

Tutorial 11: Introduction to CNNs

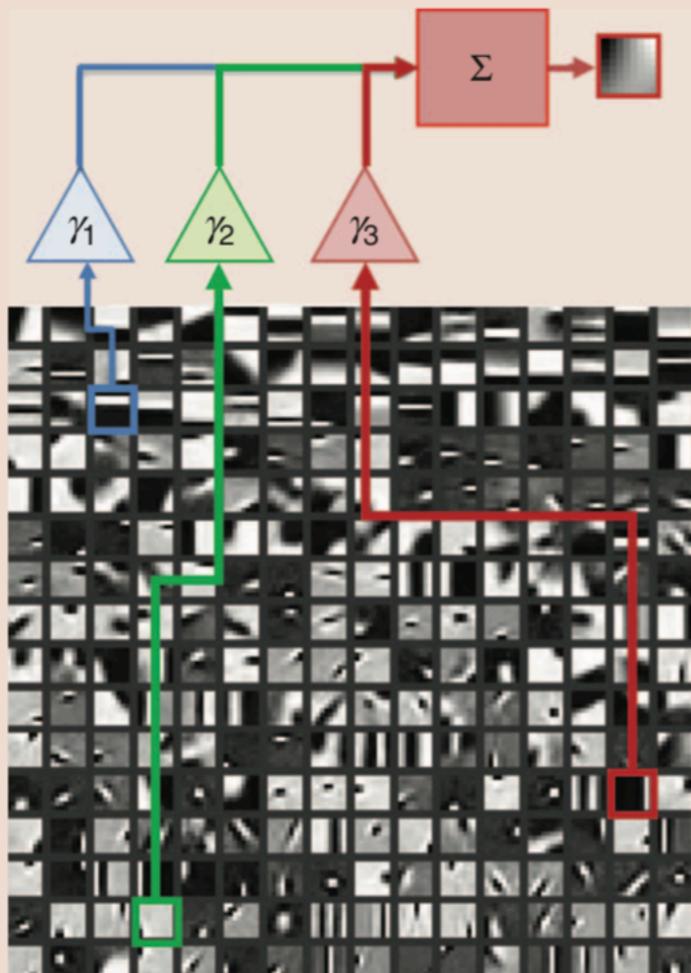
Digital Image Processing (236860)



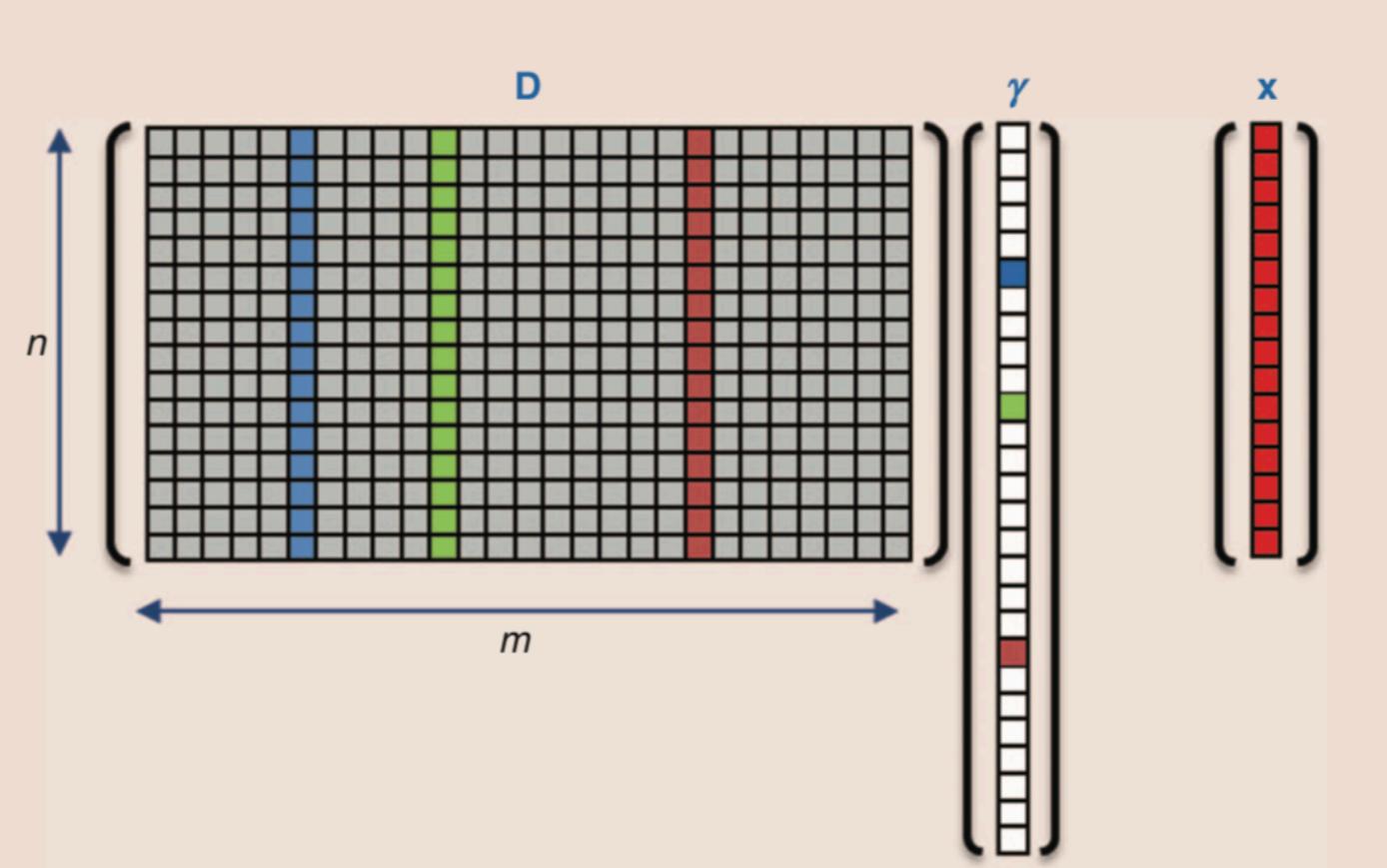
Agenda

- Intuition
- Neural networks
- Convolutional neural networks
- Deep learning for solving inverse problems

Reminder



(a)



(b)

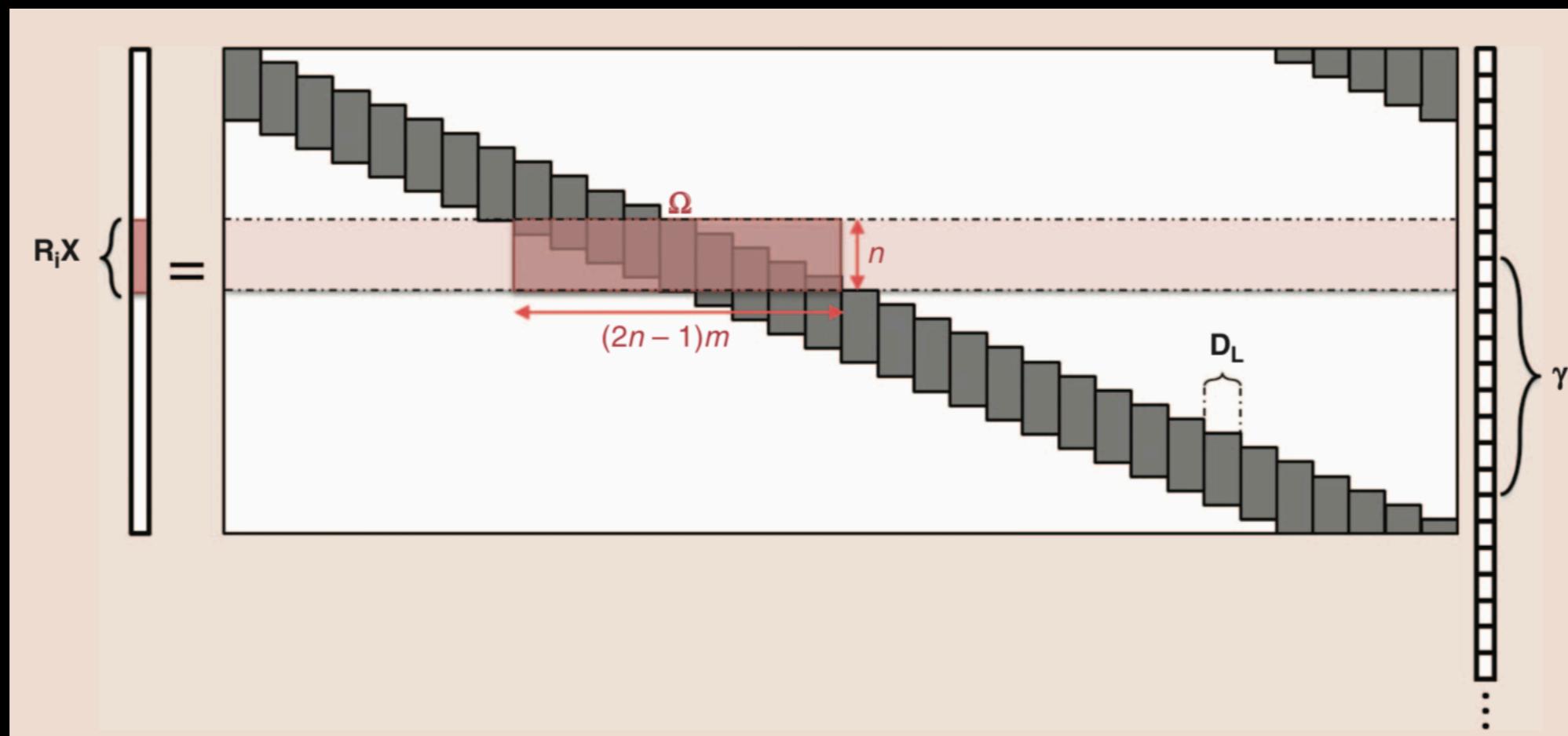
Difficulties of dictionary learning

- Slow (to train and to use)
- Cannot be used in high-dimensions (must work on patches)
- Why?
 - Too many degrees of freedom
 - Multiplying by \mathbf{D} is costly
- Solution?
 - Force \mathbf{D} to have a certain structure.

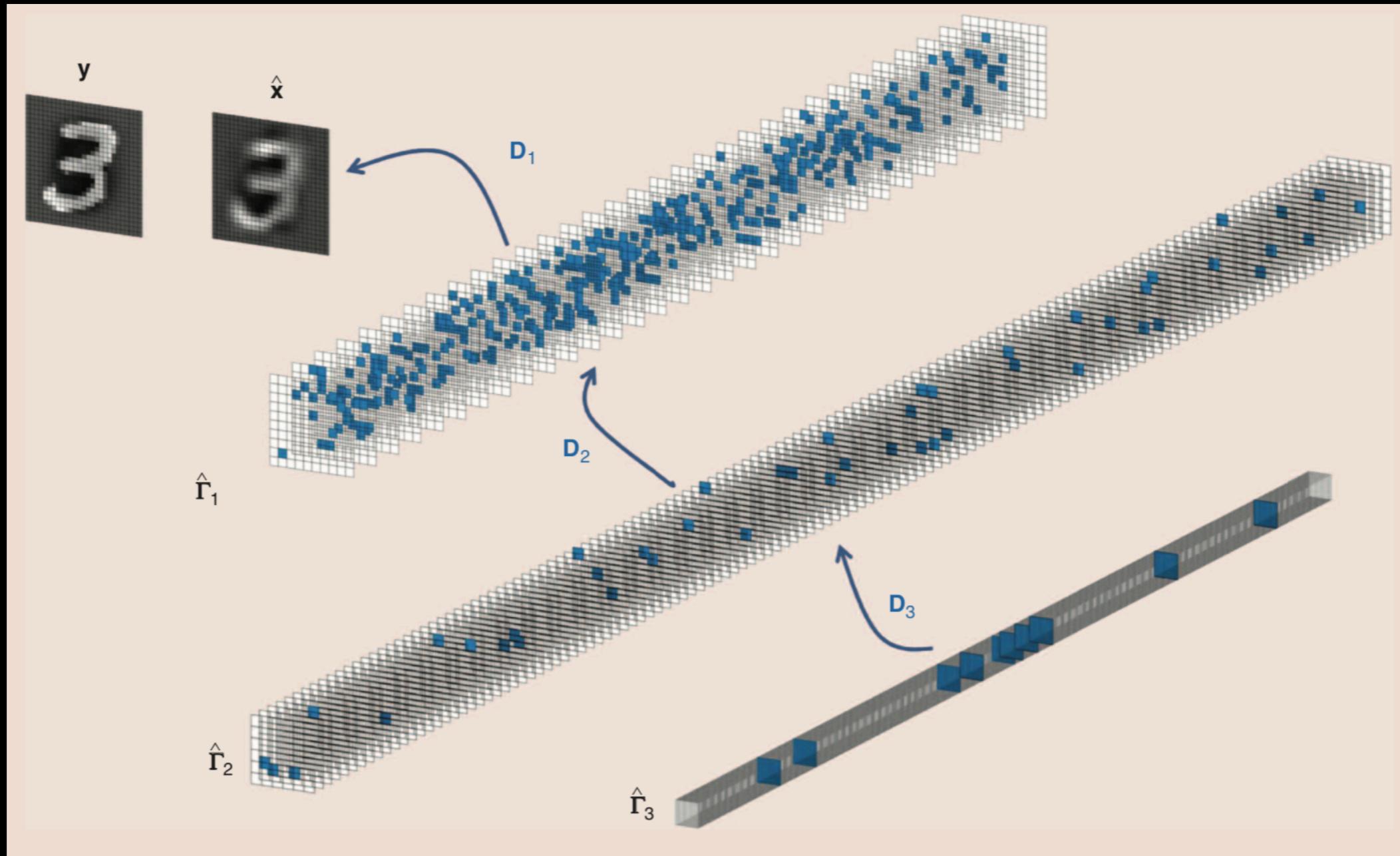
$$\mathbf{X} = \sum_{i=1}^m \mathbf{d}_i * \Gamma_i$$

Convolutional sparse coding (CSC)

$$\mathbf{X} = \sum_{i=1}^m \mathbf{C}_i \boldsymbol{\Gamma}_i = [\mathbf{C}_1 \ \mathbf{C}_2 \ \dots \ \mathbf{C}_m] \begin{bmatrix} \boldsymbol{\Gamma}_1 \\ \boldsymbol{\Gamma}_2 \\ \vdots \\ \boldsymbol{\Gamma}_m \end{bmatrix} = \mathbf{D}\boldsymbol{\Gamma}.$$

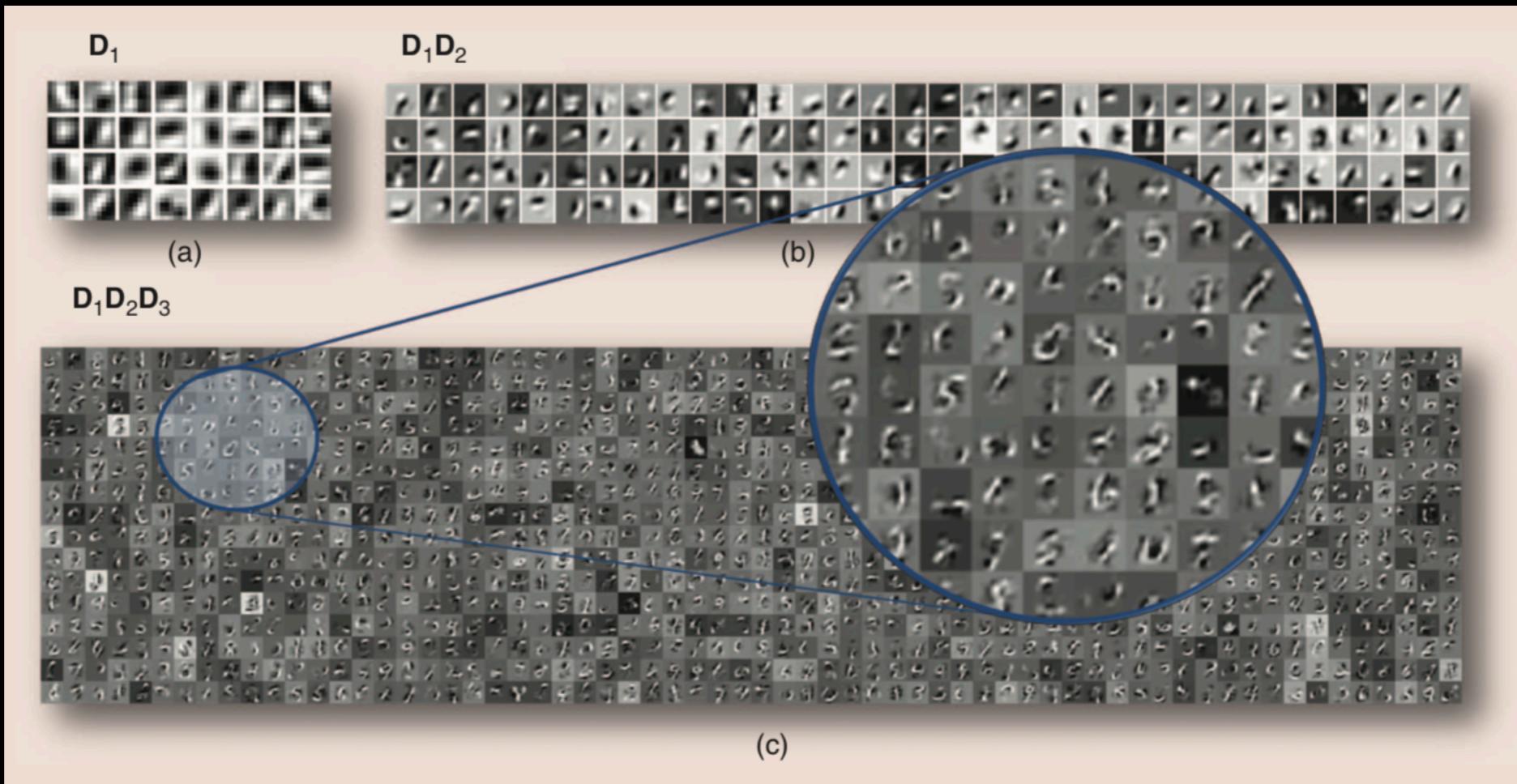


Multi-layered CSC (ML-CSC)



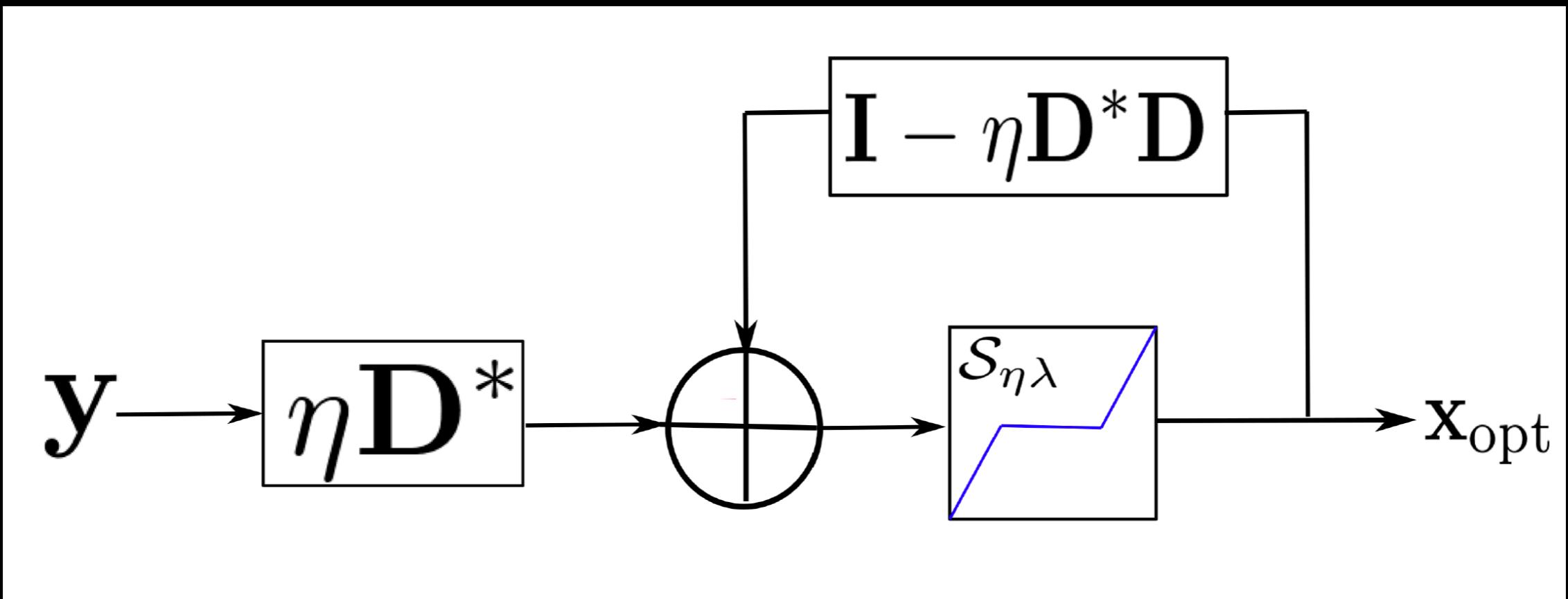
Papyan, Vardan, et al. "Theoretical Foundations of Deep Learning via Sparse Representations."

Multi-layered CSC (ML-CSC)



Papyan, Vardan, et al. "Theoretical Foundations of Deep Learning via Sparse Representations."

ISTA



Single iteration ISTA for CSC



First layer MAP estimator

$$\hat{\mathbf{c}}^1(y) = \arg \min_{\mathbf{c}^1 \geq 0} \frac{1}{2} \|y - \Phi^1 \mathbf{c}^1\|_2^2 + \lambda_1 \|\mathbf{c}^1\|_1 \quad \Phi^1: \mathbf{c}^1 \rightarrow \sum_{k=1}^{K_1} \phi_k^1 * c_k^1$$

$$\hat{\mathbf{c}}^1(y) \approx \tilde{\mathbf{c}}^1(y) = S_{\eta_1 \lambda_1}^+(\eta_1 \Phi^{1*} y) \quad \Phi^{1*}: y \rightarrow (\bar{\phi}_1^1 * y, \dots, \bar{\phi}_{K_1}^1 * y)$$

$$= S_{\lambda_1}^+(\Phi^{1*} y) = [\Phi^{1*} y - \lambda_1]_+$$

$$= ([\bar{\phi}_1^1 * y - \lambda_1]_+, \dots, [\bar{\phi}_{K_1}^1 * y - \lambda_1]_+)$$

Single iteration ISTA for CSC



Next layer MAP estimator

$$\hat{\mathbf{c}}^k(\mathbf{c}^{k-1}) = \arg \min_{\mathbf{c}^k \geq 0} \frac{1}{2} \|\mathbf{c}^{k-1} - \Phi^k \mathbf{c}^k\|_2^2 + \lambda_k \|\mathbf{c}^k\|_1$$

$$\hat{\mathbf{c}}^k(\mathbf{c}^{k-1}) \approx \tilde{\mathbf{c}}^k(\tilde{\mathbf{c}}^{k-1}) = S_{\eta_k \lambda_k}^+(\eta_k \Phi^{k*} \tilde{\mathbf{c}}^{k-1})$$

$$= ([\bar{\Phi}_1^k * \tilde{\mathbf{c}}^{k-1} - \lambda_k]_+, \dots, [\bar{\Phi}_{K_k}^k * \tilde{\mathbf{c}}^{k-1} - \lambda_k]_+)$$

$$= \left(\left[\sum_{i=1}^{K_{k-1}} \bar{\phi}_{1i}^k * \tilde{c}_i^{k-1} - \lambda_k \right]_+, \dots, \left[\sum_{i=1}^{K_{k-1}} \bar{\phi}_{K_k i}^k * \tilde{c}_i^{k-1} - \lambda_k \right]_+ \right)$$

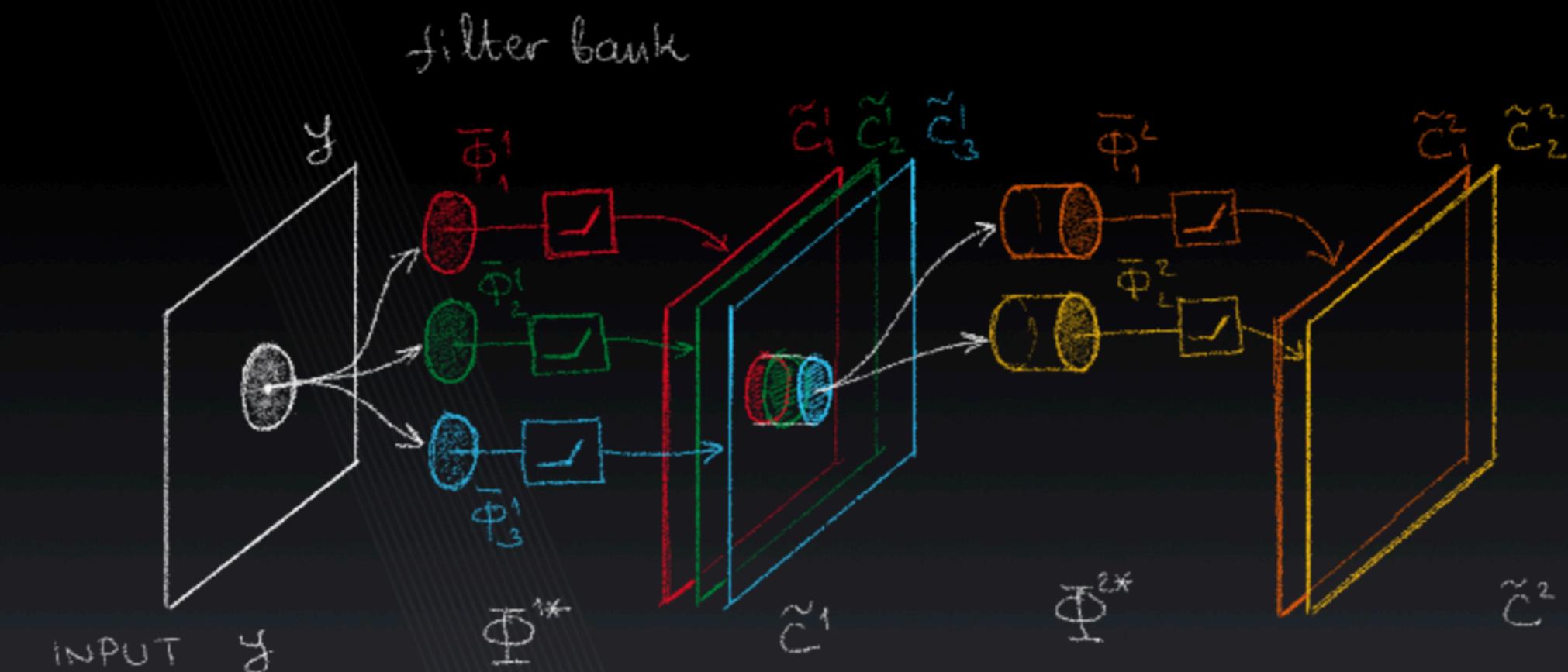
Convolutional neural network (CNN)



Visual Sensing Theory
& Applications Laboratory



Convolutional neural network (CNN)



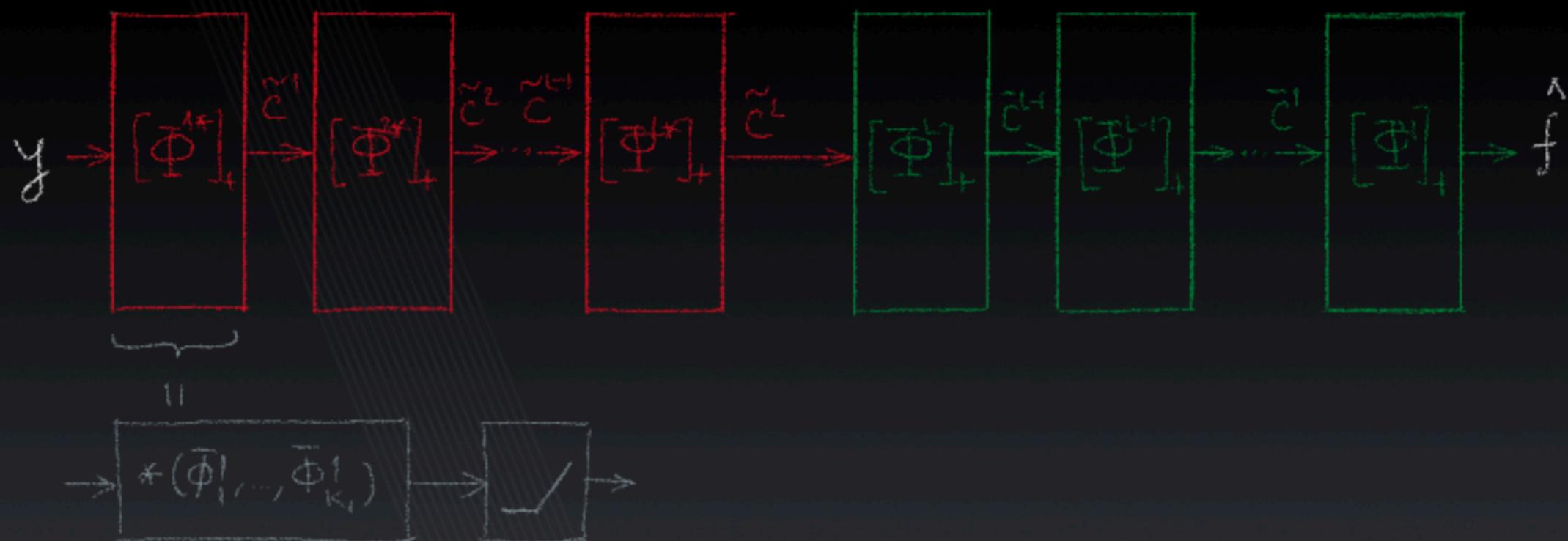
Encoder-decoder



Visual Sensing Theory
& Applications Laboratory



Convolutional autoencoder



Back to our main interest

Inverse problems

Inverse problems

$$\text{Measurement} \quad \mathbf{y} = \begin{matrix} \text{Degradation operator} \\ \mathbf{D} \end{matrix} \mathbf{x} + \boldsymbol{\eta} \quad \text{vector}$$

If D is:

1. Focal blur
2. Motion blur
3. Identity
4. Identity with missing values
5. Random gaussian matrix
6. Projection matrix
and so on.....

Then recovering x from y is:

1. Super-resolution
2. Deblurring
3. Denoising
4. Inpainting
5. Compressed sensing (x is sparse)
6. Tomography
and so on.....

Supervised learning

$$\mathbf{y} = \mathbf{D}\mathbf{x} + \eta$$

\mathbf{y} – degraded image
 \mathbf{x} – clean image

Ideally, we would like to minimize the following risk:

$$\arg \min_{\theta} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \ell(f_{\theta}(\mathbf{y}), \mathbf{x})$$

$f_{\theta}(.)$ = parametric inverse model θ = parameters

Supervised learning

$$\mathbf{y} = \mathbf{D}\mathbf{x} + \eta$$

\mathbf{y} – degraded image
 \mathbf{x} – clean image

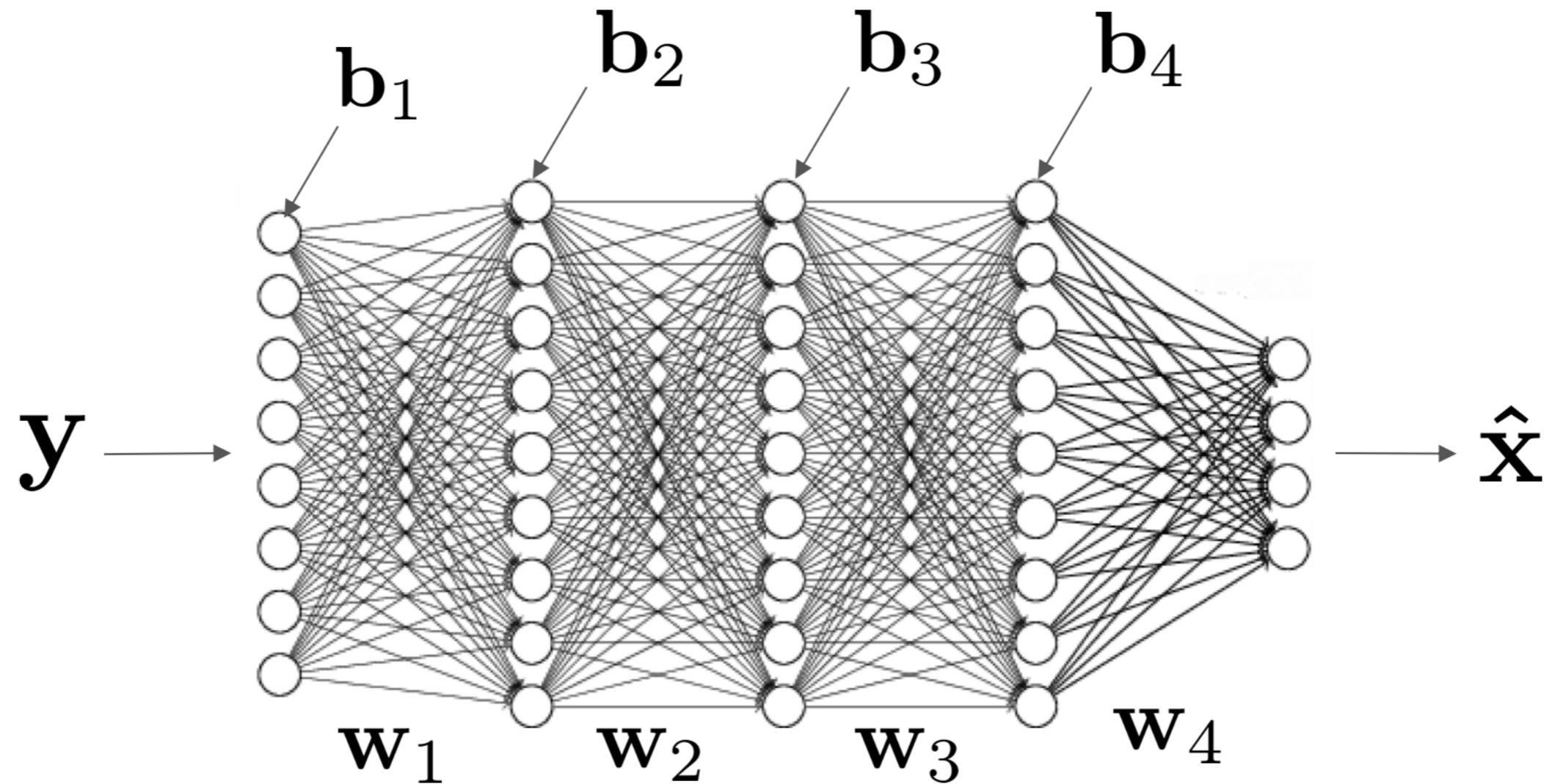
Empirically, we do not have the distribution of \mathbf{x} , but we do have access to a few examples. Then the loss function becomes:

$$\arg \min_{\theta} \sum_{i=1}^N \ell(f_{\theta}(\mathbf{y}_i), \mathbf{x}_i)$$

$f_{\theta}(.)$ = parametric inverse model θ = parameters

Samples: $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1\dots M}$

Neural networks

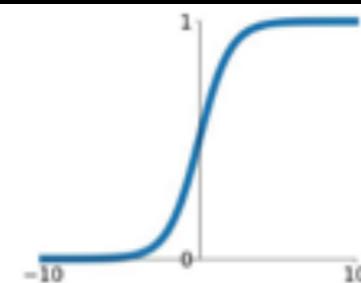


$$\hat{\mathbf{x}} = \mathbf{w}_4(\sigma(\mathbf{w}_3(\sigma(\mathbf{w}_2(\sigma(\mathbf{w}_1\mathbf{y} + \mathbf{b}_1)) + \mathbf{b}_2)) + \mathbf{b}_3)) + \mathbf{b}_4$$

Activation functions

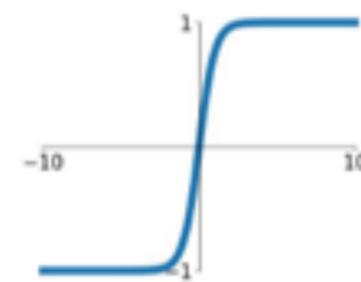
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



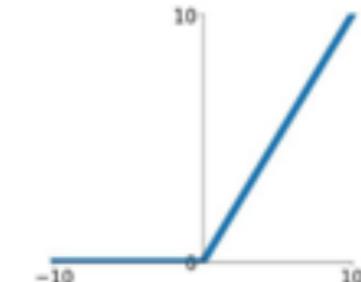
tanh

$$\tanh(x)$$



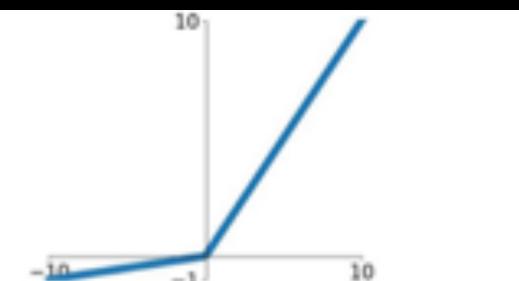
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

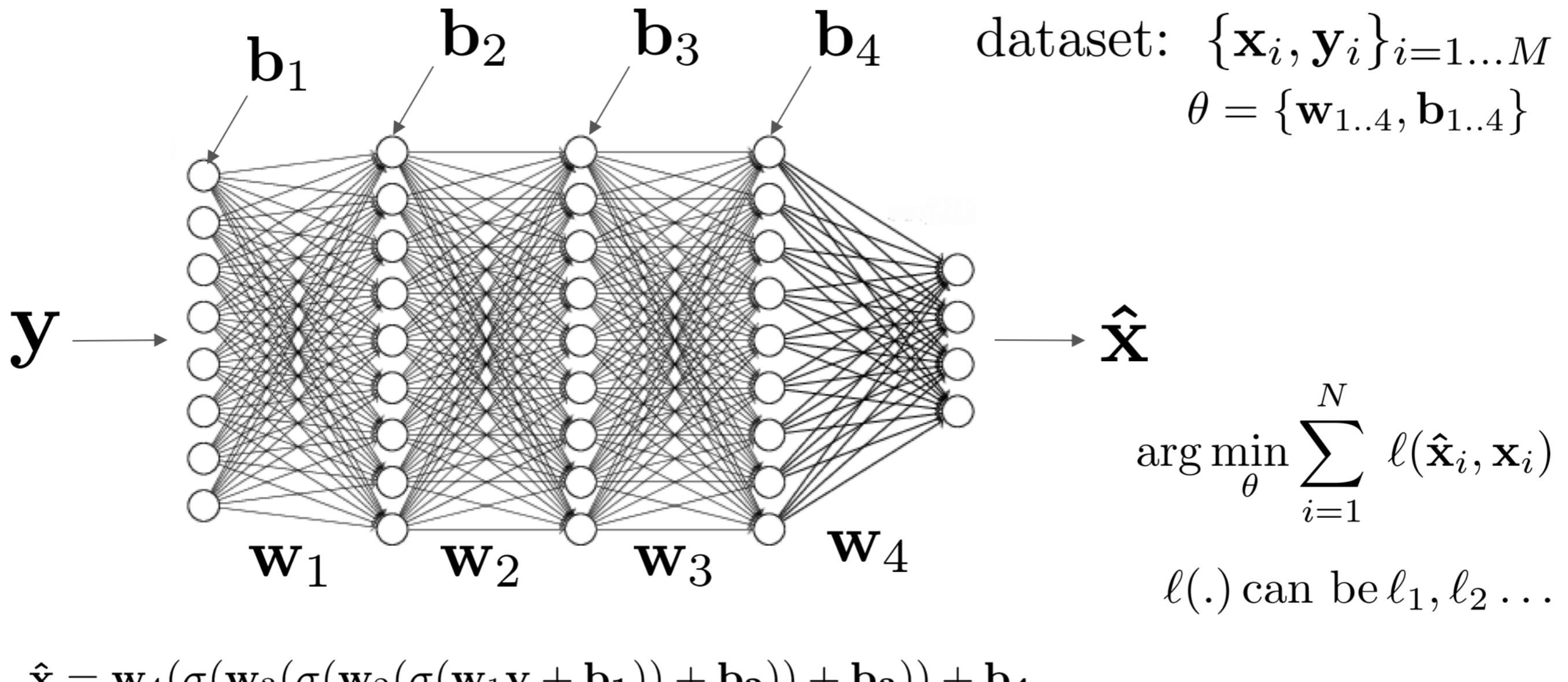
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Loss function



Optimization



Visual Sensing Theory
& Applications Laboratory



Gradient descent

Naïve optimization algorithm: gradient descent

Initialization: start with some Θ^0

For $k = 1, \dots$ **until convergence**

Update: $\Theta^k \leftarrow \Theta^{k-1} - \eta_k \nabla L(\Theta^{k-1})$

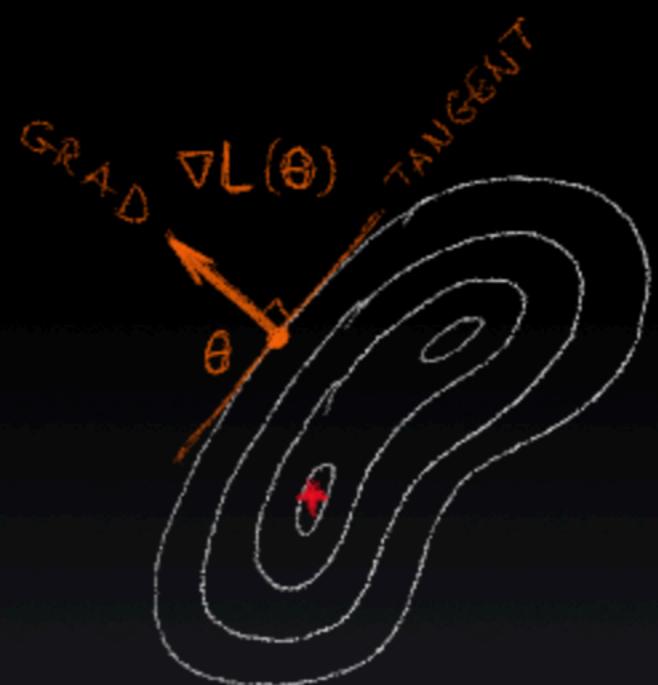


Image denoising using fully-connected neural networks

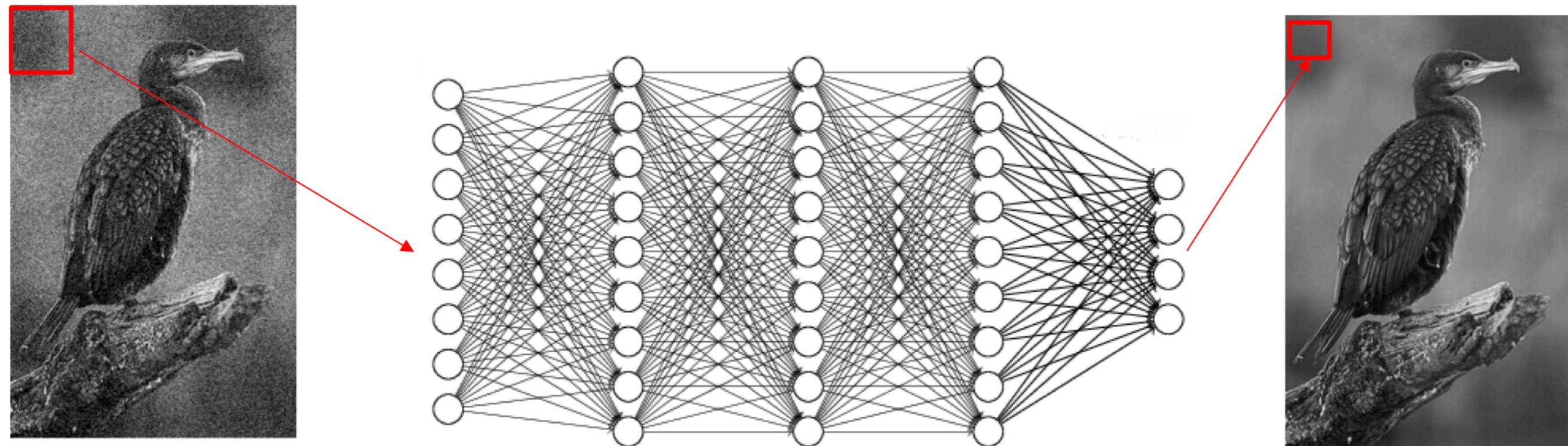


Image denoising using fully-connected neural networks

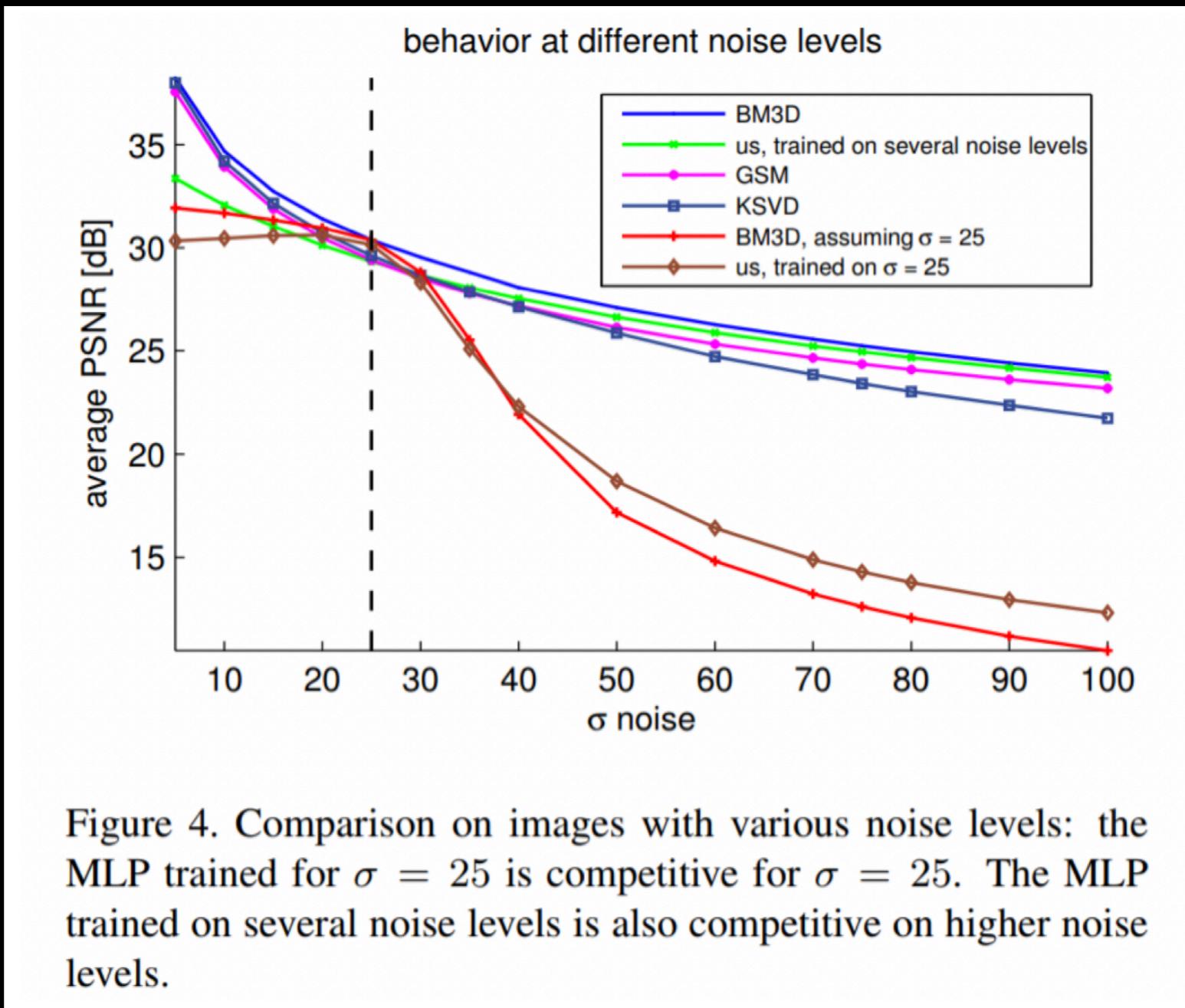


Figure 4. Comparison on images with various noise levels: the MLP trained for $\sigma = 25$ is competitive for $\sigma = 25$. The MLP trained on several noise levels is also competitive on higher noise levels.

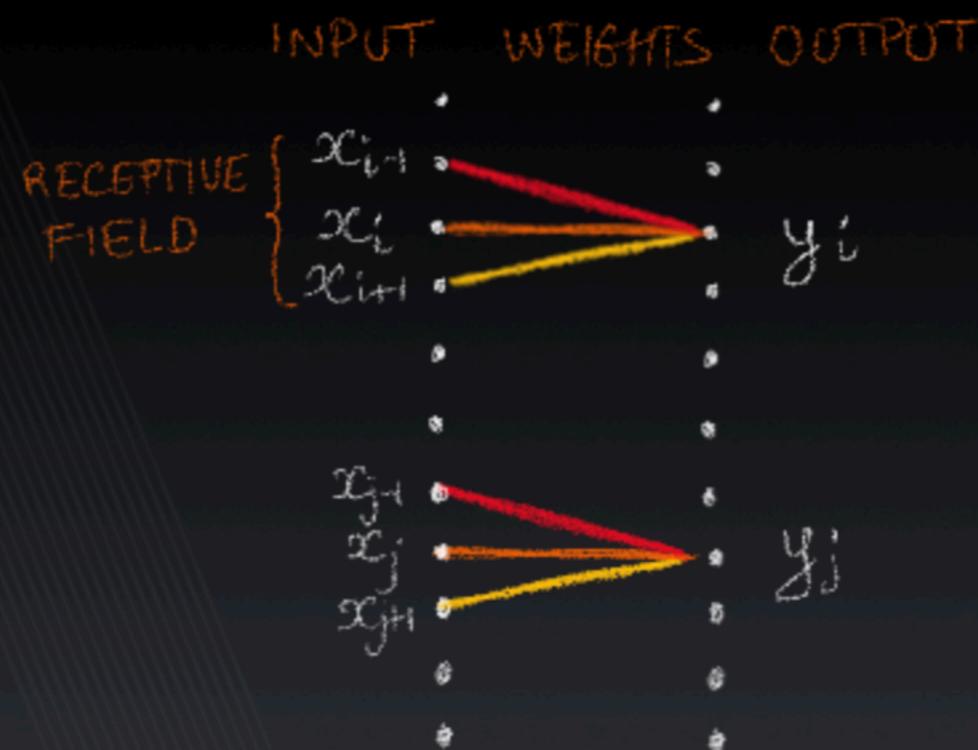
Problems

- Too many degrees of freedom.
- Slow and hard to train.
- High memory consumption.
- Each patch is being evaluated separately.
- The architecture doesn't capture the spatial structure of an image.
- Solution?

Solution

Weight sharing

Different locations receive same processing



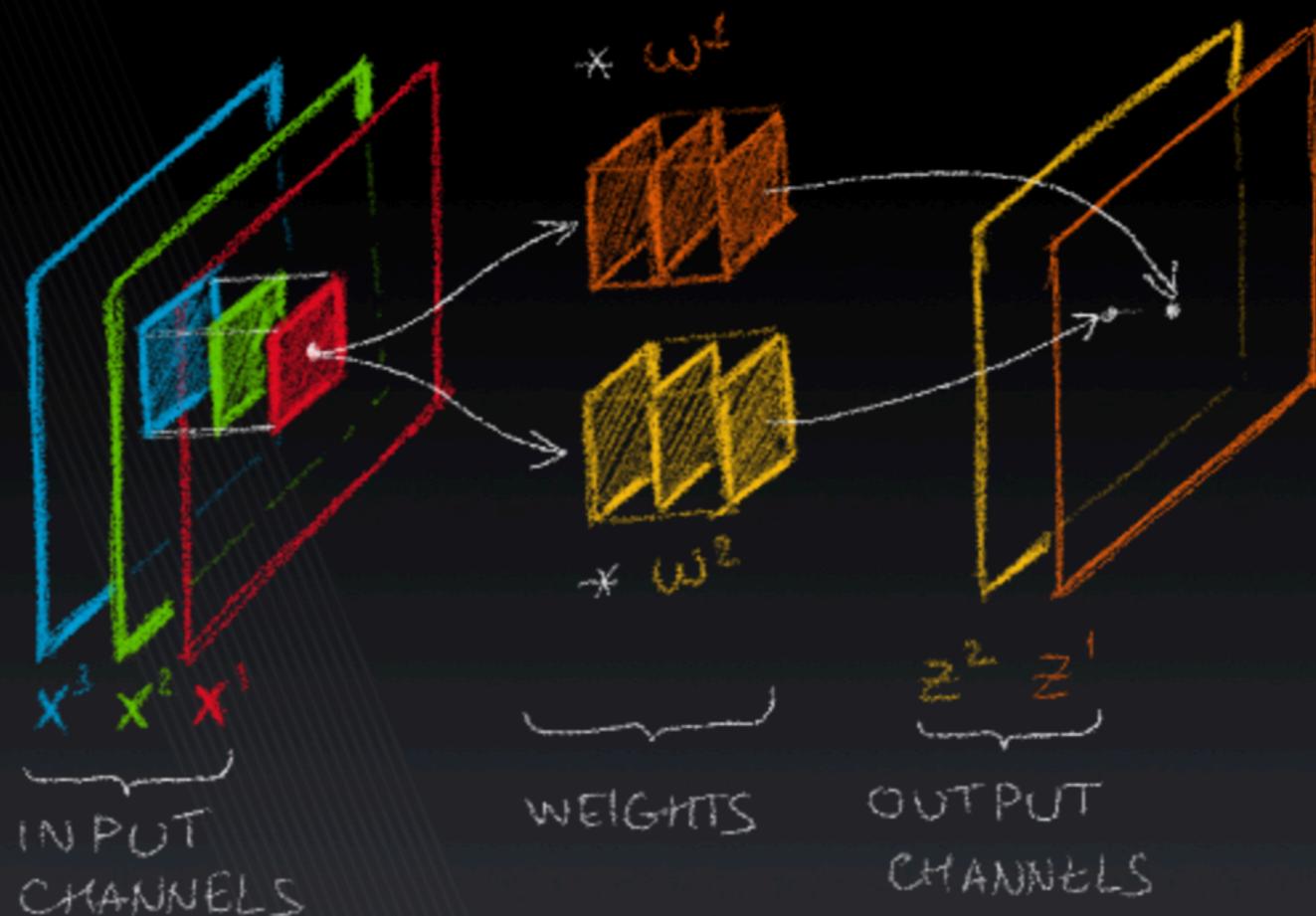
Convolutional neural networks (CNNs)



Visual Sensing Theory
& Applications Laboratory



Convolutional layers



Pooling



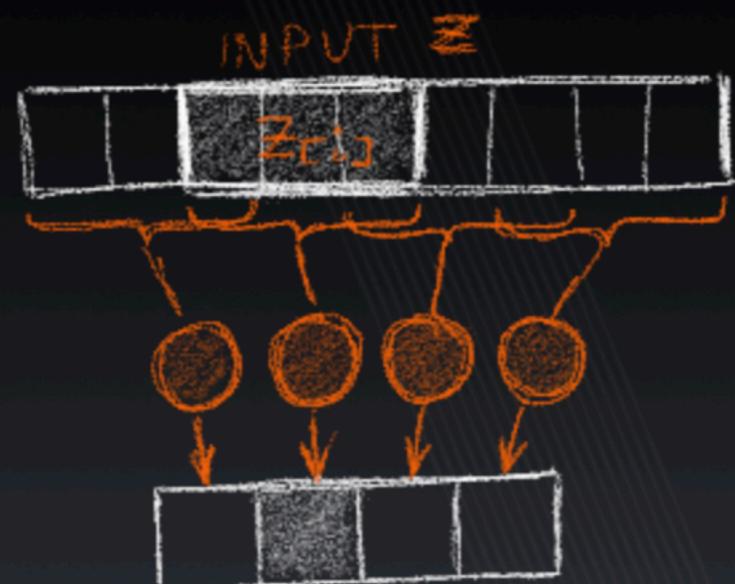
Visual Sensing Theory
& Applications Laboratory



Pooling

Break input into (possibly overlapping) **windows** $\{\mathbf{z}_{[i]}\}_{i=1}^m$

Apply a **transformation** to each window $\mathbf{y} = (\psi(\mathbf{z}_{[1]}), \dots, \psi(\mathbf{z}_{[m]}))$



$\max \mathbf{z}_{[i]}$

max pooling

$\text{avg } \mathbf{z}_{[i]}$

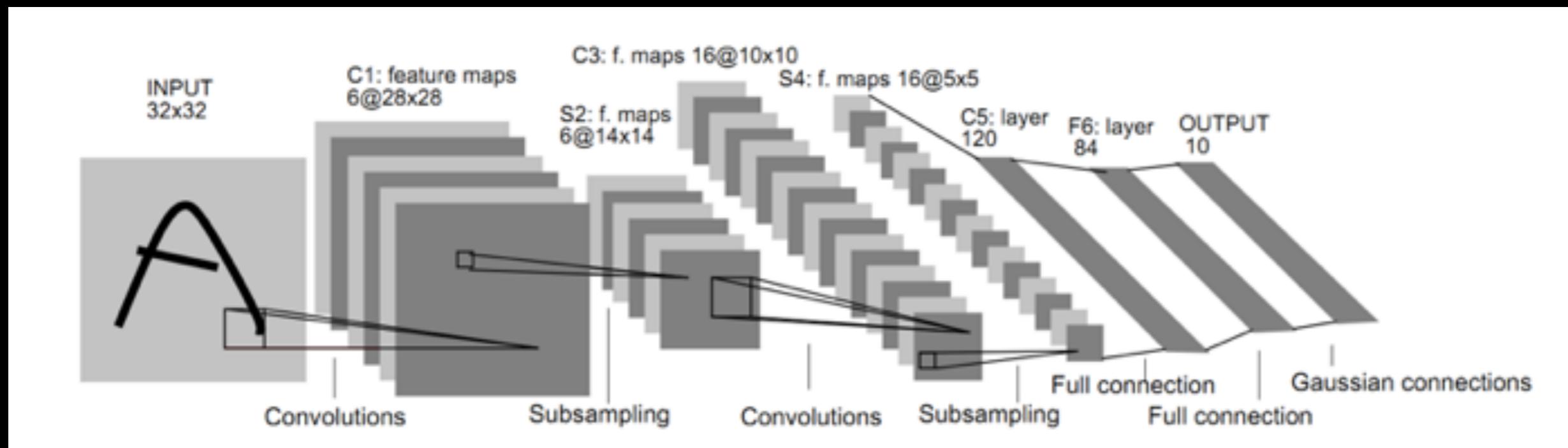
average pooling

central sample of $\mathbf{z}_{[i]}$

striding

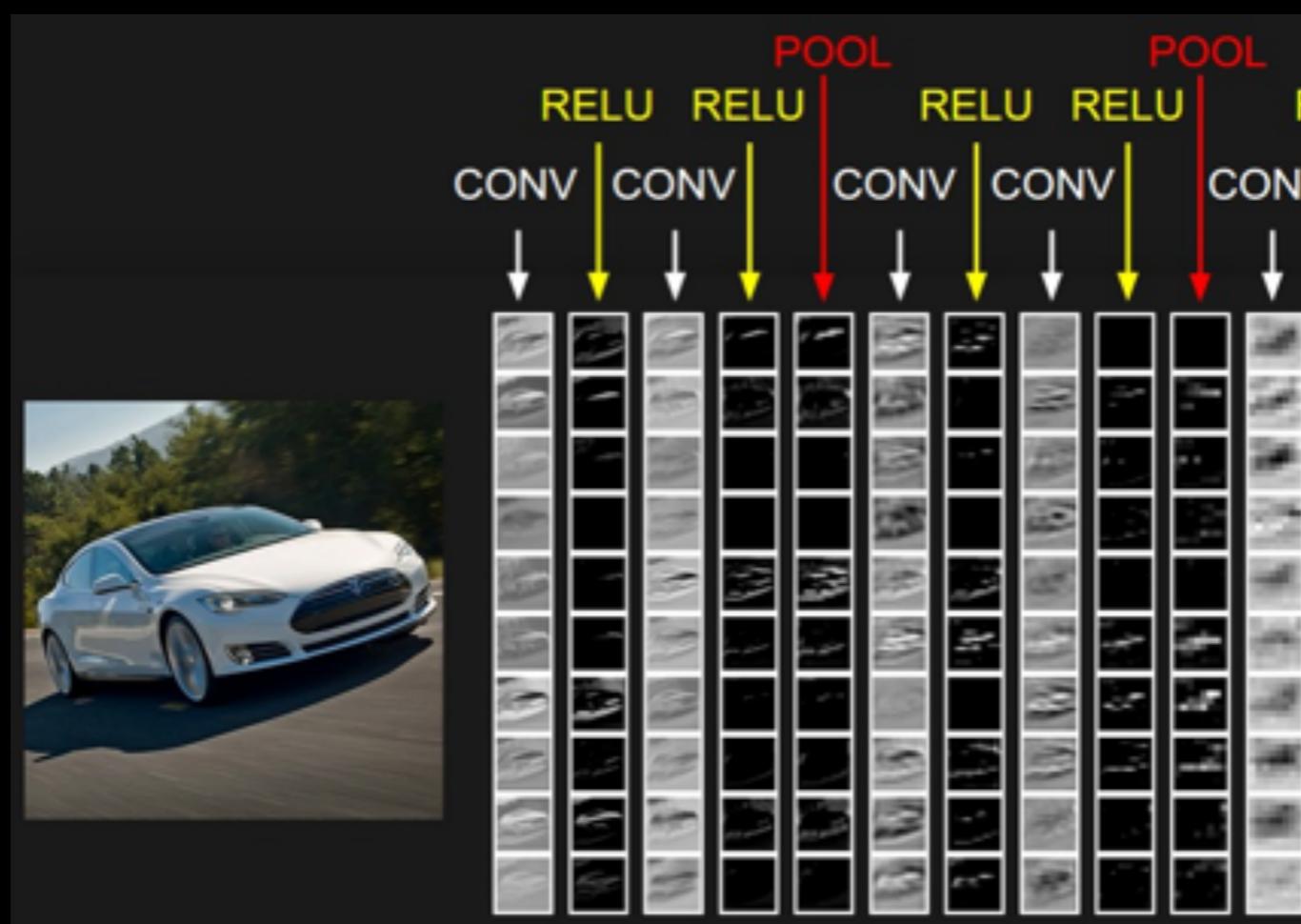
CNNs

- Repeating convolutions increase the receptive field.



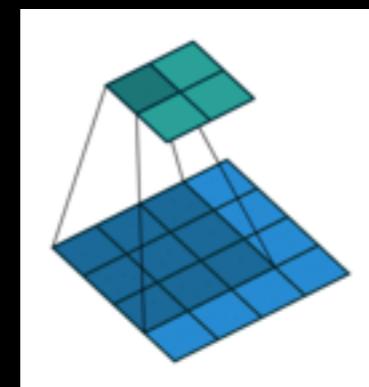
CNNs

- Convolutions capture local structure.
- Hierarchies of features can be obtained by stacking convolution layers.



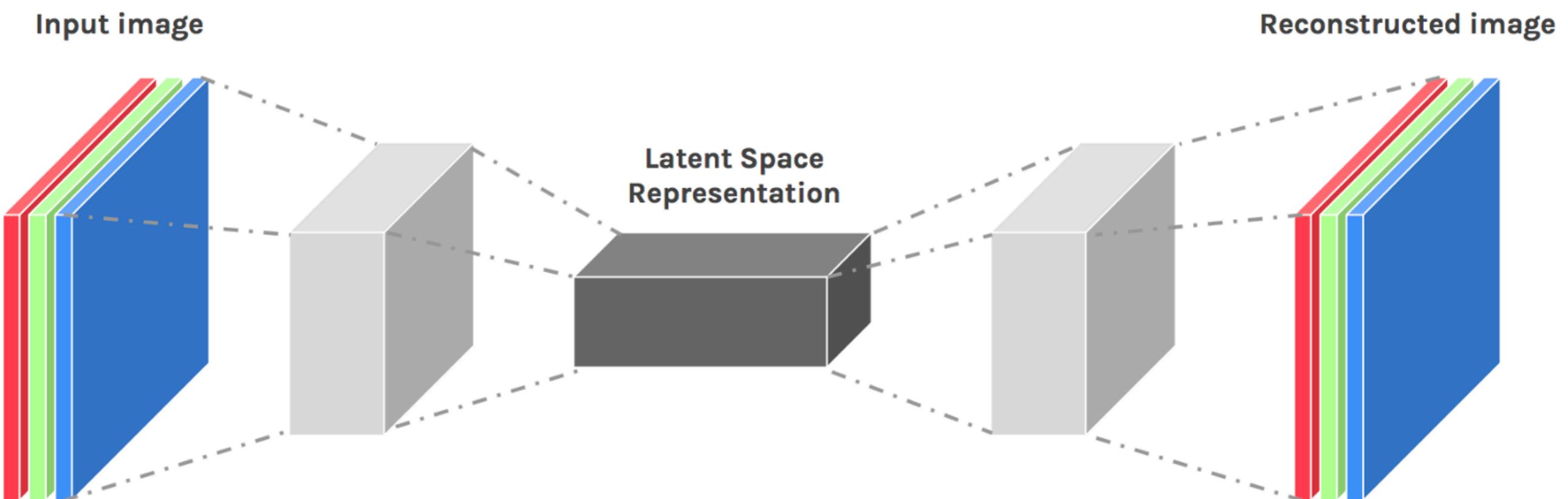
CNNs

- Each convolutions weight matrix is equivalent to a fully-connected network with the block diagonal structure.
- GPUs perform them efficiently as sparse matrix multiplication.



$$\begin{pmatrix} w_{0,0} & 0 & 0 & 0 \\ w_{0,1} & w_{0,0} & 0 & 0 \\ w_{0,2} & w_{0,1} & 0 & 0 \\ 0 & w_{0,2} & 0 & 0 \\ w_{1,0} & 0 & w_{0,0} & 0 \\ w_{1,1} & w_{1,0} & w_{0,1} & w_{0,0} \\ w_{1,2} & w_{1,1} & w_{0,2} & w_{0,1} \\ 0 & w_{1,2} & 0 & w_{0,2} \\ w_{2,0} & 0 & w_{1,0} & 0 \\ w_{2,1} & w_{2,0} & w_{1,1} & w_{1,0} \\ w_{2,2} & w_{2,1} & w_{1,2} & w_{1,1} \\ 0 & w_{2,2} & 0 & w_{1,2} \\ 0 & 0 & w_{2,0} & 0 \\ 0 & 0 & w_{2,1} & w_{2,0} \\ 0 & 0 & w_{2,2} & w_{2,1} \\ 0 & 0 & 0 & w_{2,2} \end{pmatrix}^T$$

Encoder-decoder



<https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694>

Encoder-decoder with skip connections

- $Y = X + \eta$
- Instead of learning to approximate X , learn to approximate the noise $Y - X$.

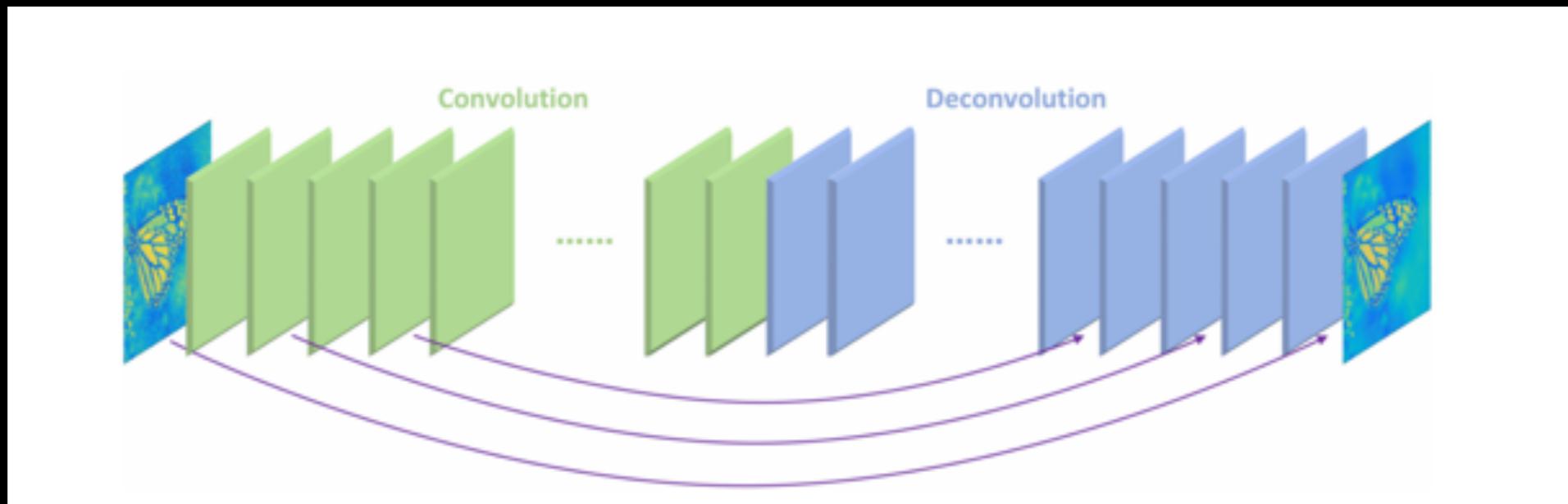


Image denoising

Table 1: Average PSNR and SSIM results of σ 10, 30, 50, 70 for the 14 images.

	PSNR									
	BM3D	EPLL	NCSR	PCLR	PGPD	WNNM	RED10	RED20	RED30	
$\sigma = 10$	34.18	33.98	34.27	34.48	34.22	34.49	34.62	34.74	34.81	
$\sigma = 30$	28.49	28.35	28.44	28.68	28.55	28.74	28.95	29.10	29.17	
$\sigma = 50$	26.08	25.97	25.93	26.29	26.19	26.32	26.51	26.72	26.81	
$\sigma = 70$	24.65	24.47	24.36	24.79	24.71	24.80	24.97	25.23	25.31	
	SSIM									
$\sigma = 10$	0.9339	0.9332	0.9342	0.9366	0.9309	0.9363	0.9374	0.9392	0.9402	
$\sigma = 30$	0.8204	0.8200	0.8203	0.8263	0.8199	0.8273	0.8327	0.8396	0.8423	
$\sigma = 50$	0.7427	0.7354	0.7415	0.7538	0.7442	0.7517	0.7571	0.7689	0.7733	
$\sigma = 70$	0.6882	0.6712	0.6871	0.6997	0.6913	0.6975	0.7012	0.7177	0.7206	

Low-dose CT reconstruction

- Convert “low-dose” CT image to a “high-dose” one.
- Less dose = less energy of X-rays entering the body = less ionising radiation.
- Dataset synthetically generated using Poisson noise.

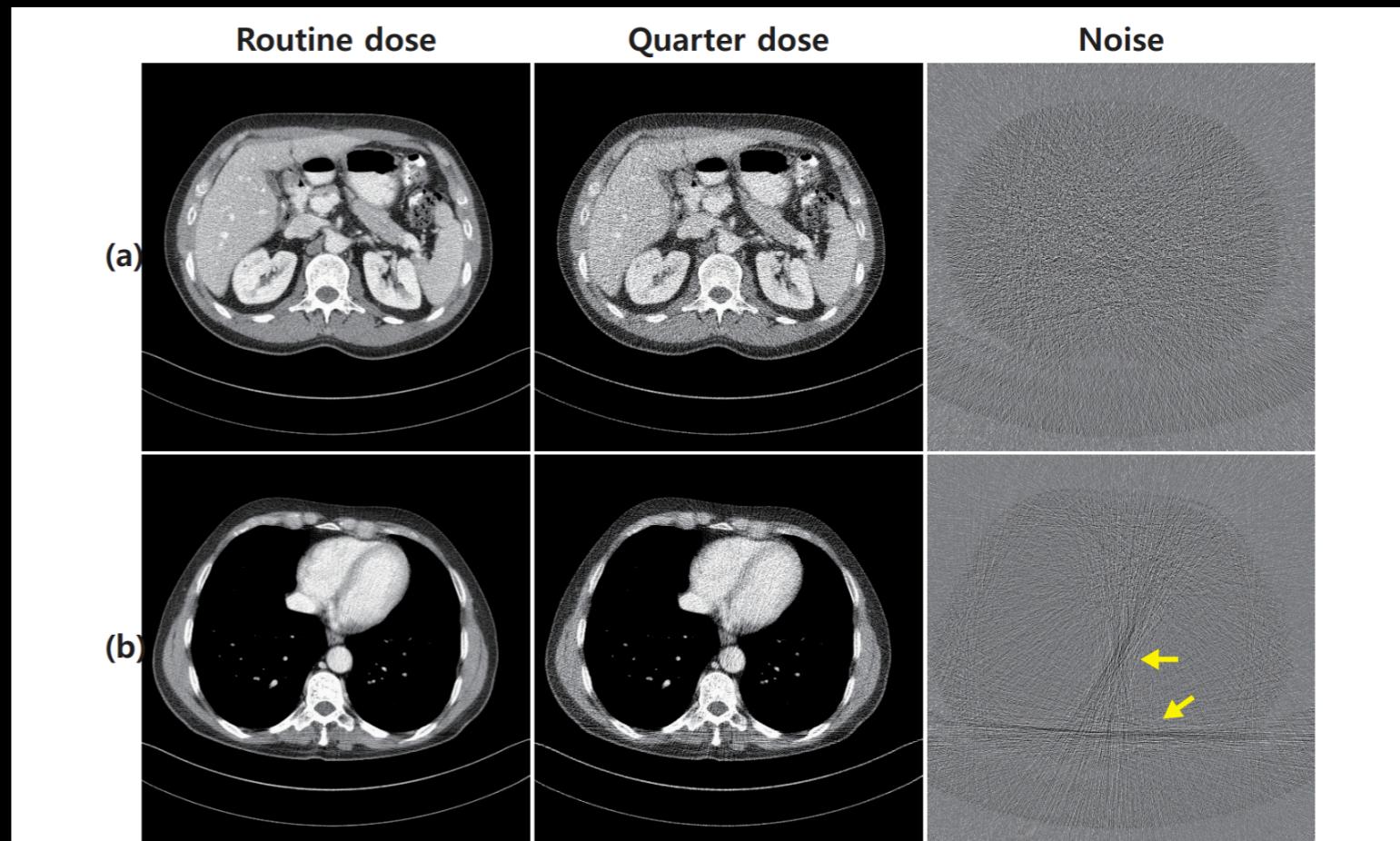


FIG. 1 Various noise patterns in low-dose CT images: (a) Gaussian noise, and (b) streaking artifacts

Low-dose CT reconstruction

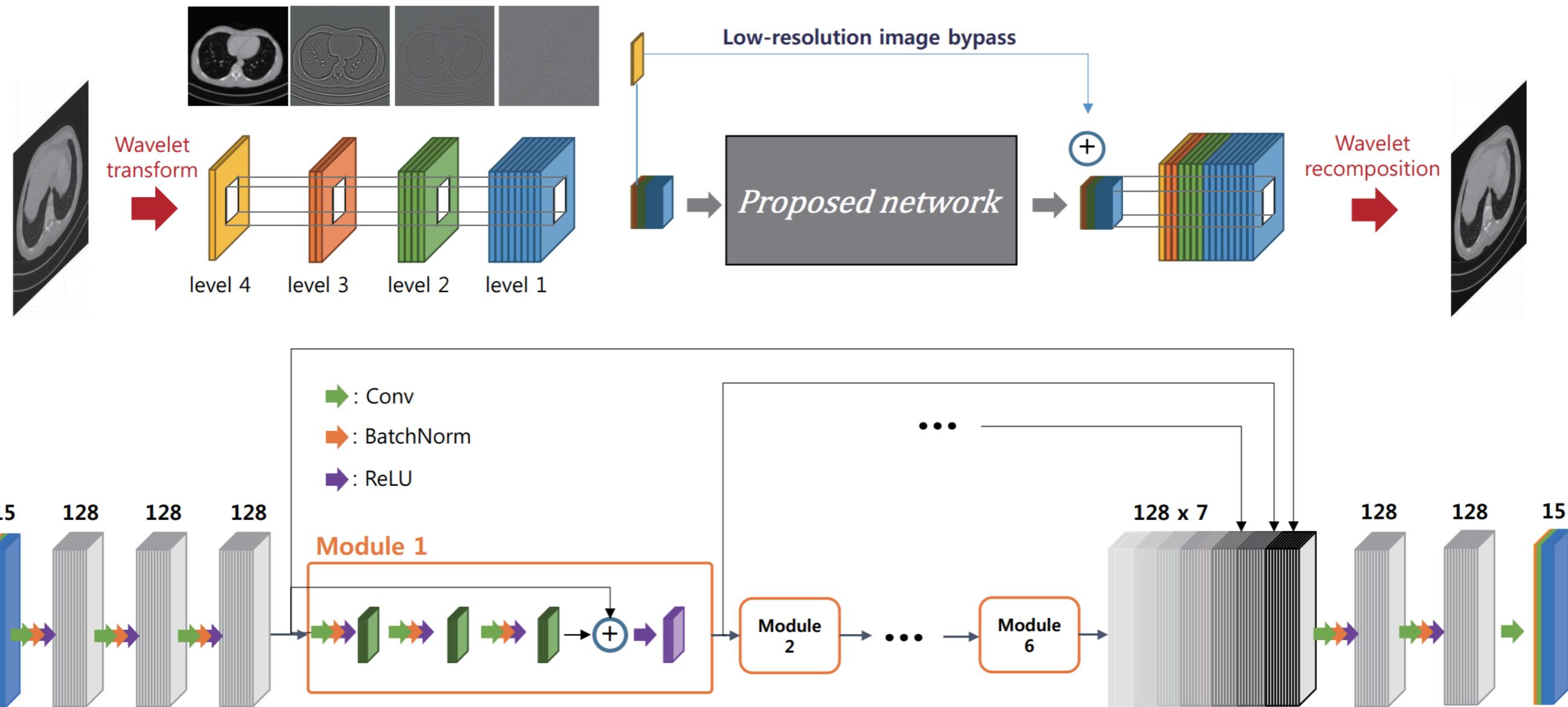
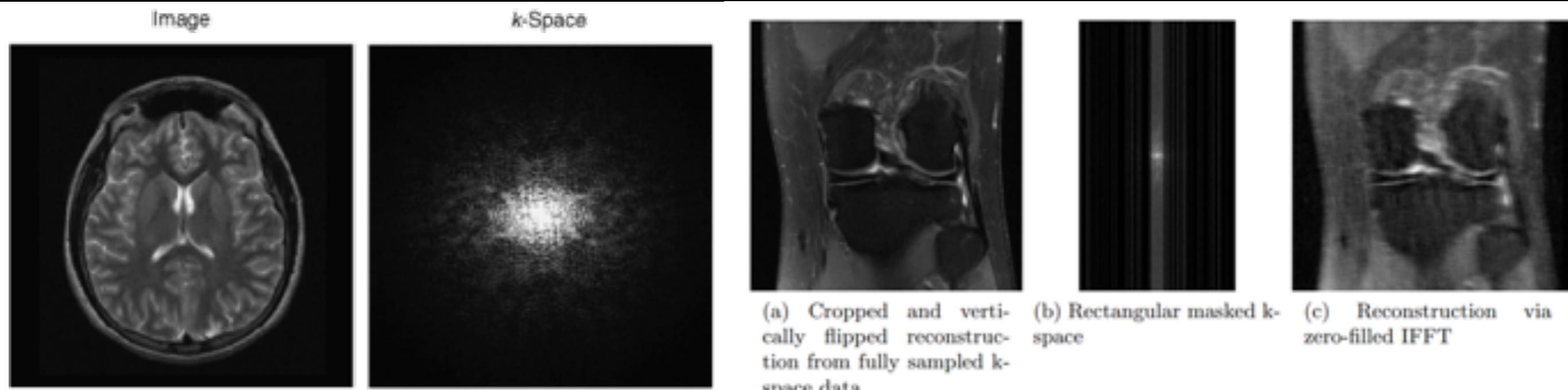


FIG. 5 Proposed deep convolutional neural network architecture for wavelet domain de-noising

Fast MRI reconstruction

- Reconstruct high-resolution image from undersampled k-space data.
- Undersampled k-space = faster MRI acquisition.



Fast MRI reconstruction

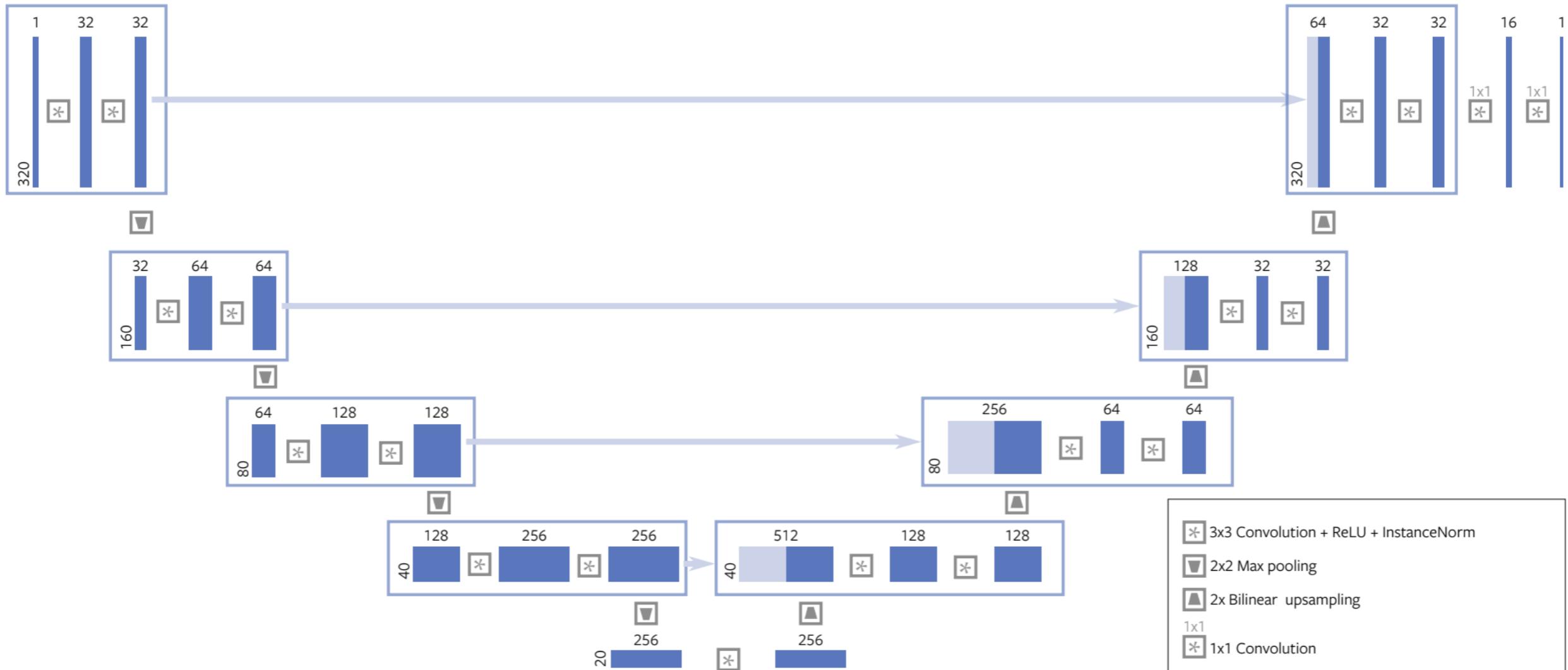
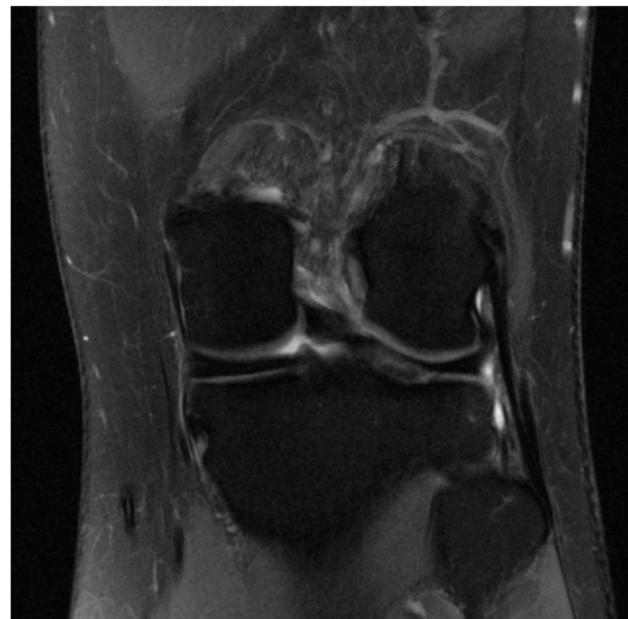
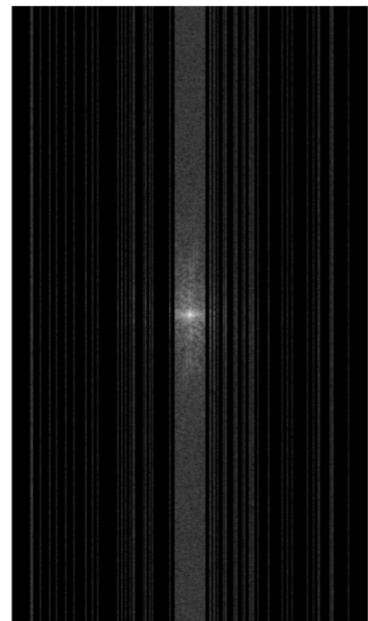


Figure 7: Single-coil baseline U-Net architecture

Fast MRI reconstruction



(a) Cropped and vertically flipped reconstruction from fully sampled k-space data



(b) Rectangular masked k-space



(c) Reconstruction via zero-filled IFFT



(d) Deep-learning baseline UNET reconstruction

Poisson denoising

- The Poisson noise model is relevant in low-light scenarios.
- Using a model trained on the specific class of the image can boost performance.



Ground truth image



Noisy image

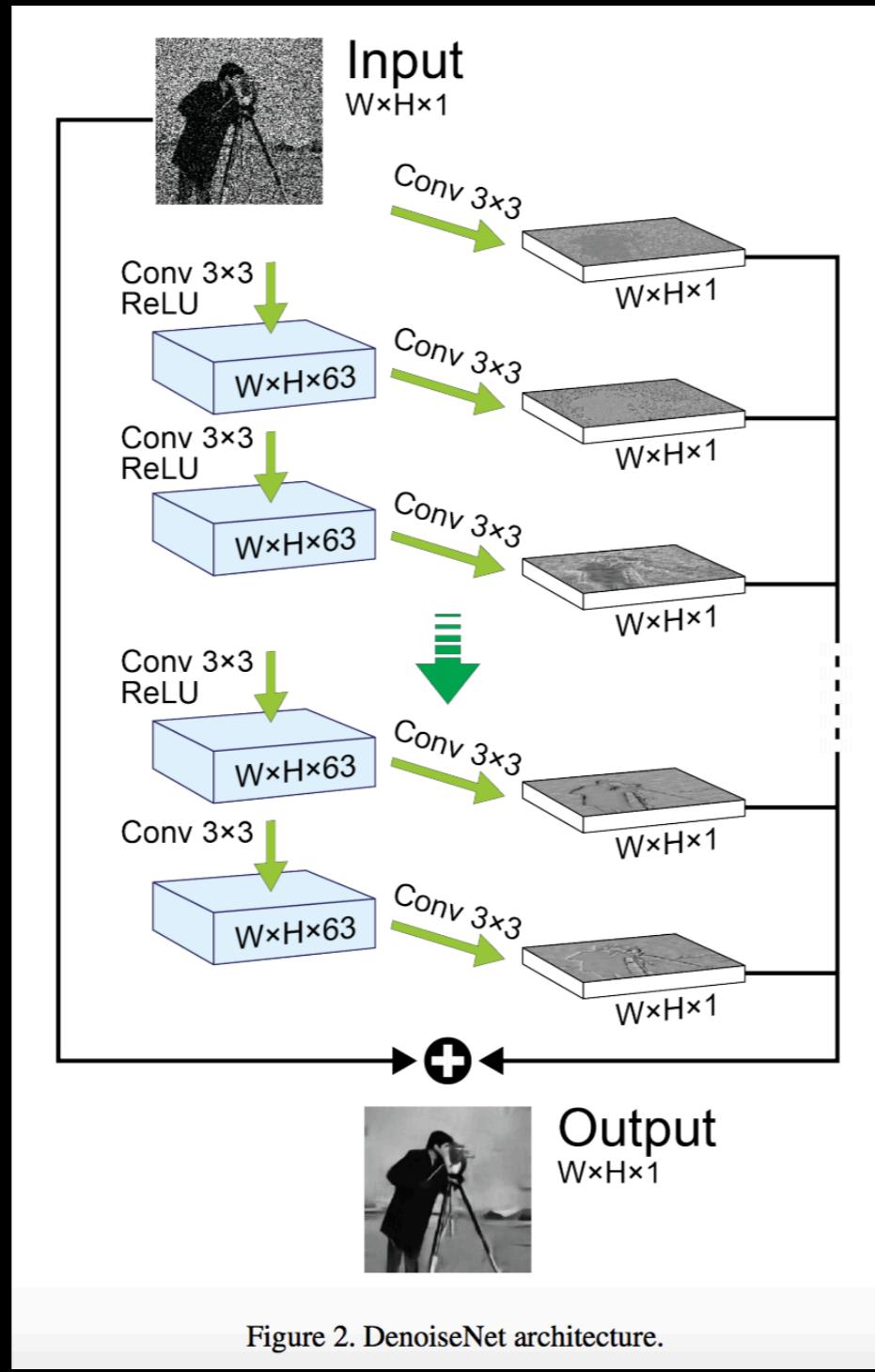


I+VST+BM3D [4]
24.45 dB

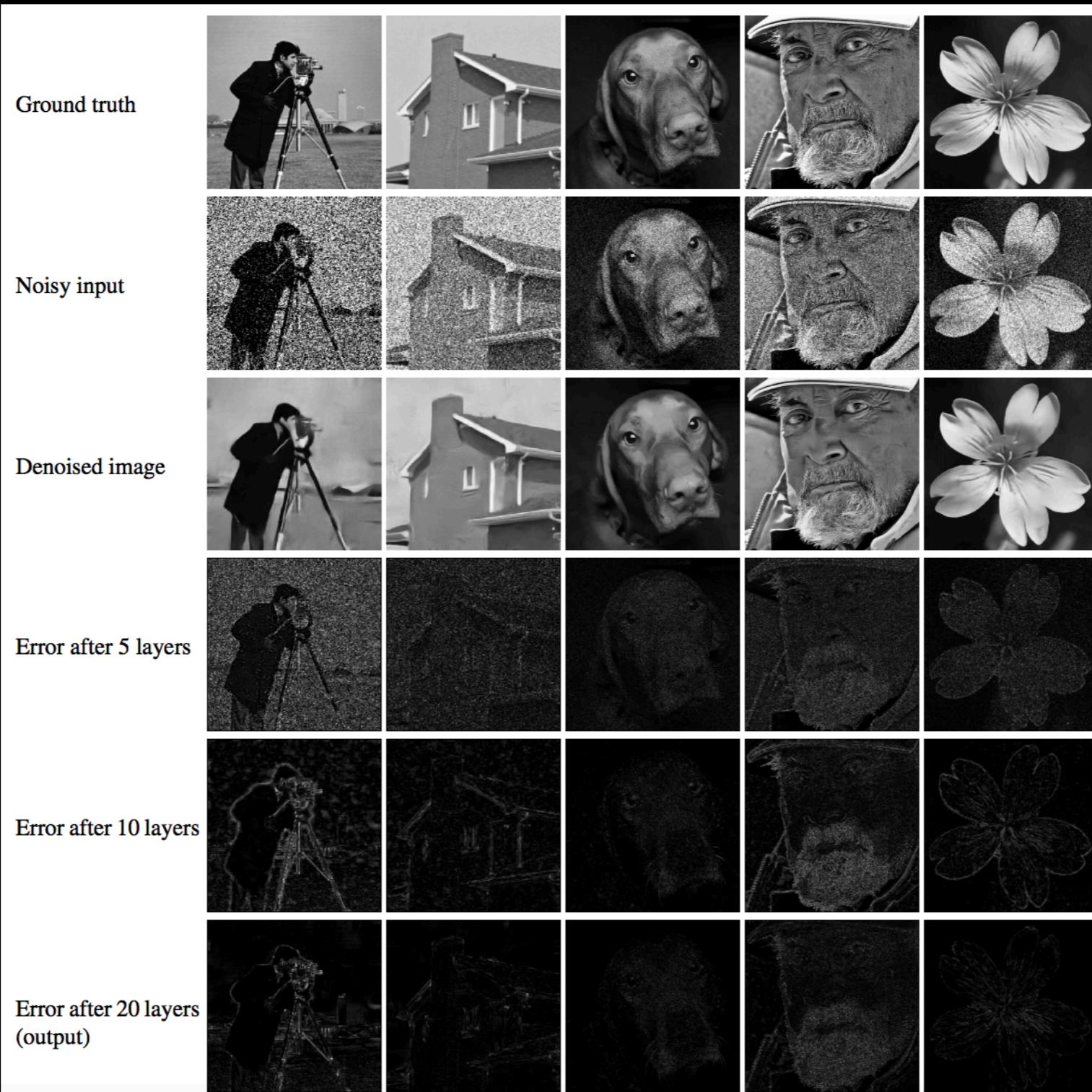


Proposed DenoiseNet
24.77 dB

Poisson denoising

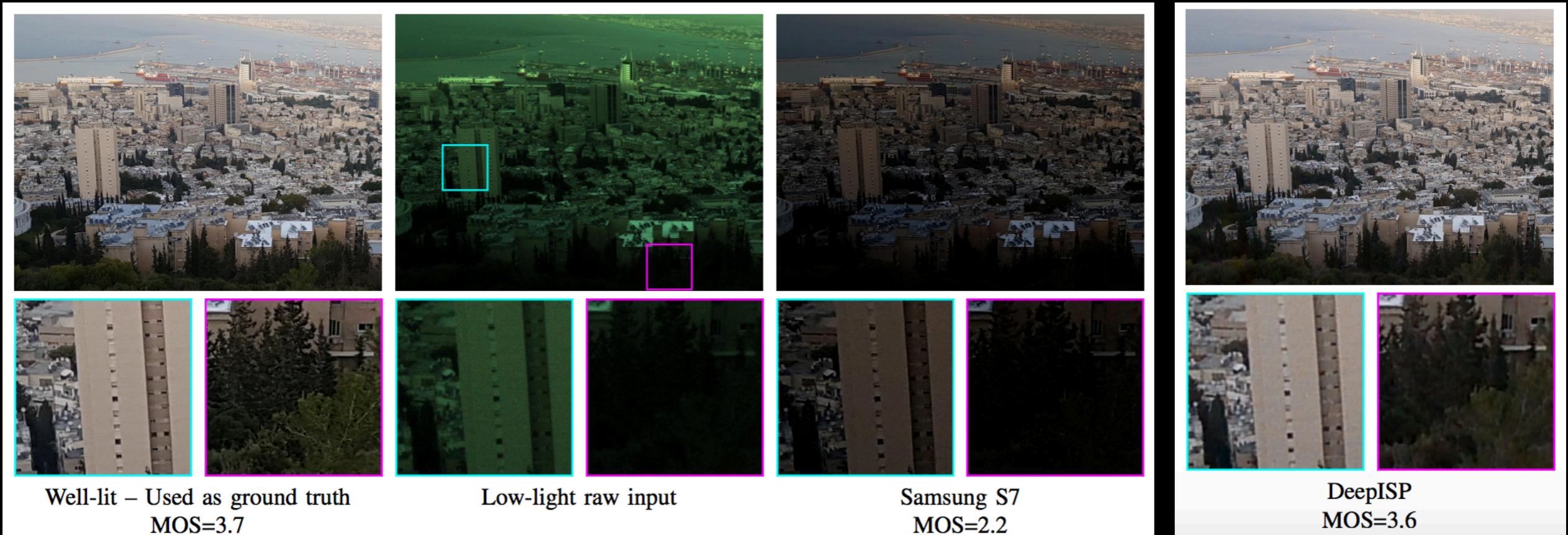


Poisson denoising



DeepISP

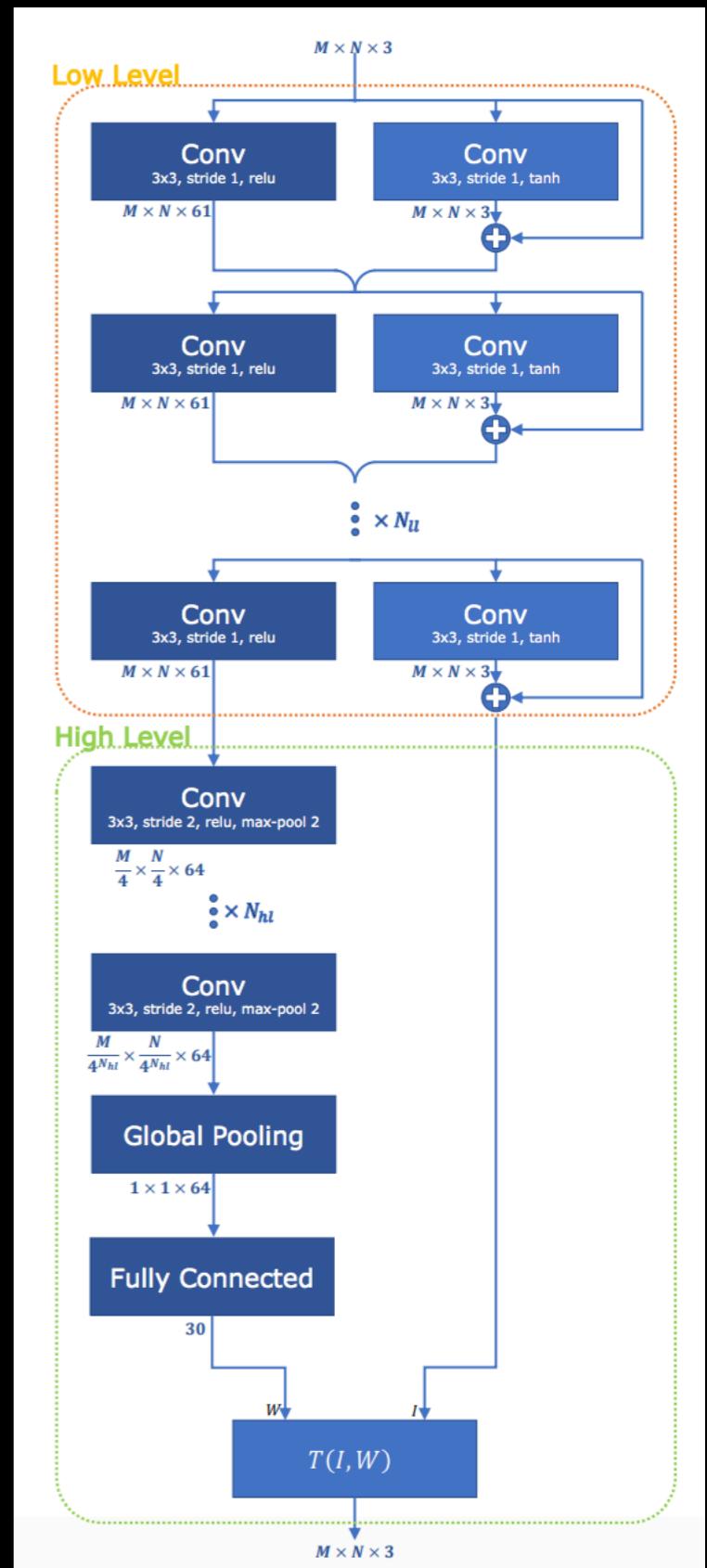
- Learn a mapping from a raw low-light mosaiced image to a final visually compelling image.



DeepISP

- Traditional ISP:
 - Low level: denoising, demosaicing.
 - High level: gamma correction, color correction.
- DeepISP:
 - Low level network
 - High level network

$$W \cdot \text{triu} \left([\begin{array}{cccc} r & g & b & 1 \end{array}]^T \cdot [\begin{array}{cccc} r & g & b & 1 \end{array}] \right)$$

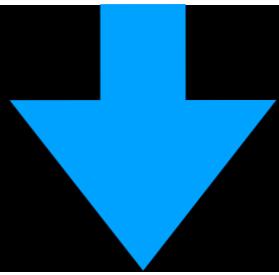


Deep burst deblurring

- Learn the weights for each Fourier coefficient in the burst.

$$u(\hat{\alpha}) = \mathcal{F}^{-1} \left(\sum_{i=1}^N w_i(\zeta) \hat{\alpha}_i(\zeta) \right) (x)$$

$$w_i(\zeta) = \frac{|\hat{\alpha}_i(\zeta)|^p}{\sum_{j=1}^N |\hat{\alpha}_j(\zeta)|^p},$$



$$u(\hat{\alpha}) = \mathcal{F}^{-1} \left(\sum_{i=1}^N h_\phi(\zeta) \hat{\alpha}_i(\zeta) \right) (x).$$

Deep burst deblurring

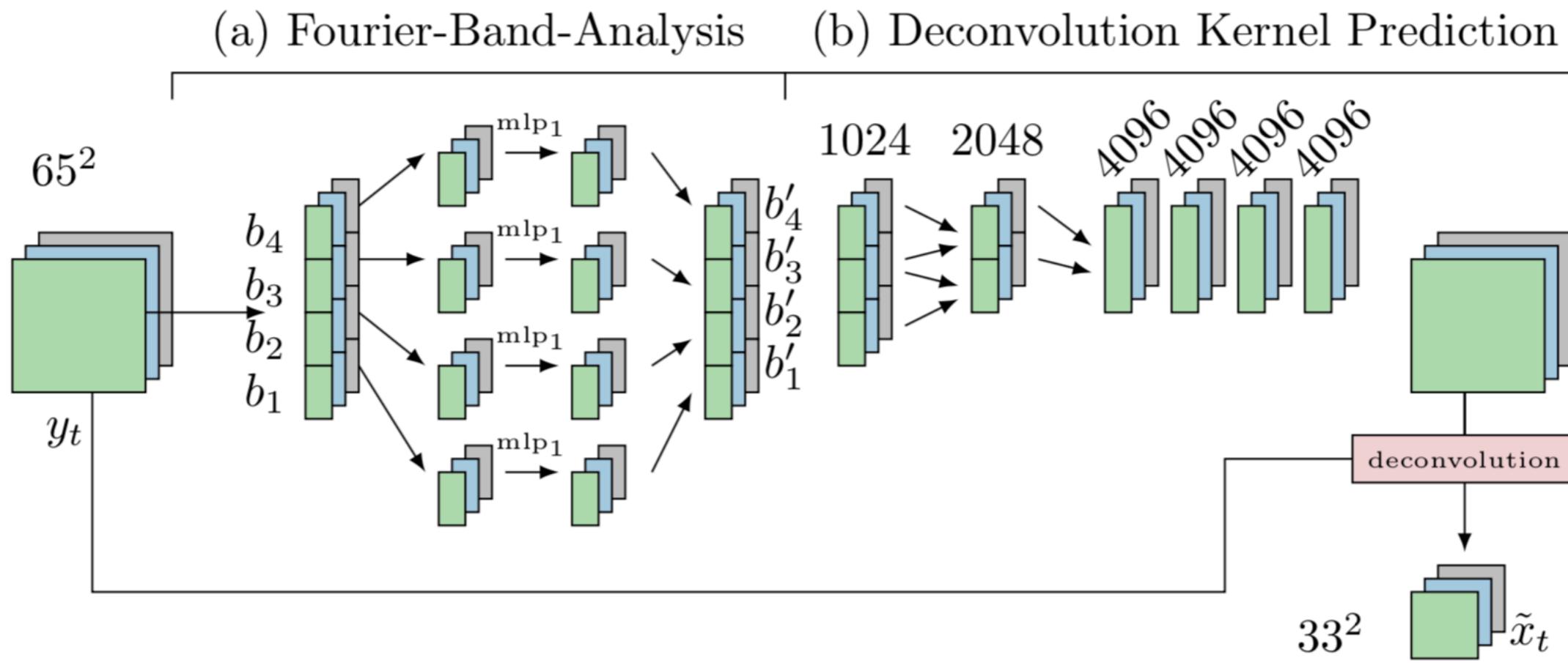


Fig. 1. Frequency band analysis and deconvolution for an image burst with 3 patches y_1, y_2, y_3 . Following the work of Chakrabarti [11] we separate the Fourier spectrum in 4 different bands b_1, \dots, b_4 . In addition, we allow each band separately to interact across all images in one burst to support early information sharing. The predicted output of the deconvolution step are smaller patches $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$.

Deep burst deblurring

