Deep Joint Demosaicking and Denoising

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Deep Learning for Noise Reduction

To support noise reduction or obtain original image from its noisy image (observed image), we can have **two approaches**:

(a) learning the denoising function F;

(b) learning the noise image (residual image) n.

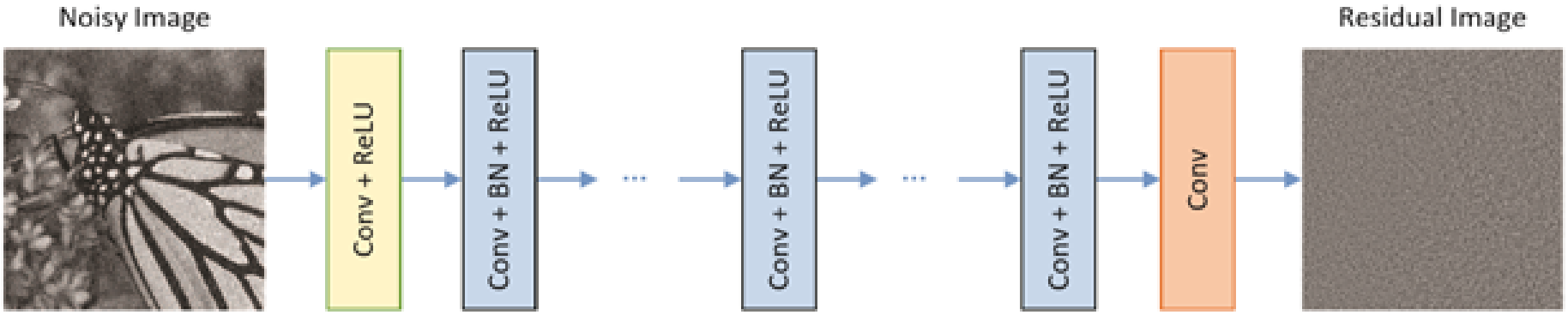
(1) **We can use deep network to learn the denoising function F, after F is available, we can generate the original image X from the noisy image Y as: X = F(Y)**

$$F = argmin_F \|F(Y) - X\|_2^2$$

Deep Learning for Noise Reduction

(2) We can use deep network to learn the noise image (residual image) n, after n is available, we can obtain the original image X from the noisy image Y as:

$$X = Y - n.$$



Deep Learning for Noise Reduction

The input of DnCNN is a noisy observation y = x + v. Discriminative denoising models such as MLP and CSF aim to learn a mapping function

$$\mathbf{F(y) = x}$$

to predict the latent clean image.

For DnCNN, we adopt the residual learning formulation to **train a residual mapping**

$$\mathbf{R(y) \approx v}$$

and then we have

$$\mathbf{x = y - R(y)}$$

Deep Learning for Noise Reduction

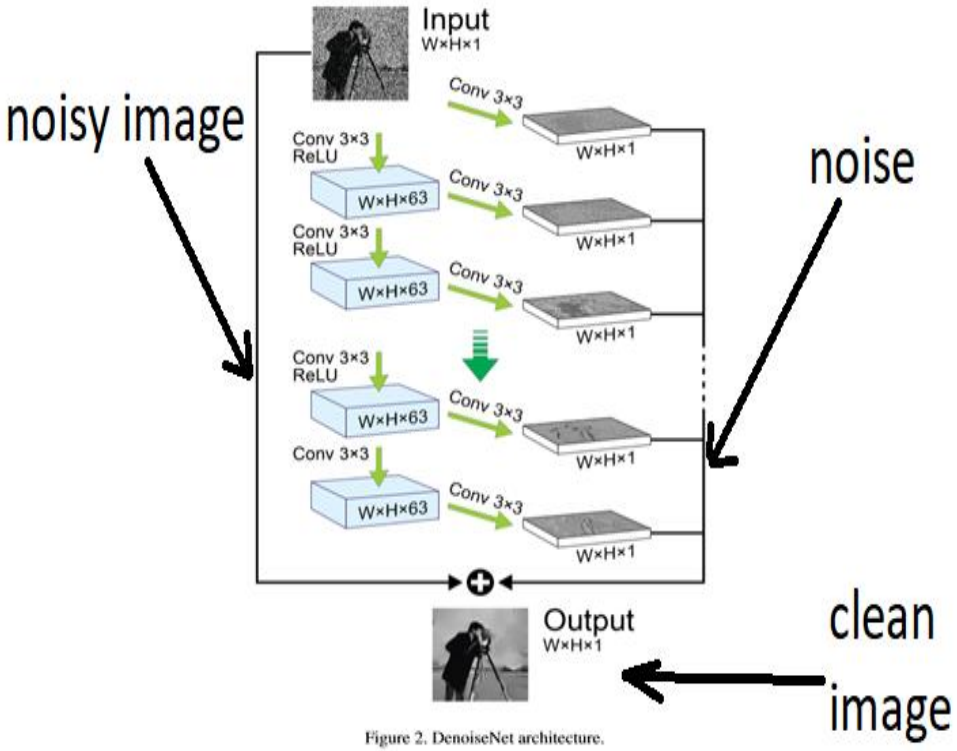


Figure 2. DenoiseNet architecture.

The objective function for learning the denoising function F is defined as:

$$F = argmin_F \|F(Y) - X\|_2^2$$

The objective function for learning the residual image n is defined as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{x}\|^2 + \lambda\Phi(\mathbf{x}), \qquad (1)$$

Deep Learning for Noise Reduction

The problem of denoising can be considered as a designing task in which an appropriate filter is constructed to remove the noise. This view aligns well with CNN in which a combination of filters in different layers is learned through real examples. The nonlinear property of the overall filter lends itself to complex distributions, which otherwise are hard to design.

The denoising function F depends on the noise level

Deep Learning for Noise Reduction

$$F = argmin_F \|F(Y) - X\|_2^2$$

The objective here is to determine the function F that creates the **closest image** to X from **noisy image** Y. The **denoising function** $\boldsymbol{F}$ can be considered as a nonlinear filter whose coefficients are not known but can be learned through training images. In this study, convolutional neural networks are used to learn such a filter. The denoising convolutional neural network is a modified CNN whose its parameters, e.g. weights and biases should be learned and calculated with the backpropagation training.

Deep Learning for Noise Reduction

The network for denoising consists of four layers. Each of these layers consists of a number of convolution filters followed by an activation layer.

The loss function is calculated at the end of every feed-forward step to update the parameters (weights and biases). The number of feature maps that are used from the first to the last layer is 64, 32, 32, and 1 respectively. The size of all convolutional filters is 9 × 9. The input of the network is the low-dose noisy image (Y), that passes through some convolutional layers to extract sets of feature maps to produce a denoised image (Y). Hence, a number of feature maps, which require going through the activation function, form the first convolutional layer. The subsequent convolutional layers receive the output of the previous layer then pass its results through activation functions for nonlinear processing to make higher-level feature maps in order to create a corresponding noise free image. As a result, the network trained hierarchically structured feature maps from low-level (blobs, edges, etc.) to high-level (more complex and detailed shapes). Moreover, we came up with a pre-train convolutional network, which can be used as a transfer learning.

De-Mosaicking

Demosaicking and denoising are among the most crucial steps of modern digital camera pipelines and their joint treatment is a highly ill-posed inverse problem where at-least two-thirds of the information are missing and the rest are corrupted by noise. This poses a great challenge in obtaining meaningful reconstructions and a special care for the efficient treatment of the problem is required. While there are several machine learning approaches that have been recently introduced to deal with joint image demosaicking-denoising.

De-Mosaicking

Due to the modular nature of the camera processing pipelines, demosaicking and denoising were traditionally dealt in the past in a sequential manner. In detail, demosaicking algorithms reconstruct the image from unreliable spatially-shifted sensor data which introduce non-linear pixel noise, casting denoising an even harder problem.

To solve the joint demosaicking-denoising problem, one of the most frequently used approaches in the literature relies on the following linear observation model

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{n}, \qquad (1)$$

De-Mosaicking

Recovering x from the measurements y belongs to the broad class of **linear inverse problems**. For the problem under study, the operator M is clearly singular. This fact combined with the presence of noise perturbing the measurements leads to an ill-posed problem where a unique solution does not exist. One popular way to deal with this, is to adopt a Bayesian approach and seek for the Maximum A Posteriori (MAP) estimator

$$\mathbf{x}^{\star} = \arg\max_{\mathbf{x}} \log(p(\mathbf{x}|\mathbf{y})) = \arg\max_{\mathbf{x}} \log(p(\mathbf{y}|\mathbf{x})) + \log(p(\mathbf{x})), \qquad (2)$$

$$\mathbf{x}^{\star} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 + \phi(\mathbf{x}) \qquad (3)$$

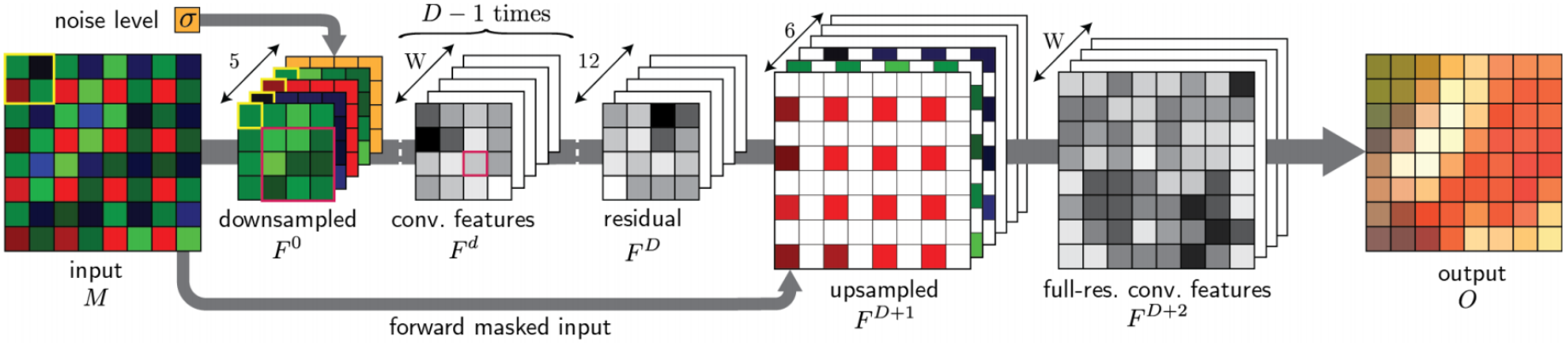
Demosaicking is further complicated by the presence of noise. Estimates of edge orientation in noisy data are less reliable which leads to noticeable artifacts in the demosaicked image. The techniques that perform these steps sequentially usually start with denoising. Recent attempts have shown the advantages of joint approaches.

Jeon and Dubois [2013] optimize a set of filters for discrete noise levels. Heide et al [2014] use a global primal-dual optimization with a self-similarity prior. The nearest neighbor search and the iterative nature of the algorithm makes it slow and somewhat impractical.

Demosaicking and denoising have traditionally been addressed using nonlinear filter design, incorporating prior heuristics about interand intra-channel correlation, behavior around edges, and exploiting intra-image patch similarity.

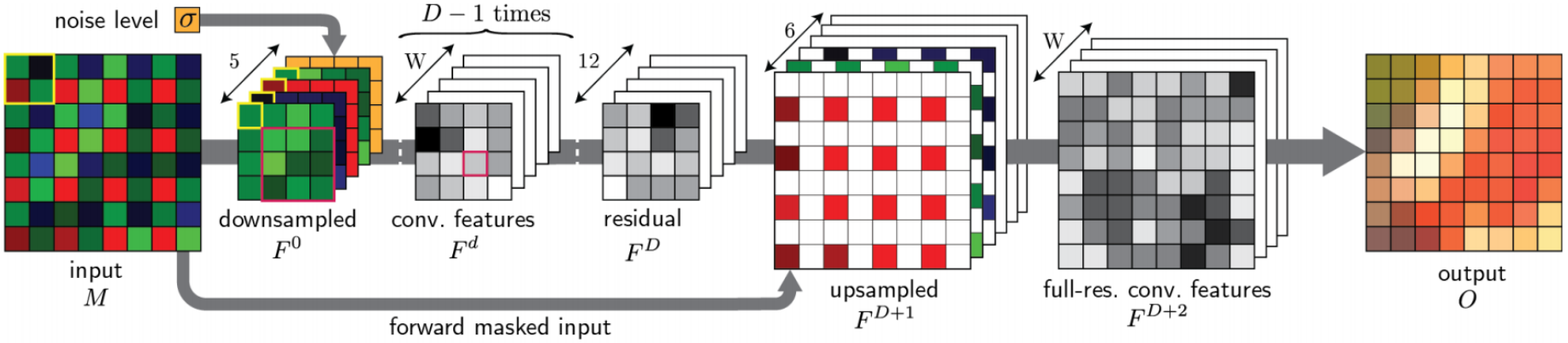
A convolutional network seems a natural choice for the problem in this context. First, it enables discovery of natural correlations in the data. Second, the network can represent a superset of the pipelines implemented by many previous techniques while all its parameters are optimized jointly to minimize a single objective.

We cast joint denoising and demosaicking as a supervised learning problem: we train our algorithm on a set of input measurements for which the desired output is known. We create the training set from millions of sRGB images, generating the corresponding mosaicked arrays by leaving out two color channels per pixel and adding noise. We then build a convolutional neural network and train it in an end-to-end fashion. The inputs are the mosaicked array M with a single channel per pixel and an estimate σ of the noise level; the output is an image O of the same size with a RGB triplet per pixel. We start our exposition focusing on demosaicking and then discuss noise.



The first layer of the network packs 2 × 2 blocks in the Bayer image into a 4D vector to restore translation invariance and speed up the processing. We augment each vector with the noise parameter σ to form 5D vectors.

A series of convolutional layers filter the image to interpolate the missing color values. We finally unpack the 12 color samples back to the original pixel grid and concatenate a masked copy of the input mosaick. We linearly combine them to produce the demosaicked output.



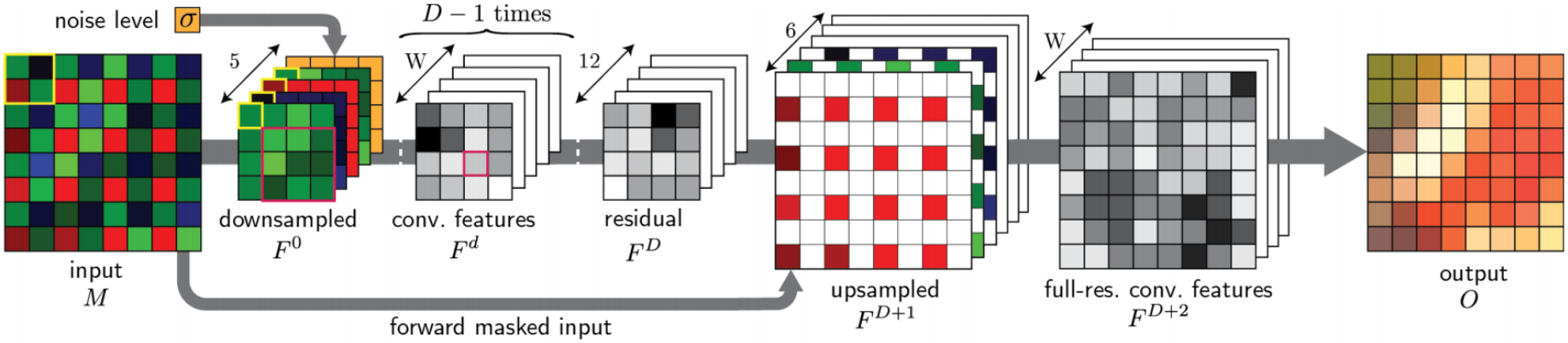
The first layer F^0 extracts 2×2 patches from M and packs them as a 4 channel feature map indexed by c.

$$F_c^0(x,y) = \mathrm{M}\left(2x + (c \bmod 2),\ 2y + \left\lfloor \frac{c}{2} \right\rfloor\right) \qquad (1)$$

$$F_c^d = f\left(b_c^d + \sum_{c'=1}^{W} w_{cc'}^d * F_{c'}^{d-1}\right) \text{ for } c \in \{1 \ldots W\} \qquad (2)$$

$$F_c^{D+1}(x,y) = m_c(x,y)\mathrm{M}(x,y) \text{ for } c \in \{1 \ldots 3\} \qquad (3)$$

$$F_c^{D+1}(x,y) = F_{c'}^D\left(\left\lfloor \frac{x}{2} \right\rfloor, \left\lfloor \frac{y}{2} \right\rfloor\right) \text{ for } c \in \{4 \ldots 6\} \qquad (4)$$



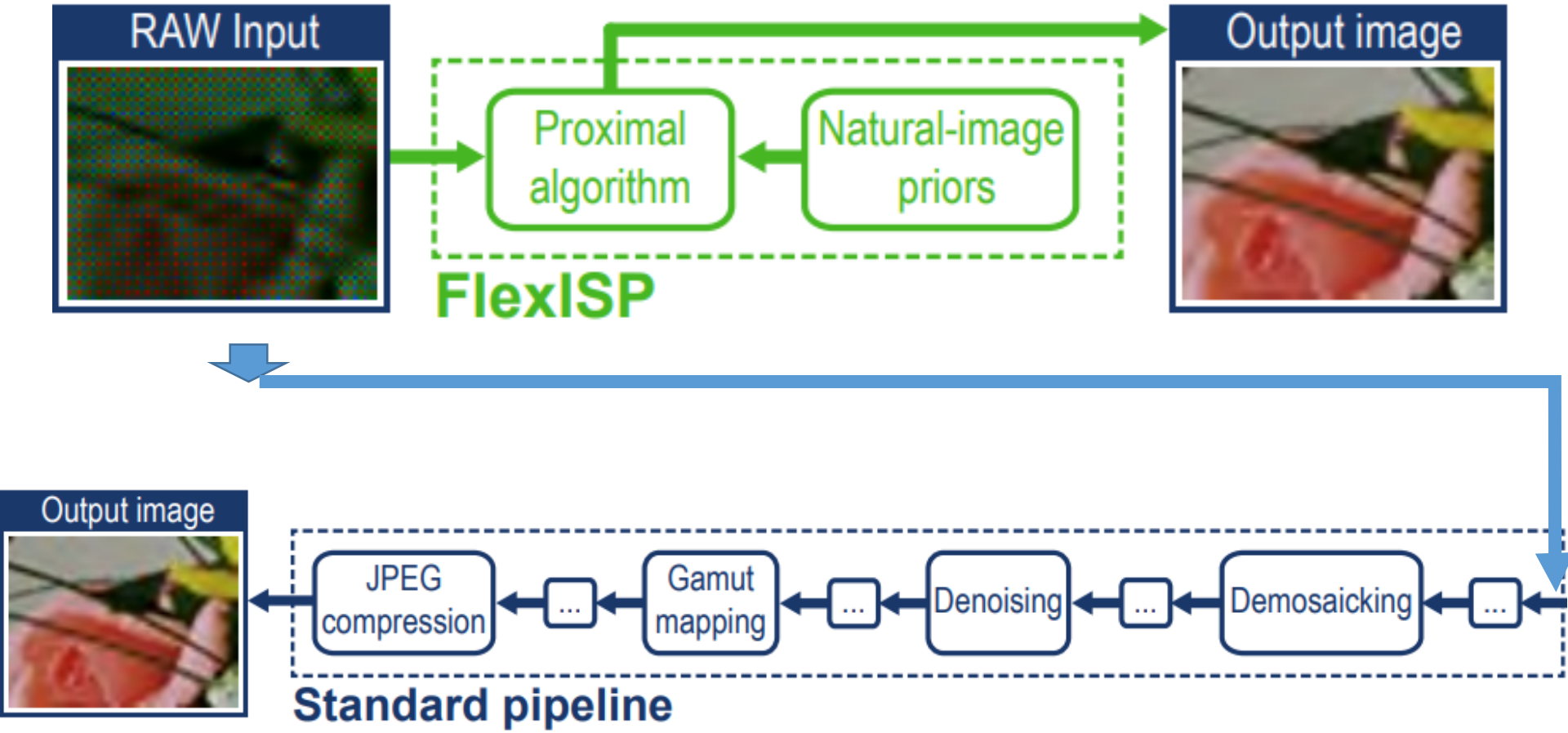
At training time, we use a dataset $\mathcal{D} = \{(\sigma_i, \mathrm{M}_i, \mathrm{I}_i)\}_i$ of mosaicked/ground-truth image patches where M_i is generated from I_i and corrupted with additive white Gaussian noise of variance σ_i^2 on-line. We optimize the weights and biases by minimizing the normalized L_2 loss on this training set:

$$\mathcal{L}\left(\left\{w^{(d)}, b^{(d)}\right\}_d\right) = \frac{1}{p^2|\mathcal{D}|}\sum_i ||\mathrm{O}_i - \mathrm{I}_i||^2$$


reference
FlexISP
32.5 dB
Ours
38.4 dB
noisy
$\sigma = 12$
ref.
ours
33.3 dB

Conventional pipelines for capturing, displaying, and storing images are usually defined as a series of cascaded modules, each responsible for addressing a particular problem. While this divide-and-conquer approach offers many benefits, it also introduces a cumulative error, as each step in the pipeline only considers the output of the previous step, not the original sensor data.

An end-to-end solution is to jointly consider the steps like demosaicking, denoising, deconvolution, and so forth, all directly in a given output representation (e.g., YUV, DCT).



Addressing subproblems separately does not yield the best-quality reconstructions, especially for complex image formation models. Recently, a number of researchers have identified this problem, and proposed joint solutions to several subproblems

Due to the linear nature of these optical processes, the image transformation can be expressed as a matrix B operating on the image vector. We subsume these transformations in a matrix A, and express our observation model as

$$\mathbf{z} = \mathbf{A}\mathbf{x} + \eta, \qquad (1)$$

Our goal is to find the underlying latent image x from the (usually) sparse and noisy observations z. Given the Gaussian noise model, this can be achieved with a standard l2 minimization.

$$\min_{\mathbf{x}} \underbrace{\frac{1}{2}\|\mathbf{z} - \mathbf{A}\mathbf{x}\|_2^2}_{\text{data fidelity}} + \underbrace{\Gamma(\mathbf{x})}_{\text{regularization}}. \qquad (2)$$