FFDNET TRAINING

Loss to minimize

$$\ell(\Theta) = \frac{1}{2N} \sum_{i=1}^N \|\mathcal{F}(z_i, \Theta, M) - y_i\|_F^2$$

Assessing on how good the network can estimate noise-free patches

## Patch-wise training

- Training samples are easy to generate: add synthetic noise,
- Loss measured on patches cropped from noisy images
- No need to train using heterogeneous noise maps, the fully convolutional network can operate on locally uniform noise maps
- Patch size 50x50 for grayscale, 70x70 in color (each patch have to include the receptive field)

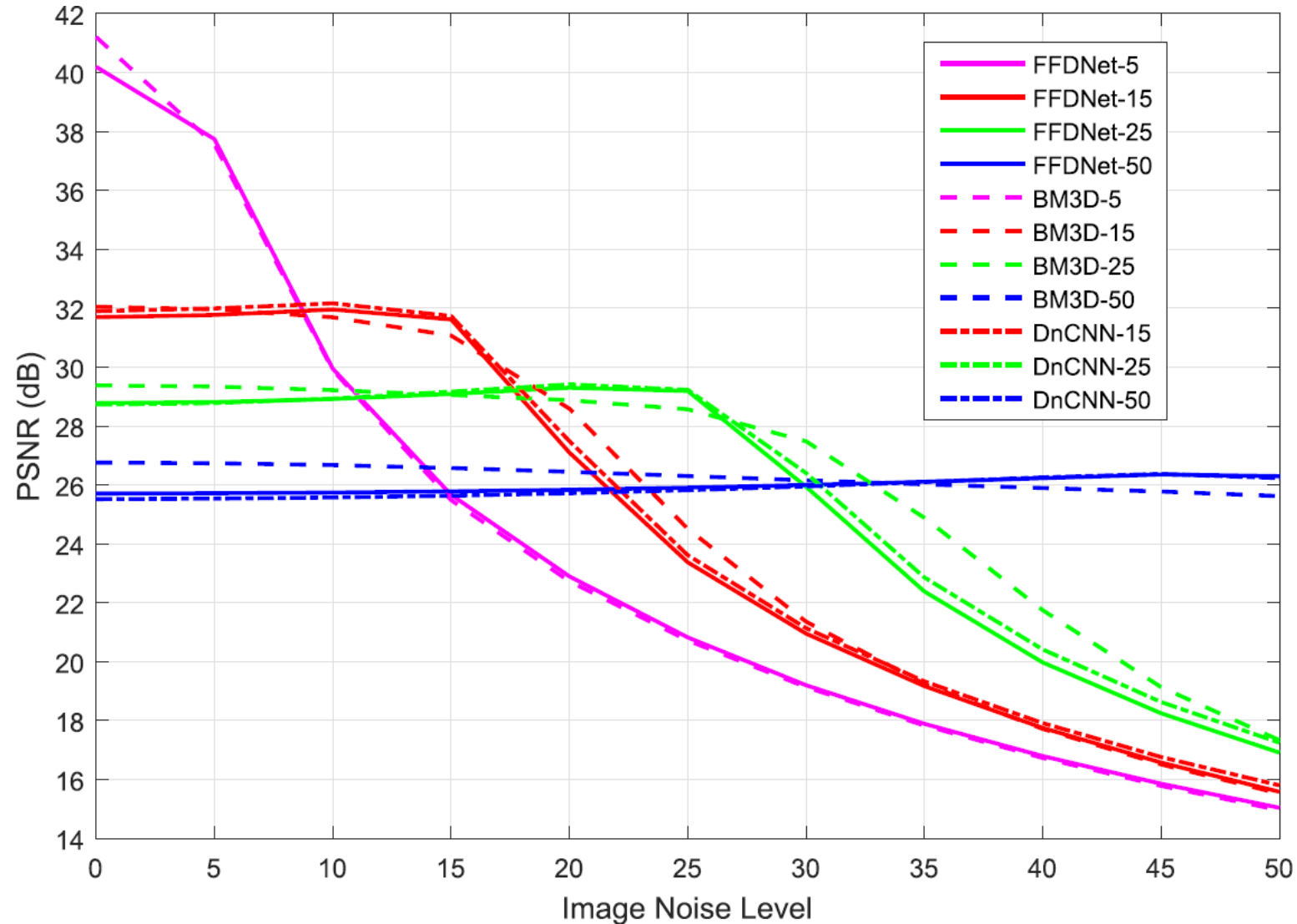
## DNCNN AND FFDNET PERFORMANCE

TABLE III

THE AVERAGE PSNR(dB) RESULTS OF DIFFERENT METHODS  
ON BSD68 WITH NOISE LEVELS 15, 25 35, 50 AND 75

Methods	BM3D	WNNM	MLP	TNRD	DnCNN	FFDNet
$\sigma = 15$	31.07	31.37	–	31.42	31.72	31.63
$\sigma = 25$	28.57	28.83	28.96	28.92	29.23	29.19
$\sigma = 35$	27.08	27.30	27.50	–	27.69	27.73
$\sigma = 50$	25.62	25.87	26.03	25.97	26.23	26.29
$\sigma = 75$	24.21	24.40	24.59	–	24.64	24.79

## DNCNN AND FFDNET PERFORMANCE (WRONG $M$ )



Overestimating  $\sigma$  results in oversmoothing, thus the PSNR is constant but does not reach the same level as when the noise level is correct. This holds for all the denoising algorithms.

Underestimating  $\sigma$  results in noisy outputs, thus decreases the PSNR.



FDD-Net



C-BM3D



original





# ROBUST AND INTERPRETABLE BLIND IMAGE DENOISING VIA BIAS-FREE CONVOLUTIONAL NEURAL NETWORKS

**Sreyas Mohan\***

Center for Data Science  
New York University  
sm7582@nyu.edu

**Zahra Kadkhodaie\***

Center for Data Science  
New York University  
zk388@nyu.edu

**Eero P. Simoncelli**

Center for Neural Science, and  
Howard Hughes Medical Institute  
New York University  
eero.simoncelli@nyu.edu

**Carlos Fernandez-Granda**

Center for Data Science, and  
Courant Inst. of Mathematical Sciences  
New York University  
cfgranda@cims.nyu.edu



## GENERALIZATION TO DIFFERENT NOISE LEVELS

An important advantage of deep-learning techniques over traditional methodology is that **a single neural network can be trained to perform denoising at a wide range of noise levels.**

*Empirical evidence that current state-of-the-art architectures systematically overfit to the noise levels in the training set, performing very poorly at new noise levels.*

*Generalization can be achieved through a simple architectural modification: removing all additive constants.*

*"bias-free" networks attain state-of-the-art performance over a broad range of noise levels, even when trained over a narrow range.*

# BS-CNN: REMOVING BIASES

The shaded region in the plots denote the training range for the two networks (standard DnCNN in red and the equivalent Bias-Free CNN in blue)

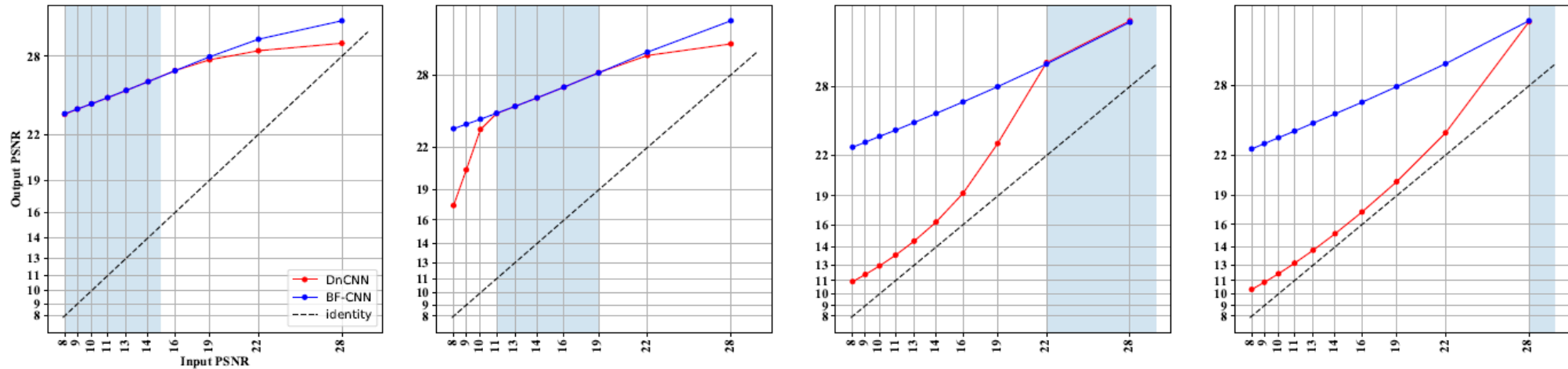


Figure 3: Comparison of the performance of a CNN and a BF-CNN with the same architecture for the experimental design described in Section 5. The performance is quantified by the PSNR of the denoised image as a function of the input PSNR. Both networks are trained over a fixed ranges of noise levels indicated by a blue background. In all cases, the performance of BF-CNN generalizes robustly beyond the training range, while that of the CNN degrades significantly. The CNN used for this example is DnCNN (Zhang et al., 2017); using alternative architectures yields similar results (see Figures 11 and 12).

# BS-CNN: REMOVING BIASES

Results are consistent with other CNN architectures

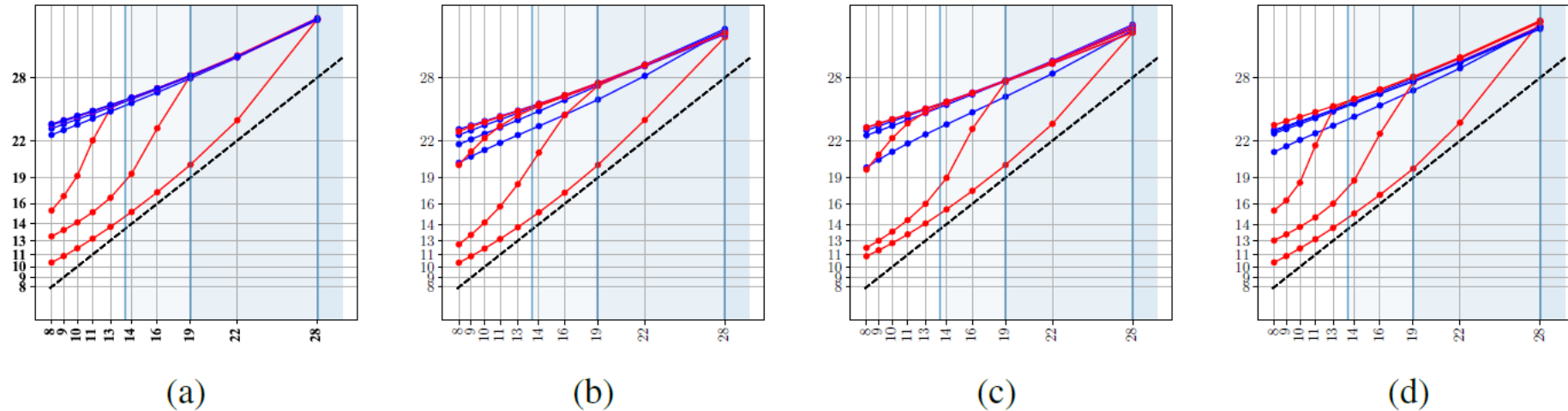


Figure 11: Comparisons of architectures with (red curves) and without (blue curves) a net bias for the experimental design described in Section 5. The performance is quantified by the PSNR of the denoised image as a function of the input PSNR of the noisy image. All the architectures with bias perform poorly out of their training range, whereas the bias-free versions all achieve excellent generalization across noise levels. **(a)** Deep Convolutional Neural Network, DnCNN (Zhang et al., 2017). **(b)** Recurrent architecture inspired by DURR (Zhang et al., 2018a). **(c)** Multiscale architecture inspired by the UNet (Ronneberger et al., 2015). **(d)** Architecture with multiple skip connections inspired by the DenseNet (Huang et al., 2017).

# BS-CNN: REMOVING BIASES

Results are consistent with other CNN architectures

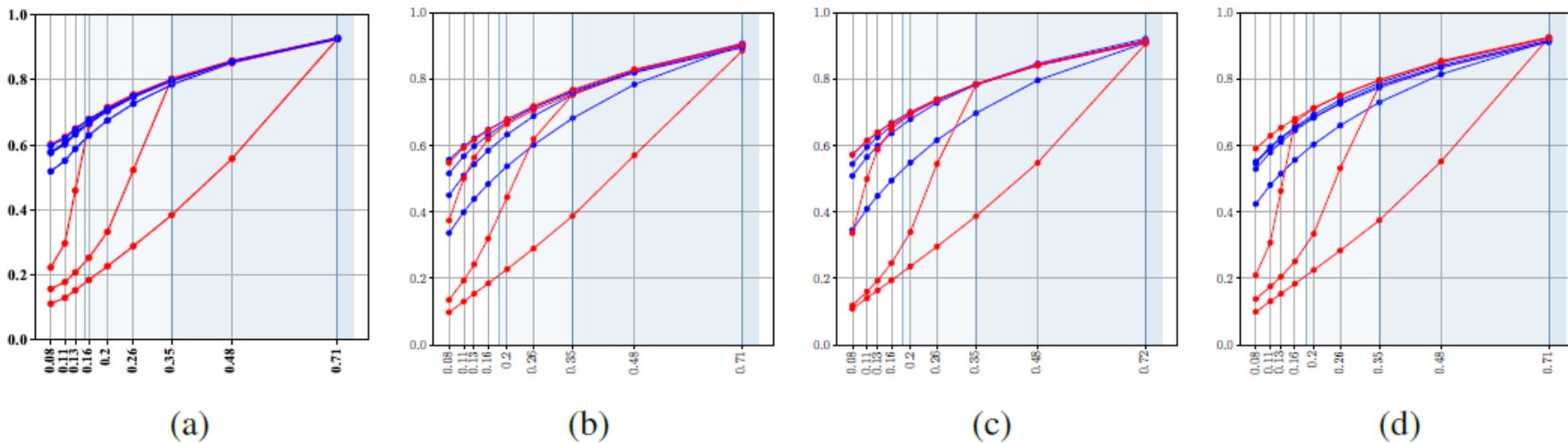


Figure 12: Comparisons of architectures with (red curves) and without (blue curves) a net bias for the experimental design described in Section 5. The performance is quantified by the SSIM of the denoised image as a function of the input SSIM of the noisy image. All the architectures with bias perform poorly out of their training range, whereas the bias-free versions all achieve excellent generalization across noise levels. (a) Deep Convolutional Neural Network, DnCNN (Zhang et al., 2017). (b) Recurrent architecture inspired by DURR (Zhang et al., 2018a). (c) Multiscale architecture inspired by the UNet (Ronneberger et al., 2015). (d) Architecture with multiple skip connections inspired by the DenseNet (Huang et al., 2017).

# CBDNET

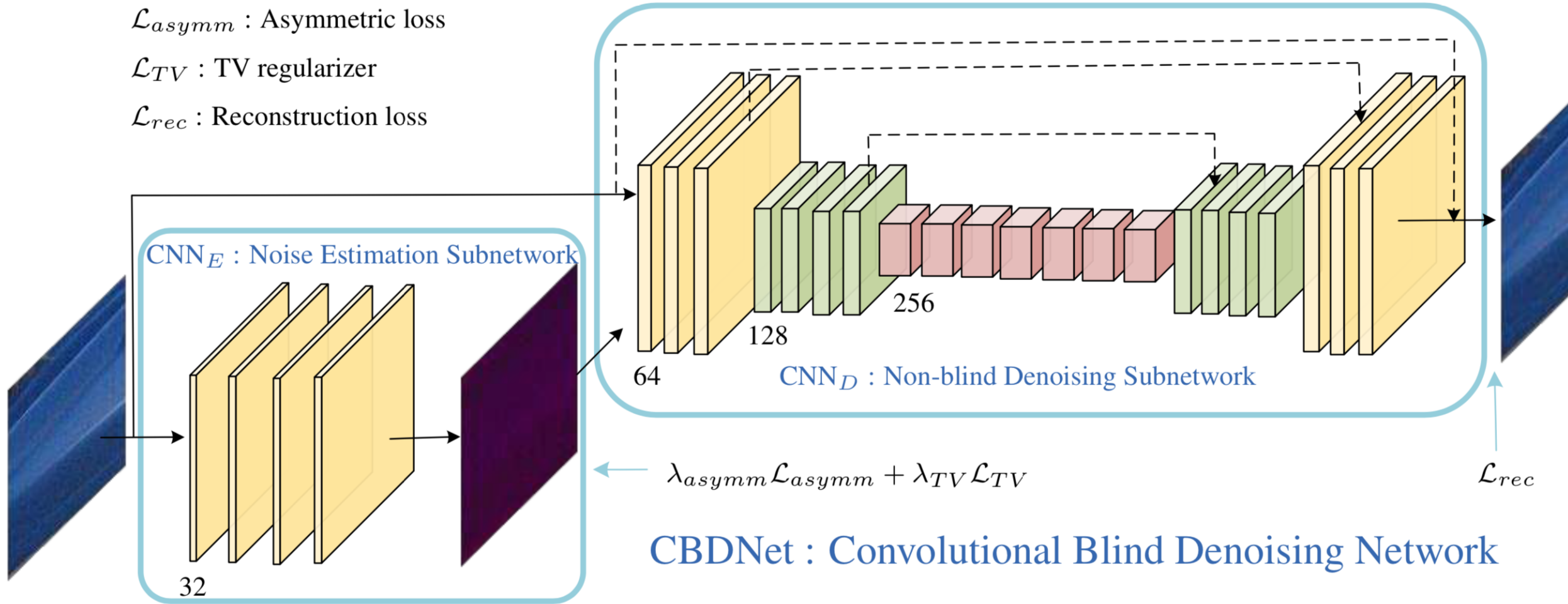
- Existing CNN denoisers tend to be over-fitted to Gaussian noise and **generalize poorly to real-world noisy images with more sophisticated noise.**
- Realistic noise model is the foremost issue for blind denoising of real photographs.
- **Take into account both Poisson-Gaussian model and in-camera processing pipeline** (e.g. de-mosaicing, Gamma correction, and JPEG compression).
- CBDNet is comprised of two subnetworks:
  - noise estimation and
  - non-blind denoising.
- Adopt an asymmetric loss by imposing more penalty on under-estimation error of noise level.
- The network has a bottleneck

# CBDNET: NETWORK

$\mathcal{L}_{asymm}$  : Asymmetric loss

$\mathcal{L}_{TV}$  : TV regularizer

$\mathcal{L}_{rec}$  : Reconstruction loss



# **CLASS-AWARE DENOISING**

# Class-Aware Fully Convolutional Gaussian and Poisson Denoising

Tal Remez<sup></sup>, *Member, IEEE*, Or Litany, *Member, IEEE*, Raja Giryes<sup></sup>, *Member, IEEE*,  
and Alex M. Bronstein, *Fellow, IEEE*



## KEY INGREDIENTS

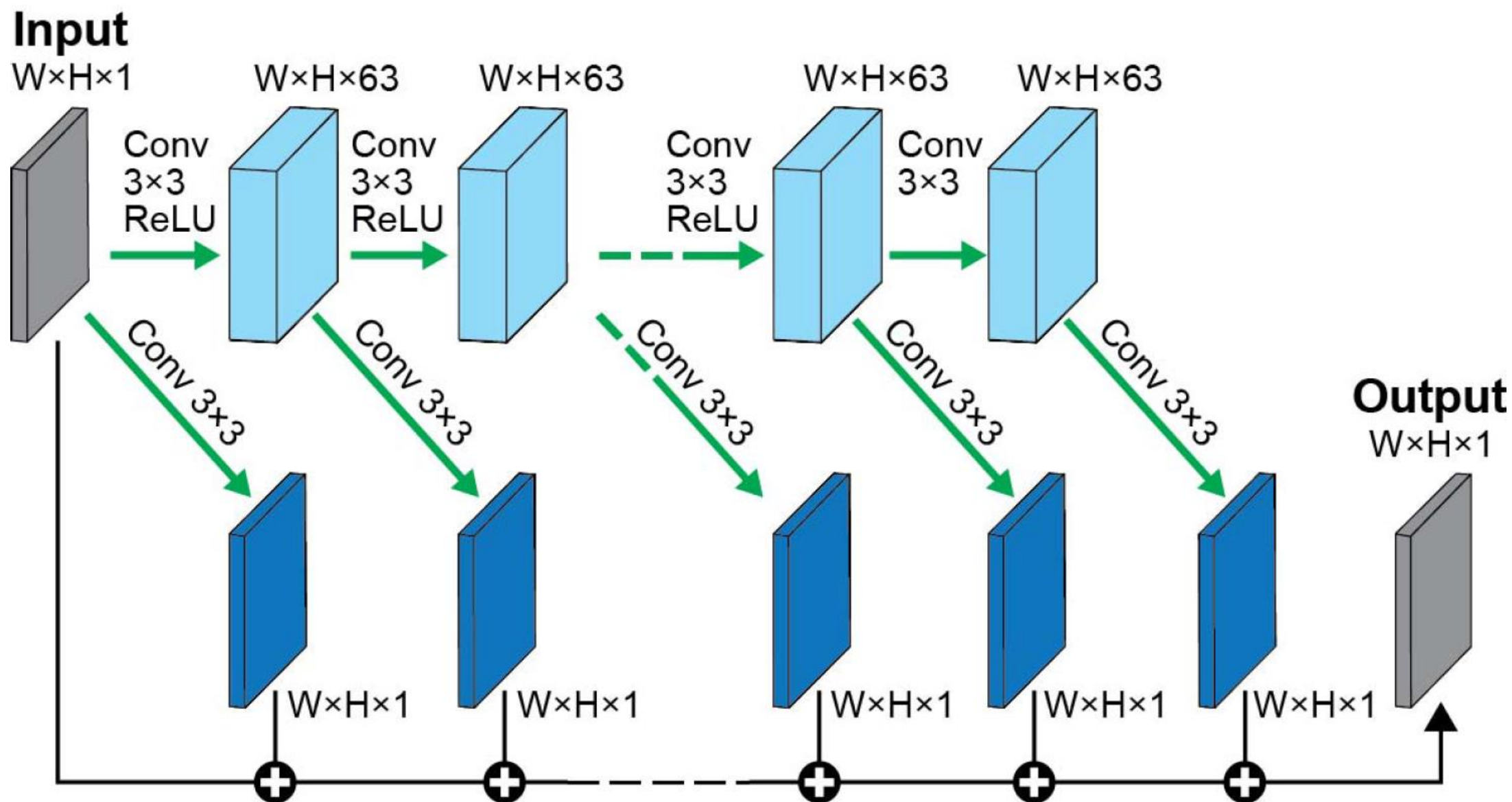
**Fully convolutional** to enable denoising of images having different size.

- Fast execution and relatively low number of parameters

**Gradual denoising** as the volume circulates through the network

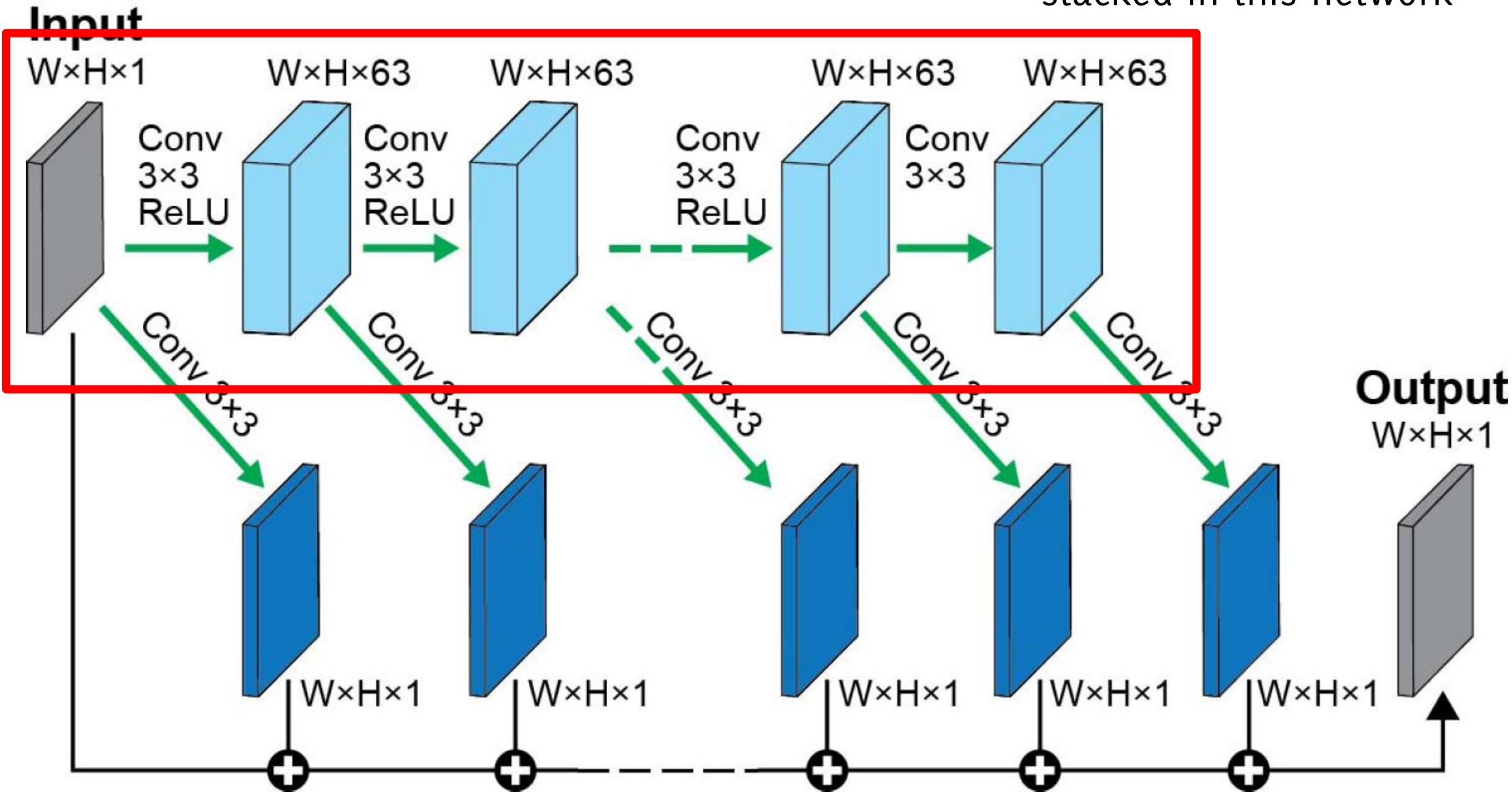
**Improve prior on images:** narrow down the space of images and **train a network for a specific class of images.**

## THE NETWORK ARCHITECTURE



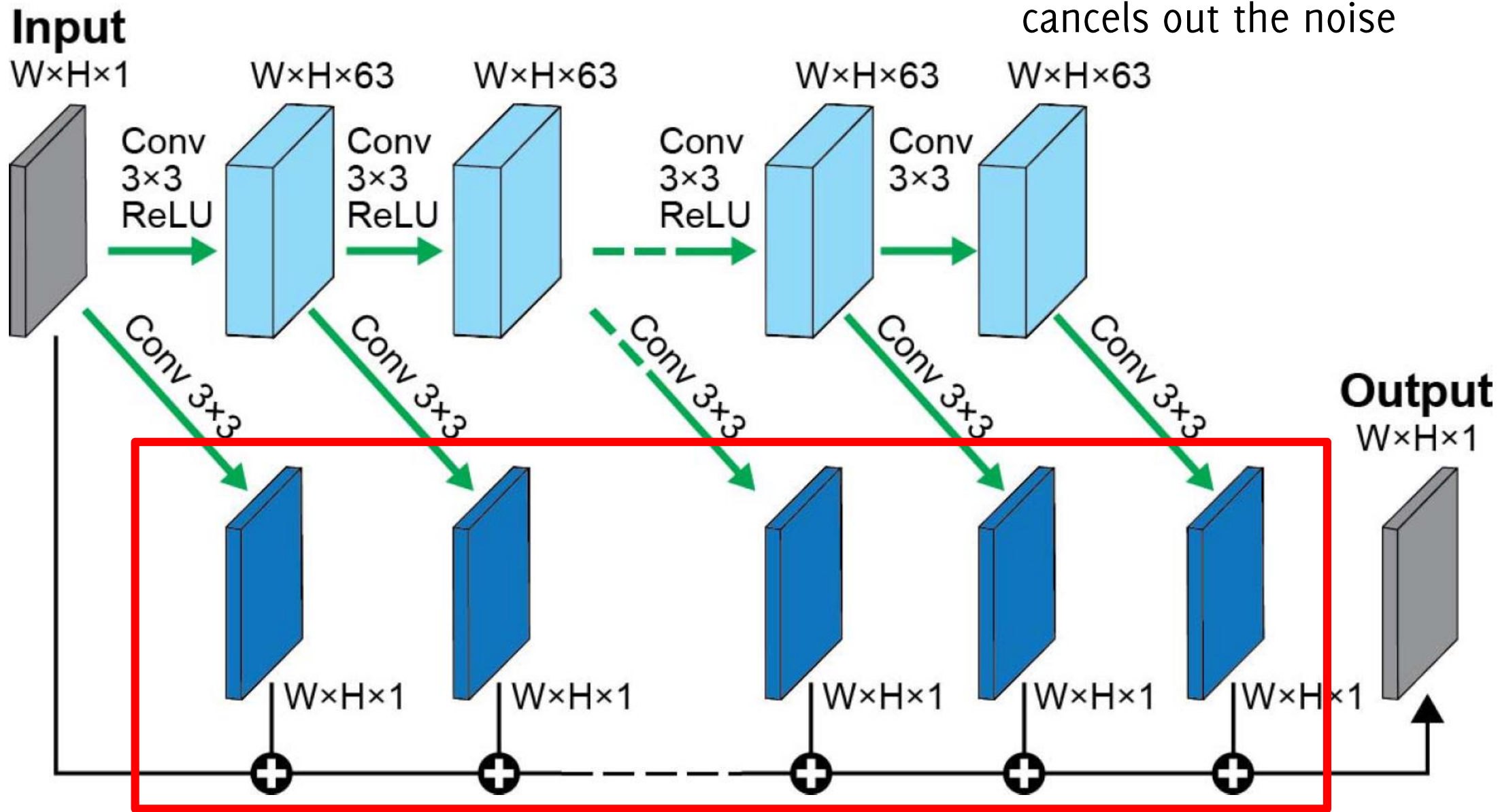
THE NETWORK ARCHITECTURE

20 of such layers are being stacked in this network



# THE NETWORK ARCHITECTURE

These are referred to as noise estimates, because their sum cancels out the noise





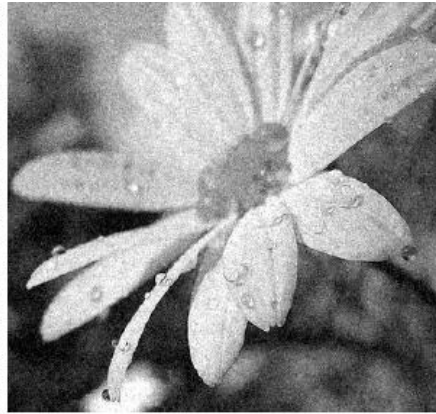
# NOISE ESTIMATES

It is possible to analyze the process by visualizing noise estimates

Gradual Denoising



Noisy input 20.16 dB



24.52 dB



29.81 dB



31.08 dB

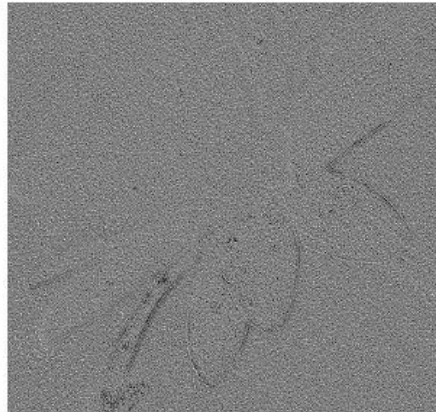


Output 32.84 dB

Noise Estimates



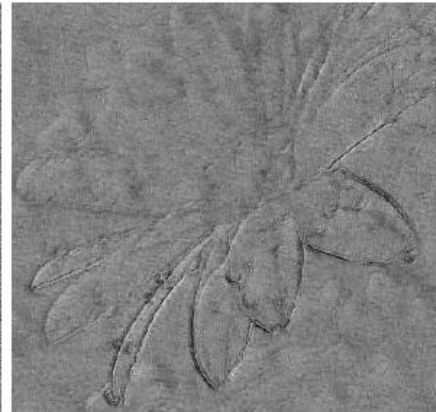
Ground truth



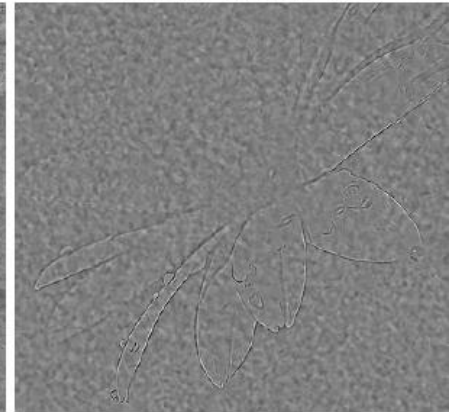
Layer 5



Layer 10



Layer 15

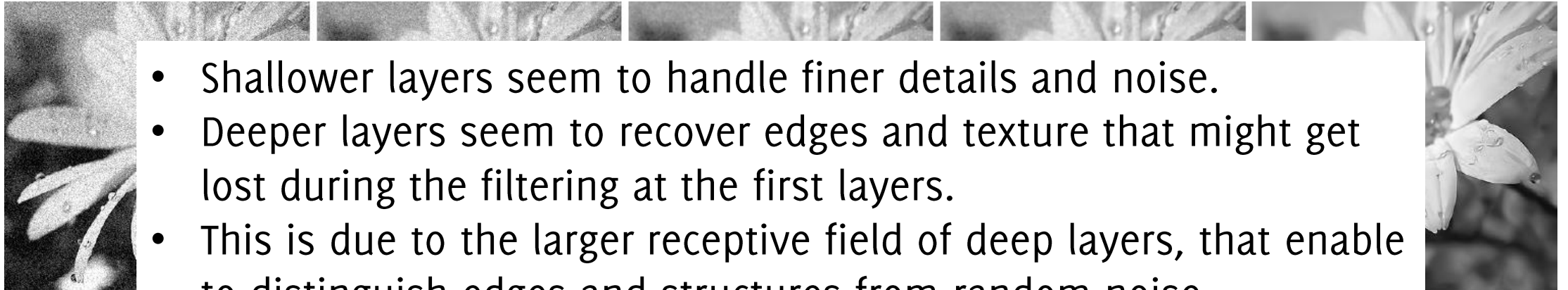


Layer 20

# NOISE ESTIMATES

It is possible to analyze the process by visualizing noise estimates

Gradual Denoising



- Shallower layers seem to handle finer details and noise.
- Deeper layers seem to recover edges and texture that might get lost during the filtering at the first layers.
- This is due to the larger receptive field of deep layers, that enable to distinguish edges and structures from random noise

.84 dB

Noise Estimates



Ground truth

Layer 5

Layer 10

Layer 15

Layer 20

# CLASS-AWARE DENOISING

## Training

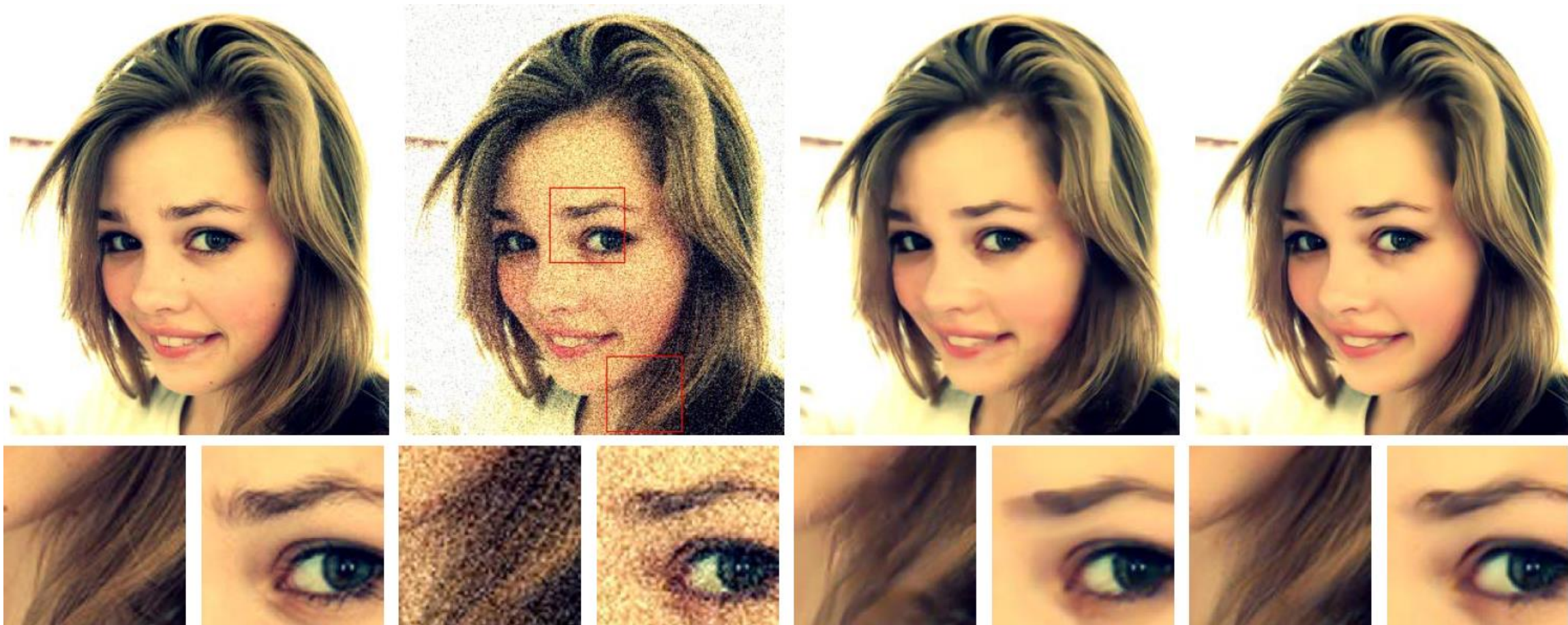
- Train different networks for a specific class of images. Use the same FCNN architecture. These are the classes: *face*, *pet*, *flower*, *living room*, *street*.

## Testing

- Estimate the class from a classification CNN, thus the corresponding network



## CLASS-AWARE VS CLASS-AGNOSTIC DENOISING



Ground truth image

Noisy image

Class-agnostic denoiser [1]  
30.50 dB

Our class-aware method  
30.88 dB



# NONLOCALITY

# Understanding the Effective Receptive Field in Deep Convolutional Neural Networks

---

Wenjie Luo\*    Yujia Li\*    Raquel Urtasun    Richard Zemel

Department of Computer Science

University of Toronto

{wenjie, yujiali, urtasun, zemel}@cs.toronto.edu

## THE EFFECTIVE RECEPTIVE FIELD

*Not all pixels in a receptive field contribute equally to an output unit's response. Intuitively it is easy to see that pixels at the center of a receptive field have a much larger impact on an output.*

*In the forward pass, central pixels can propagate information to the output through many different paths, while the pixels in the outer area of the receptive field have very few paths to propagate its impact.*

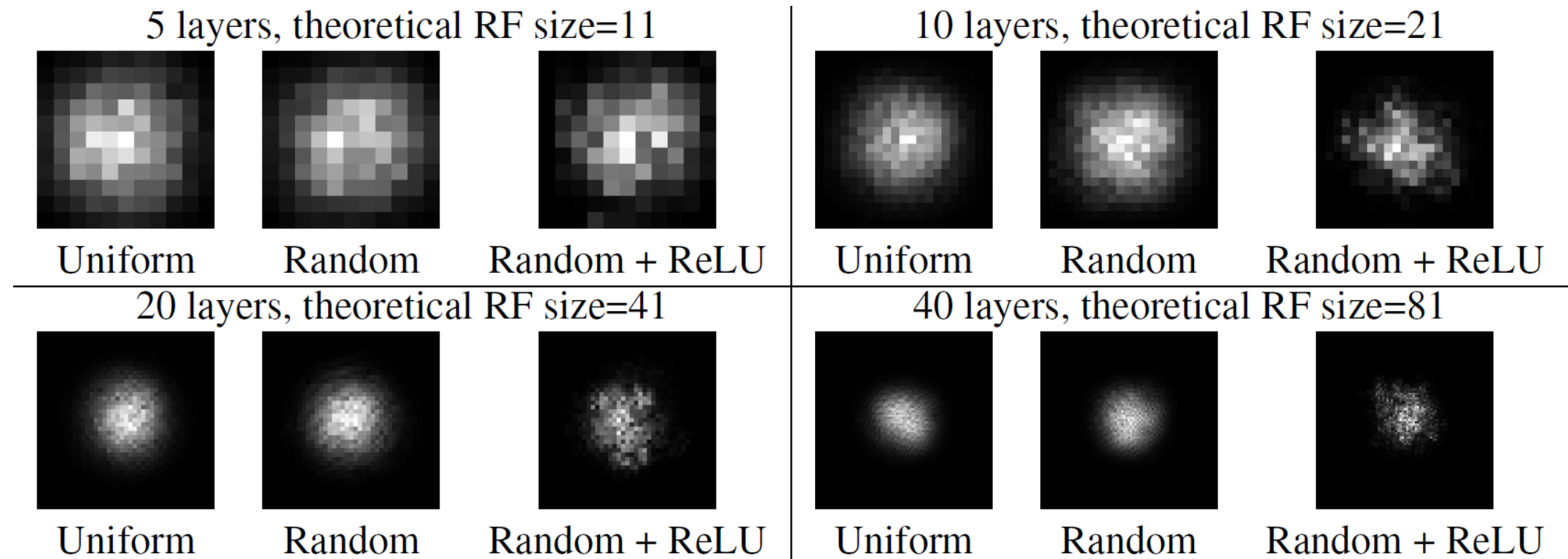
*In the backward pass, gradients from an output unit are propagated across all the paths, and therefore the central pixels have a much larger magnitude for the gradient from that output.*

In many cases, the impact in a receptive field distributes as a Gaussian: **the effective receptive field**, only occupies a fraction of the **theoretical receptive field**.

## THE EFFECTIVE RECEPTIVE FIELD

Place a gradient signal of 1 at the center of the output plane and 0 everywhere else, and then back-propagate this gradient through the network to get input gradients.

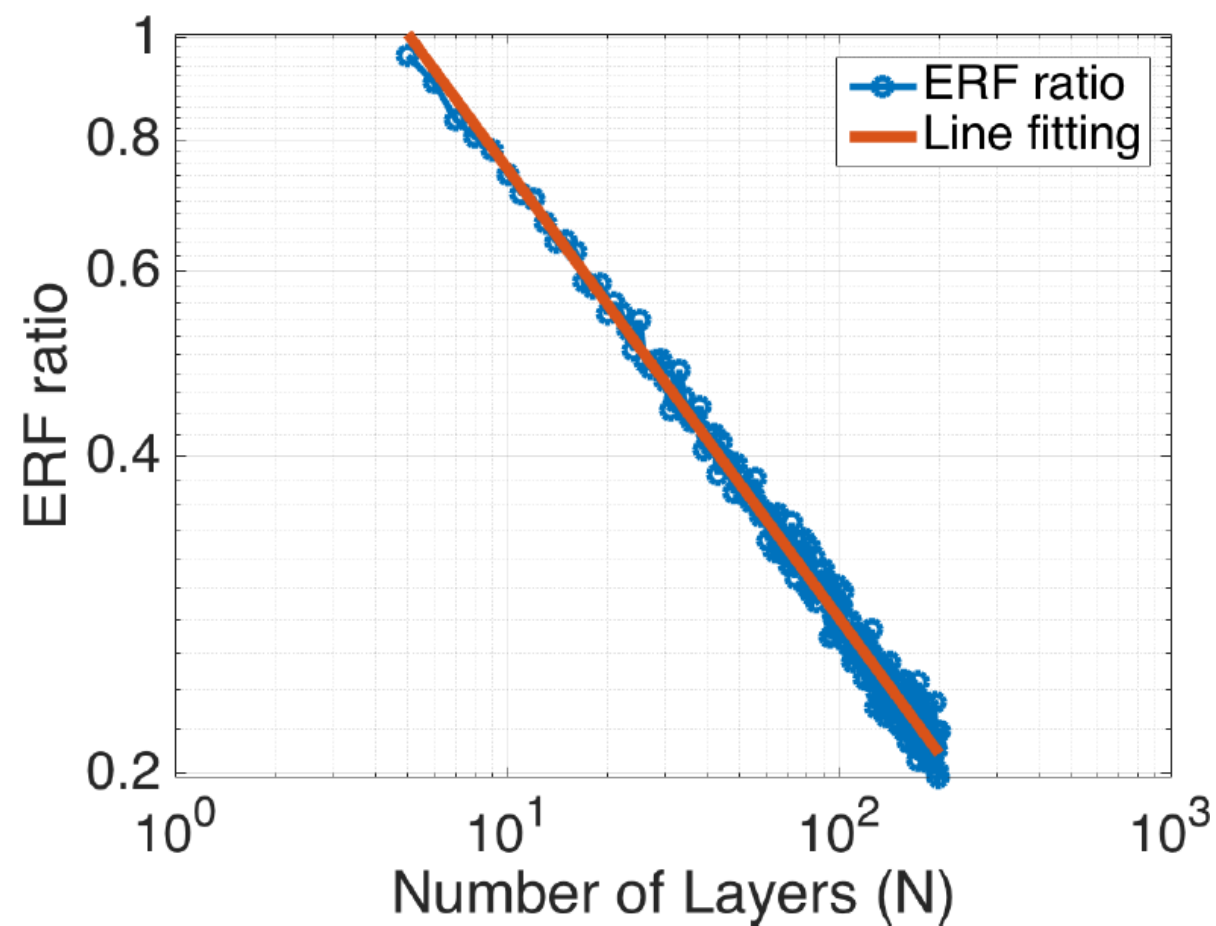
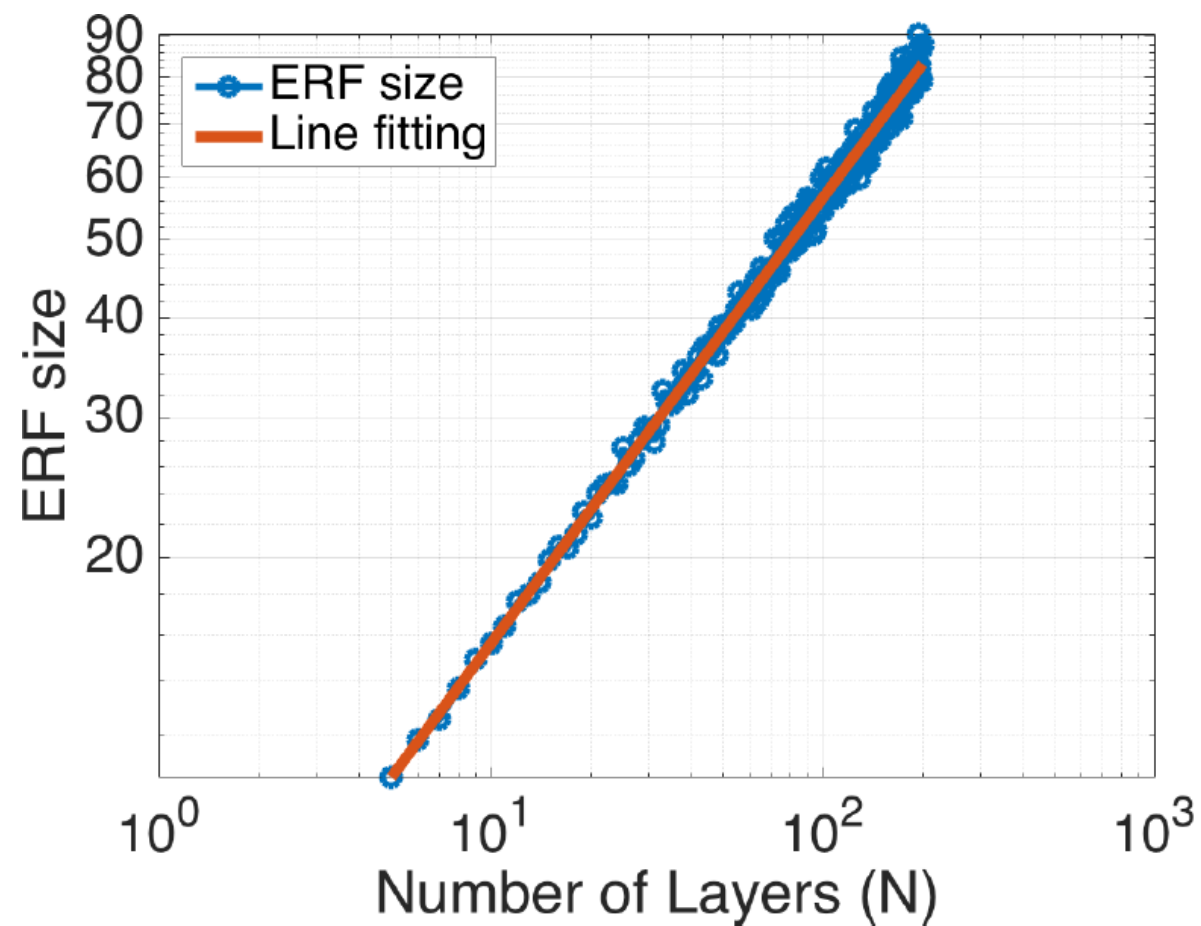
Average the resulting gradients at the input over multiple initialization of the network



The deeper the network, the smaller the ERF w.r.t. the theoretical one!

# THE EFFECTIVE RECEPTIVE FIELD

How the network depth influences the ERF and the theoretical one

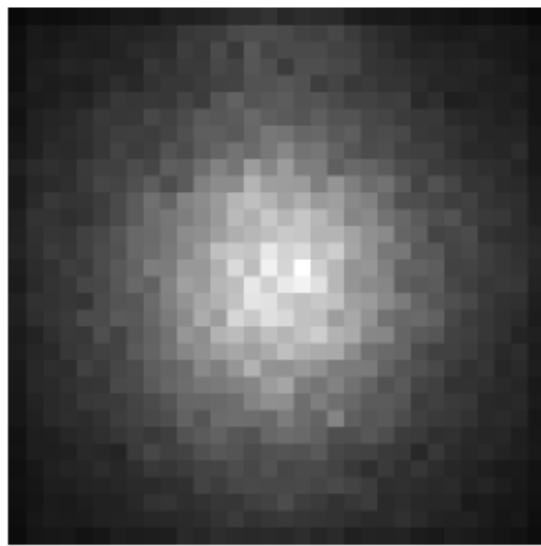
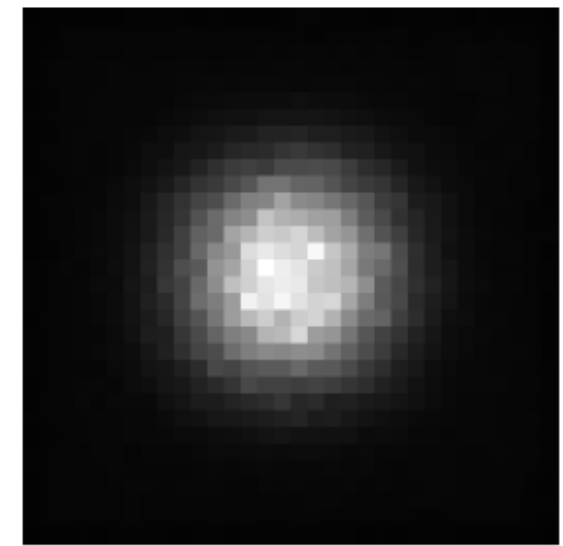


# THE EFFECTIVE RECEPTIVE FIELD

Network training increases the ERF. Is random initialization bad?

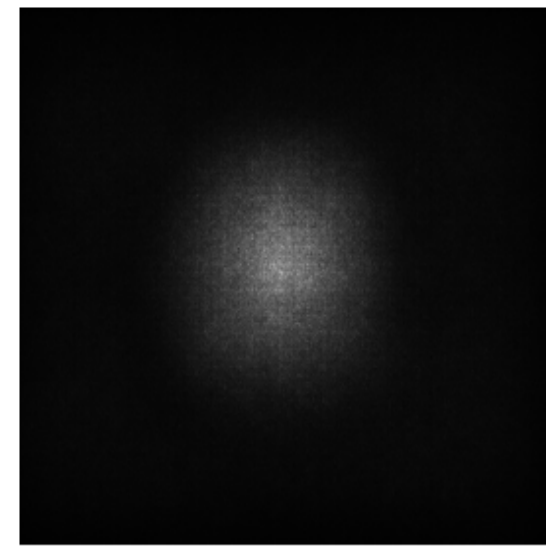
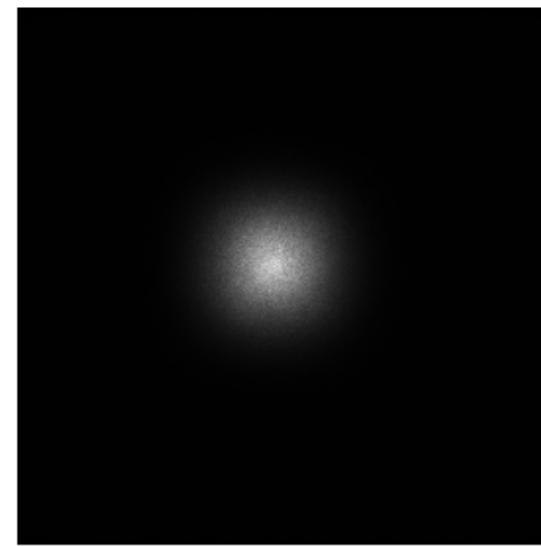
CIFAR 10

CamVid



Before Training

After Training



Before Training

After Training


## THE EFFECTIVE RECEPTIVE FIELD

*A new **random weight initialization scheme** that makes the **weights at the center** of the convolution kernel to have **a smaller scale**, and the **weights on the outside** to be **larger** this diffuses the concentration on the center out to the periphery.*

*Practically, we can initialize the network **with any initialization method**, then **scale the weights according** to a distribution that has a **lower scale at the center** and **higher scale on the outside**.*

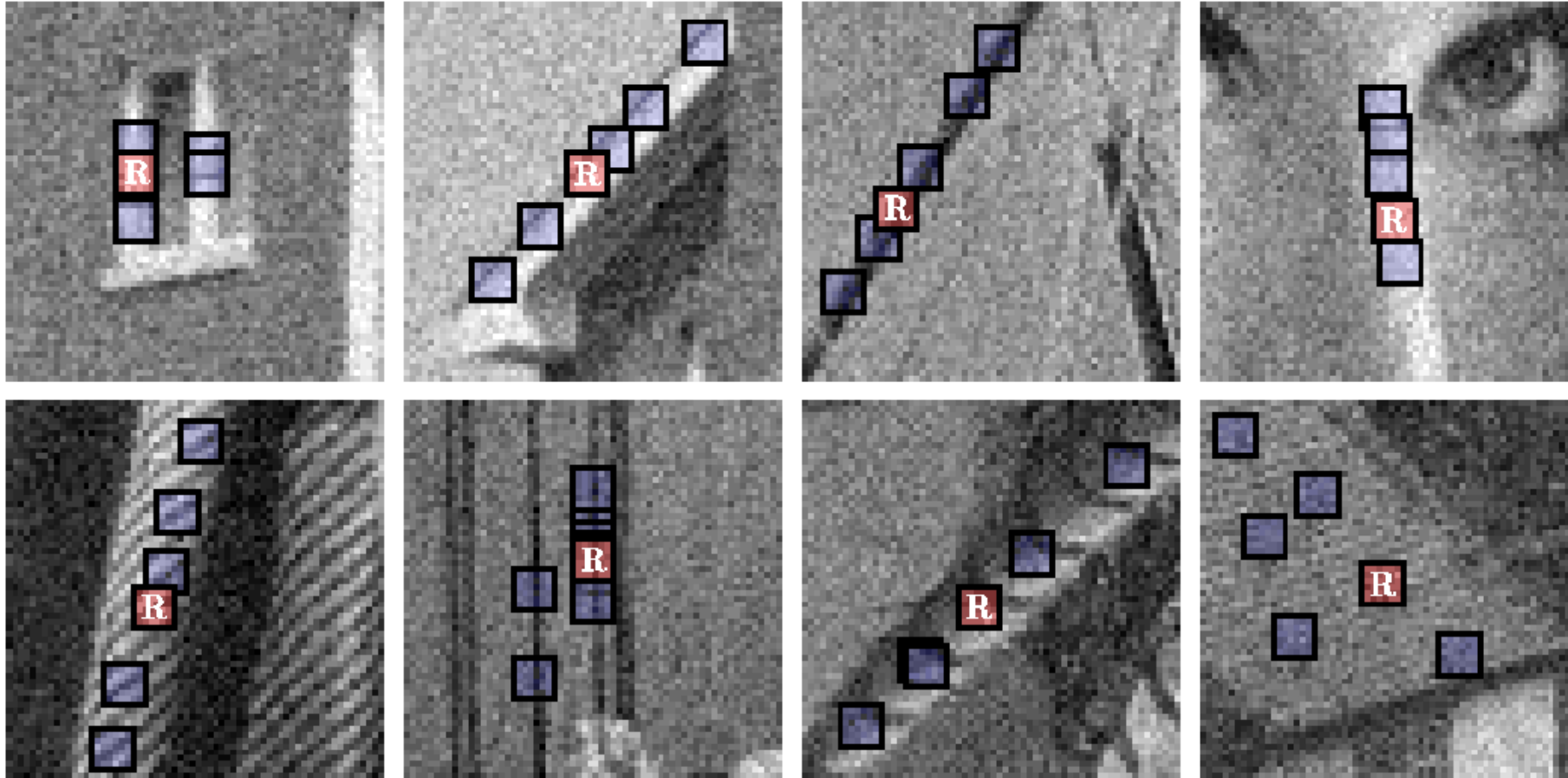
*[...] In a few cases we get a 30% speed-up of training compared to the more standard initializations. But overall the benefit of this method is not always significant.*

# Nonlocality-Reinforced Convolutional Neural Networks for Image Denoising

Cristóvão Cruz , Alessandro Foi , *Senior Member, IEEE*, Vladimir Katkovnik ,  
and Karen Egiazarian, *Fellow, IEEE*



# NONLOCAL SELF SIMILARITY



*In a natural image, for any given patch there exist **many** other **similar** looking patches at **different spatial locations**.*

Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 16(8).

## NONLOCAL FILTERS (NLF)

In image/signal processing, the NonLocal Filters are designed to exploit local image regularities and redundancies, and to jointly process similar patches for extracting signal information.

**NonLocality:** relative distance inside the receptive field is not taken into account to assess pixel-wise similarity in images.

NLF rely on rather large (effective) receptive fields

TABLE I

THE EFFECTIVE PATCH SIZES OF DIFFERENT METHODS WITH NOISE LEVEL  $\sigma = 25$ .

Methods	BM3D [2]	WNNM [13]	EPLL [33]	MLP [24]	CSF [14]	TNRD [16]
Effective Patch Size	$49 \times 49$	$361 \times 361$	$36 \times 36$	$47 \times 47$	$61 \times 61$	$61 \times 61$

# NN<sub>3</sub>D

A **simple iterative framework** that combines the advantages of

- **CNN** as powerful models to describe the image structures
- **NLF** (NonLocal Filters) that exploit image redundancies

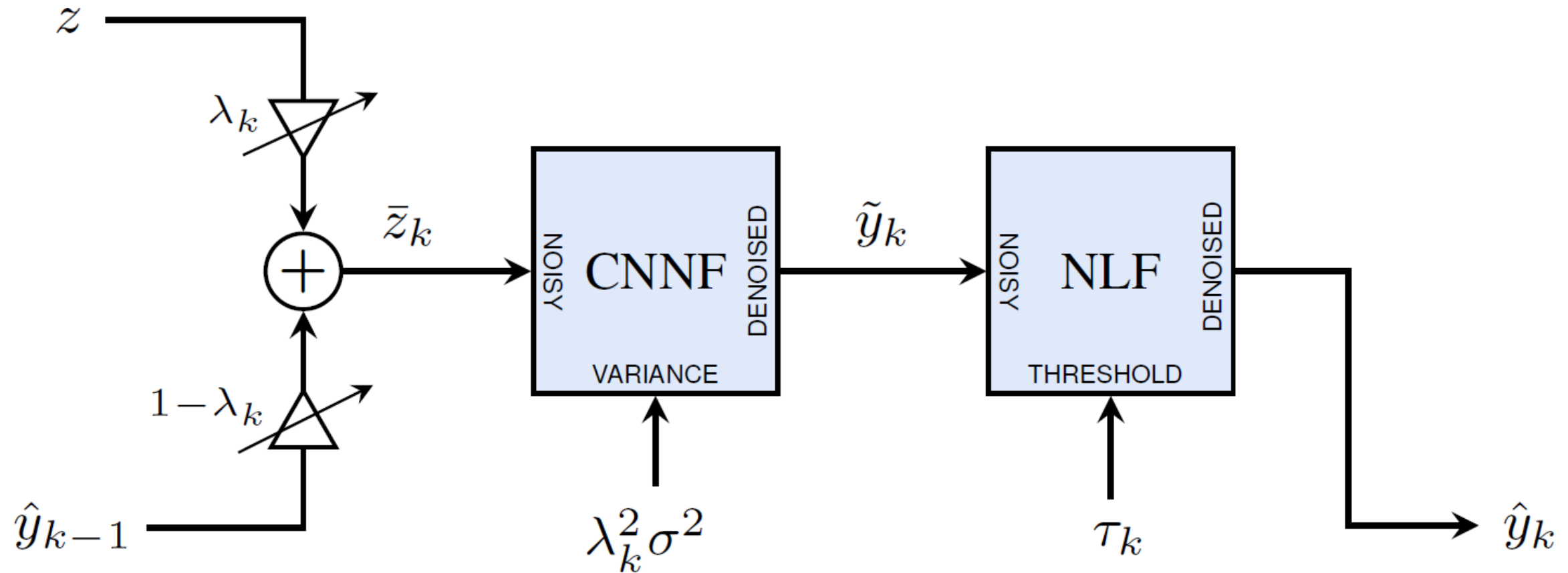
In a **plug-in approach** that uses any **generic CNN and NLF**

## The idea:

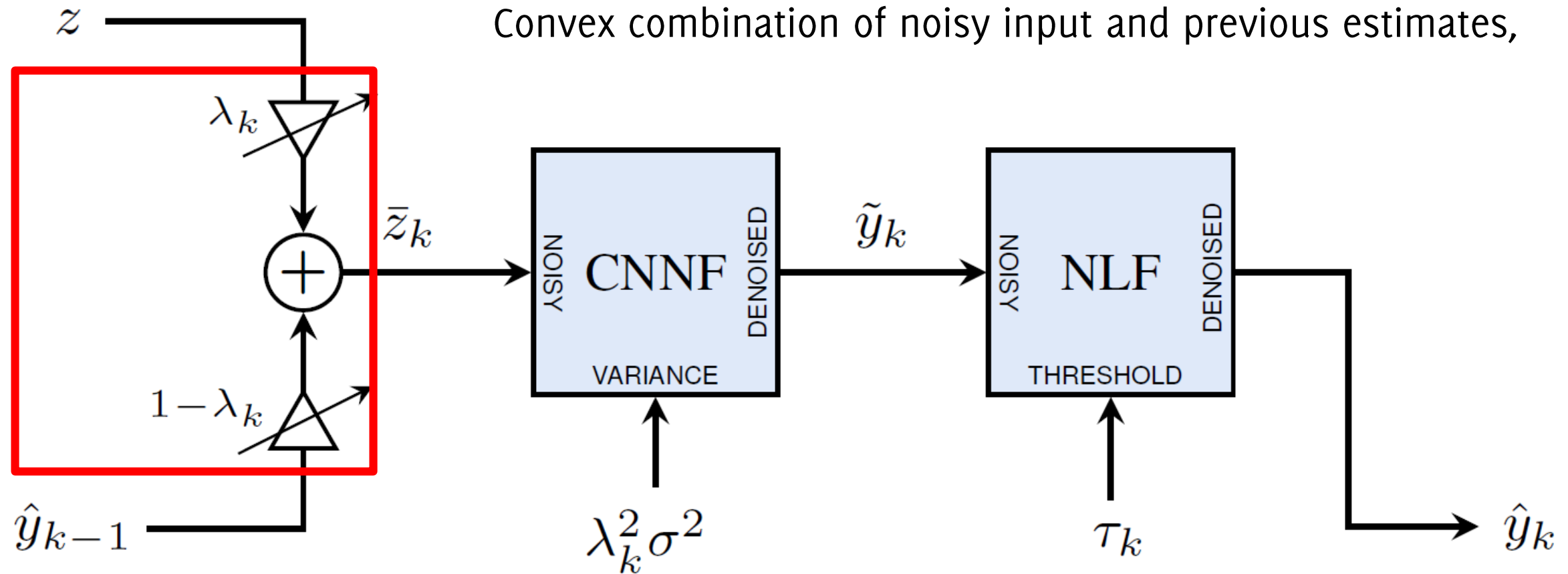
Due to the ERF size, CNN are biased towards local features. Introducing a nonlocal filtering stage can improve their performance

Alternating CNN and NLF results in a powerful denoiser where the learned filters by the CNN are used over an extended receptive field

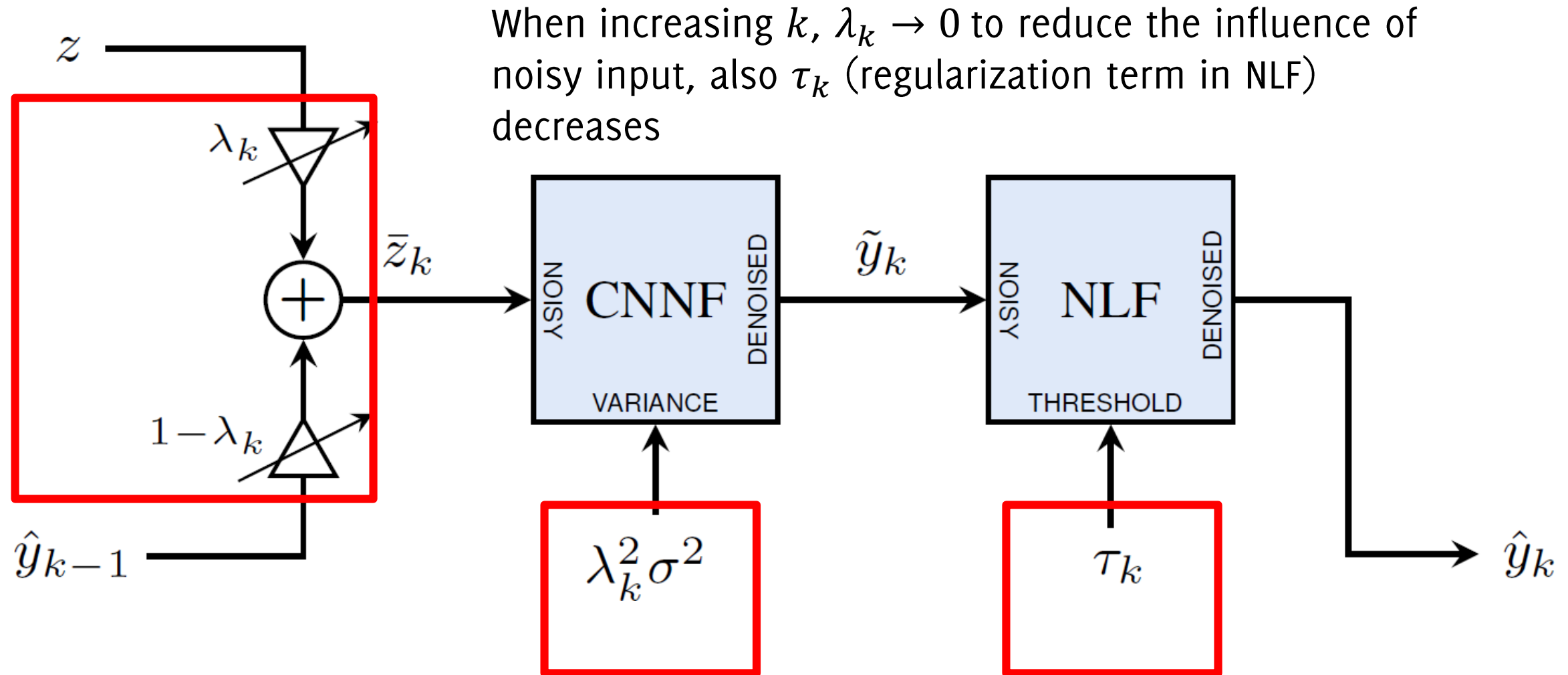
## NN<sub>3</sub>D: THE ITERATIVE SCHEME



## NN<sub>3</sub>D: THE ITERATIVE SCHEME



## NN<sub>3</sub>D: THE ITERATIVE SCHEME



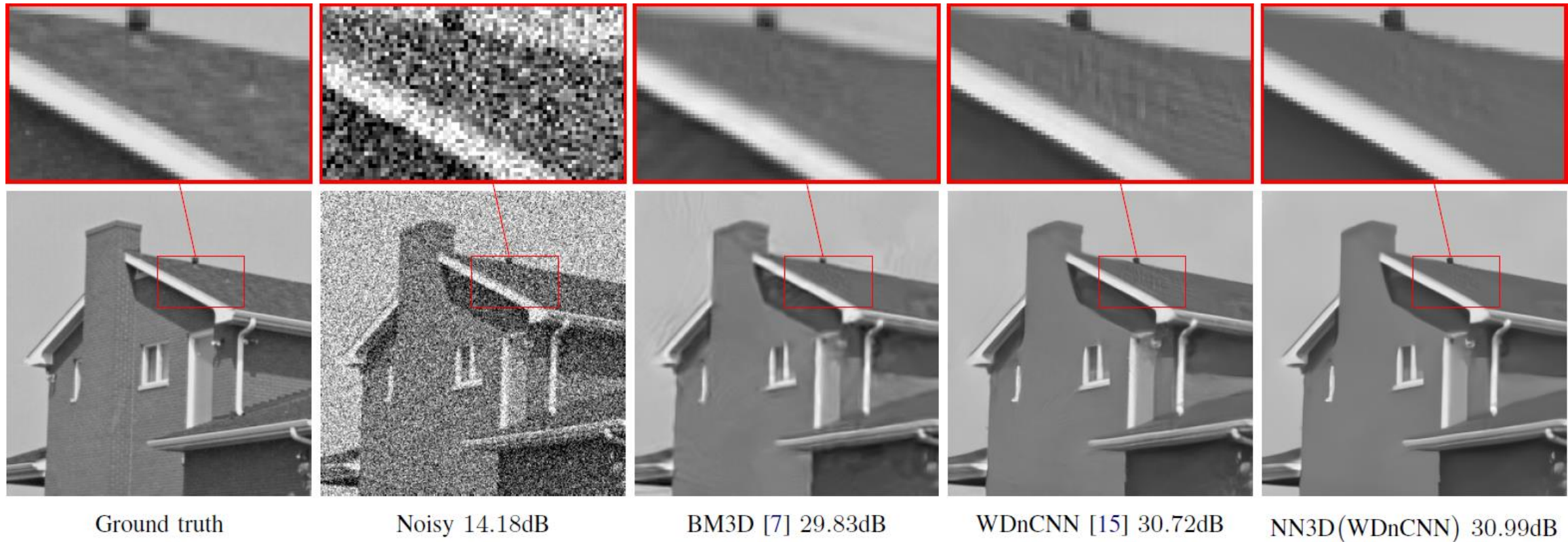
# NN3D PERFORMANCE

NN3D uses grouping and group-wise Haar filtering,  $K = 2$

TABLE I  
PSNR (dB) PERFORMANCE OF THE PROPOSED NN3D VS COMPETITIVE STATE-OF-THE-ART METHODS. ITALIC RESULTS IN THE BASELINE WDnCNN FOR  $\sigma = 75$  INDICATE SCALING OF THE NOISY INPUT BY THE FACTOR 50/75 TO MATCH THE MODEL TRAINED FOR  $\sigma = 50$ .

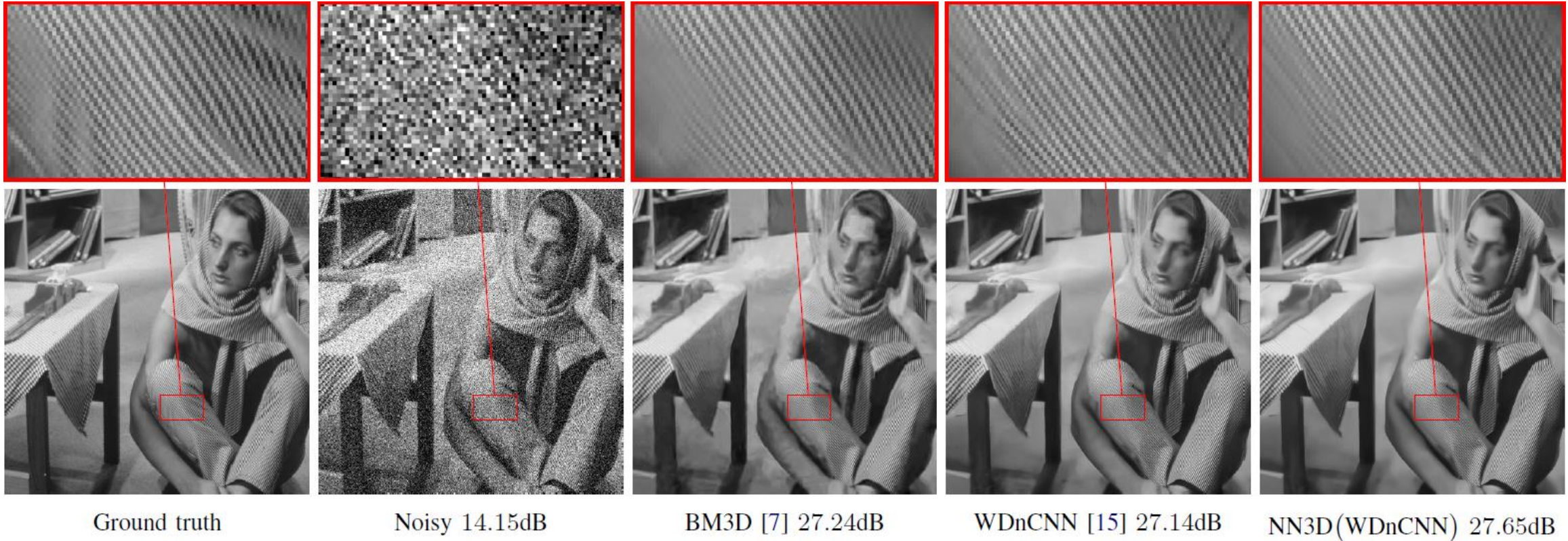
Dataset	$\sigma$	BM3D	DnCNN [11]	NN3D(DnCNN)	FFDNet [18]	NN3D(FFDNet)	WDnCNN [15]	NN3D(WDnCNN)
<i>Set12</i>	30	29.12	29.54	29.66	29.63	29.66	29.91	29.96
	50	26.70	27.19	27.32	27.33	27.39	27.48	27.61
	75	24.90	25.22	25.43	25.51	25.61	<i>25.64</i>	25.81
<i>BSD68</i>	30	27.75	28.36	28.41	28.38	28.37	28.56	28.56
	50	25.63	26.23	26.27	26.30	26.29	26.39	26.42
	75	24.22	24.64	24.71	24.79	24.80	<i>24.85</i>	24.91
<i>Urban100</i>	30	28.75	28.88	29.23	29.06	29.18	29.84	30.04
	50	25.95	26.28	26.63	26.55	26.76	27.08	27.49
	75	23.93	23.99	24.45	24.53	24.81	<i>25.06</i>	25.53

# NN<sub>3</sub>D PERFORMANCE





# NN<sub>3</sub>D PERFORMANCE



# **CONCLUDING REMARKS**

## CNN FOR IMAGE RESTORATION

- The research activity in the field has become very vivid in the last few years.
- The performance improvement achieved by deep CNN was not as clear as for visual recognition problem
  - Now these achieve the state of the art in terms of performance
  - Still, traditional algorithms are more efficient. This is crucial given the large image sizes (compared to visual recognition).  
For this reason, our phones do not have CNN for image restoration, yet
- Architectures are becoming more and more complex
- Research directions include
  - Efficient solutions
  - Practical training options (without using a noise model or clean images)
  - Robustness to noise models

# CNN FOR IMAGE RESTORATION

Research is often inspired by traditional priors, designing neural architectures able to promote these

- Self-similarity
- Sparsity
- Multiscale processing

CNN are now more recently used as powerful prior for natural images, inside iterative denoising schemes (Residual Unfolding Networks)