Tutorial-10: Dictionary learning, Deep Image Prior

Sanketh Vedula
sanketh@cs.technion.ac.il

Agenda

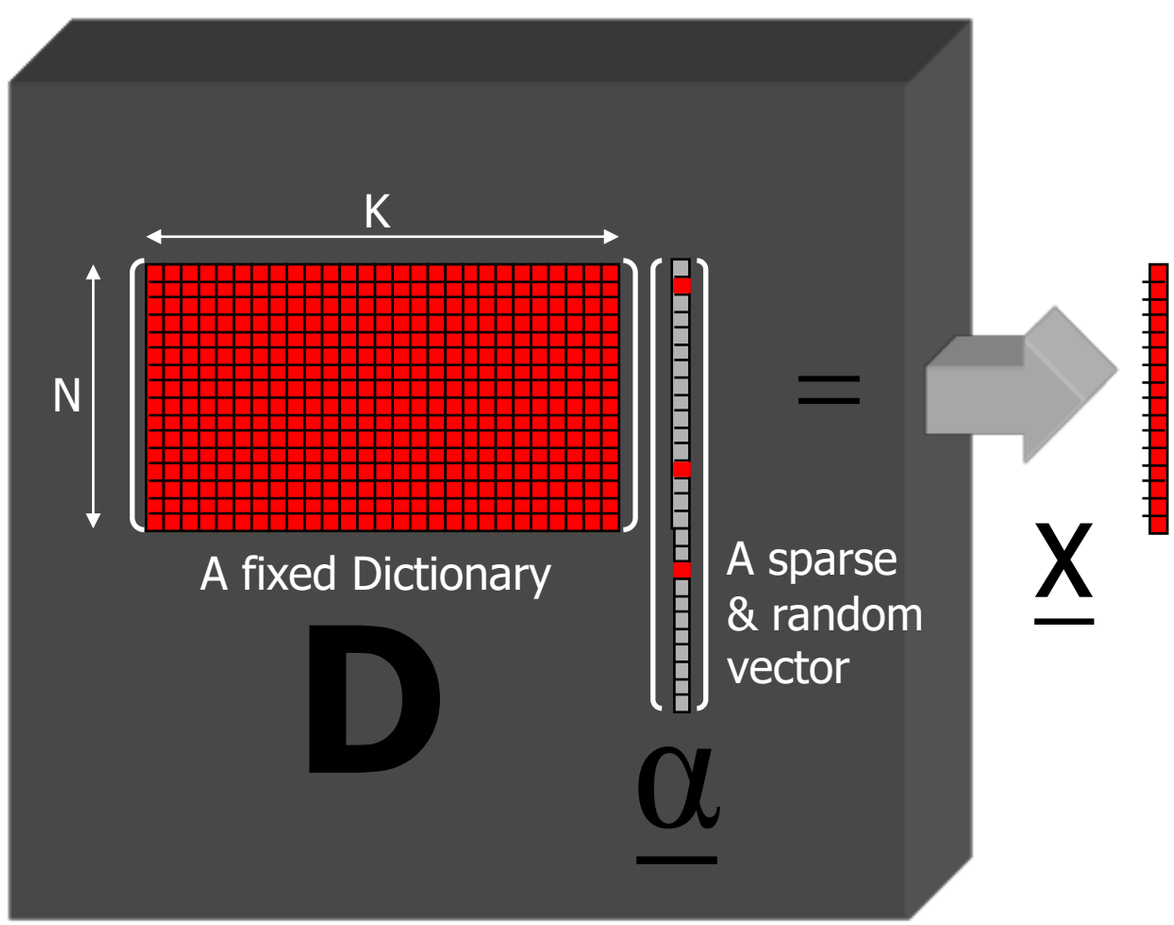
- Dictionary learning: K-SVD
  - Problem setup
  - Sparse coding
  - Dictionary update
  - Implementation
- Deep Image Prior
  - Standard image priors
  - Parametrised image priors
  - Deep Image Prior: training strategy
  - Results

Sparseland signal synthesis

$\mathbf{x} \in \mathbb{R}^N$ signal

$\mathbf{D} \in \mathbb{R}^{N \times K}$ dictionary

$\alpha \in \mathbb{R}^K$ representation



- Every column in **D** (dictionary) is a prototype signal (Atom).
- The vector α is generated randomly with few non-zeros in random locations and random values.

Elad, 2005

Sparse coding

$$\mathbf{x} \in \mathbb{R}^N \qquad \mathbf{D} \in \mathbb{R}^{N \times K} \qquad \alpha \in \mathbb{R}^K$$

signal dictionary representation

$$(P_0^\epsilon) \ : \ \min ||\alpha||_0, \text{ such that } \ ||\mathbf{x} - \mathbf{D}\alpha||^2 < \epsilon$$

NP-hard!

$$(P_0^\lambda) \ : \ \arg\min_{\mathbf{x}} \ ||\mathbf{x} - \mathbf{D}\alpha||^2 + \lambda||\alpha||_0$$

Approximate solution: Orthogonal Matching Pursuit

Greedily pick atoms that minimise the residual error

RECAP

Sparse coding

$$(P_0^{\lambda}) \quad : \quad \arg\min_{\mathbf{x}} \; ||\mathbf{x} - \mathbf{D}\alpha||^2 + \lambda||\alpha||_0$$

Orthogonal Matching Pursuit:

init: support = { }, residual (r) = x
Until reaching the desired level of sparsity:

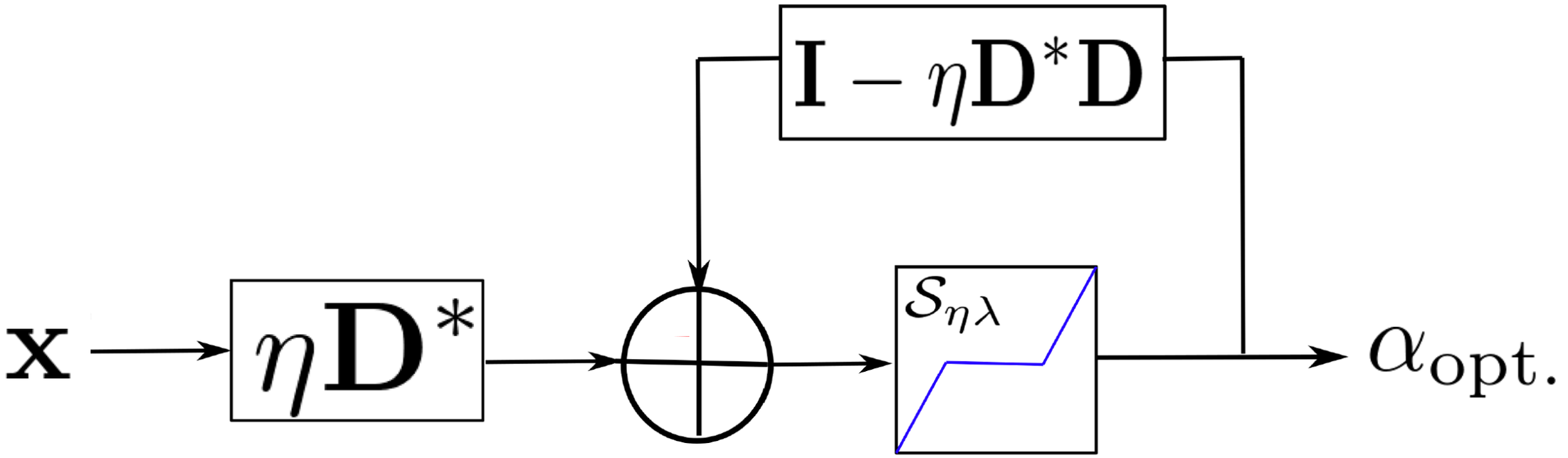
- Choose an unselected atom/column from D that minimises || r - D alpha ||
- Add the chosen atom (d_i) into the support => support = support + {d_i}
- Update the residual (r) => r = || x - D alpha{d_i} ||
- Repeat until reaching desired level of sparsity or when r is close to 0

More on this: Elad, 2009

Approximate Sparse coding

$$(P_1^{\lambda}) \;:\; \arg\min_{\mathbf{x}} \; ||\mathbf{x} - \mathbf{D}\alpha||^2 + \lambda||\alpha||_1$$

Solve this? ISTA, Basis Pursuit



Where do dictionaries come from?

Modeling?

Carefully incorporate the information we know about the data within these dictionaries:

For example:
If we are working with natural images, then D could be over-complete DCT basis, Wavelet basis, etc.

Pros:
Fast transforms
Proven optimality

Cons:
How do you adapt to different signals?
Not all dictionaries would allow sparsity?
Maybe optimal for large signal families — but are not “signal-specific”.

Elad, 2005

Where do dictionaries come from?

Learning?

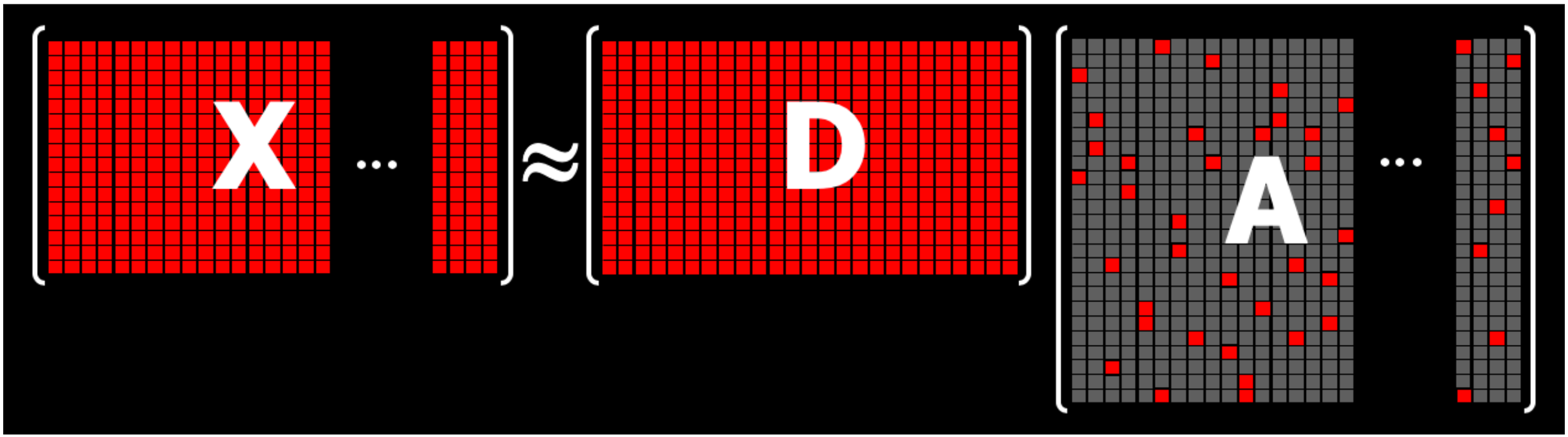
Train the dictionary directly on the given examples, optimizing w.r.to. sparsity and other desired properties (normalized atoms etc.)

Pros:
Fits the data at hand
Signal-specific or “signal family” specific modeling

Cons:
No fast transforms anymore
Training takes time
Not scalable?

Elad, 2005

Dictionary Learning

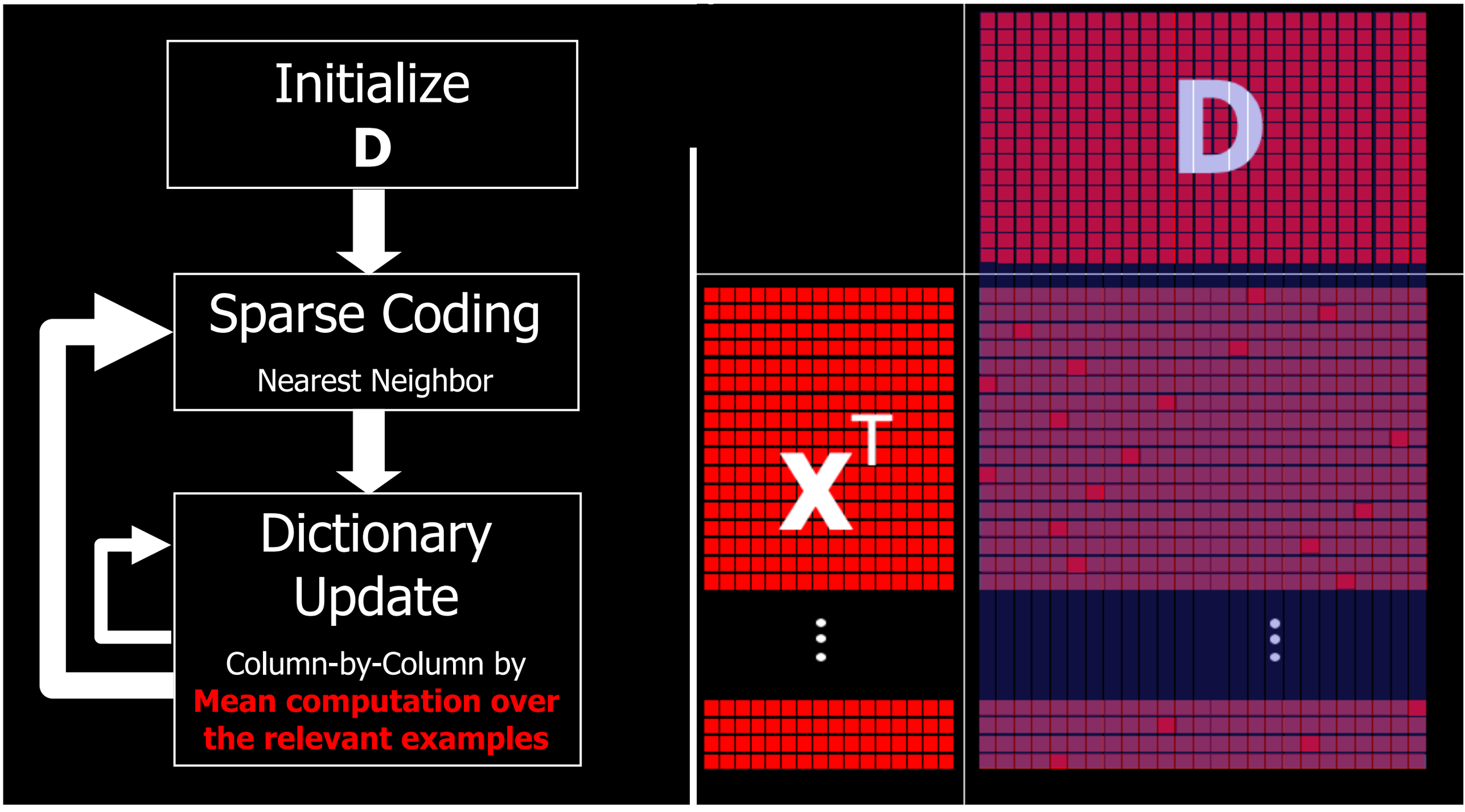


$$\underset{\mathbf{D},\mathbf{A}}{\text{Min}} \sum_{j=1}^{P} \left\| \mathbf{D}\underline{\alpha}_j - \underline{x}_j \right\|_2^2 \quad \text{s.t. } \forall j, \left\| \underline{\alpha}_j \right\|_0 \leq L$$

Objective: Update dictionary and representations such that the representation error is minimal while still maintaining sparsity.

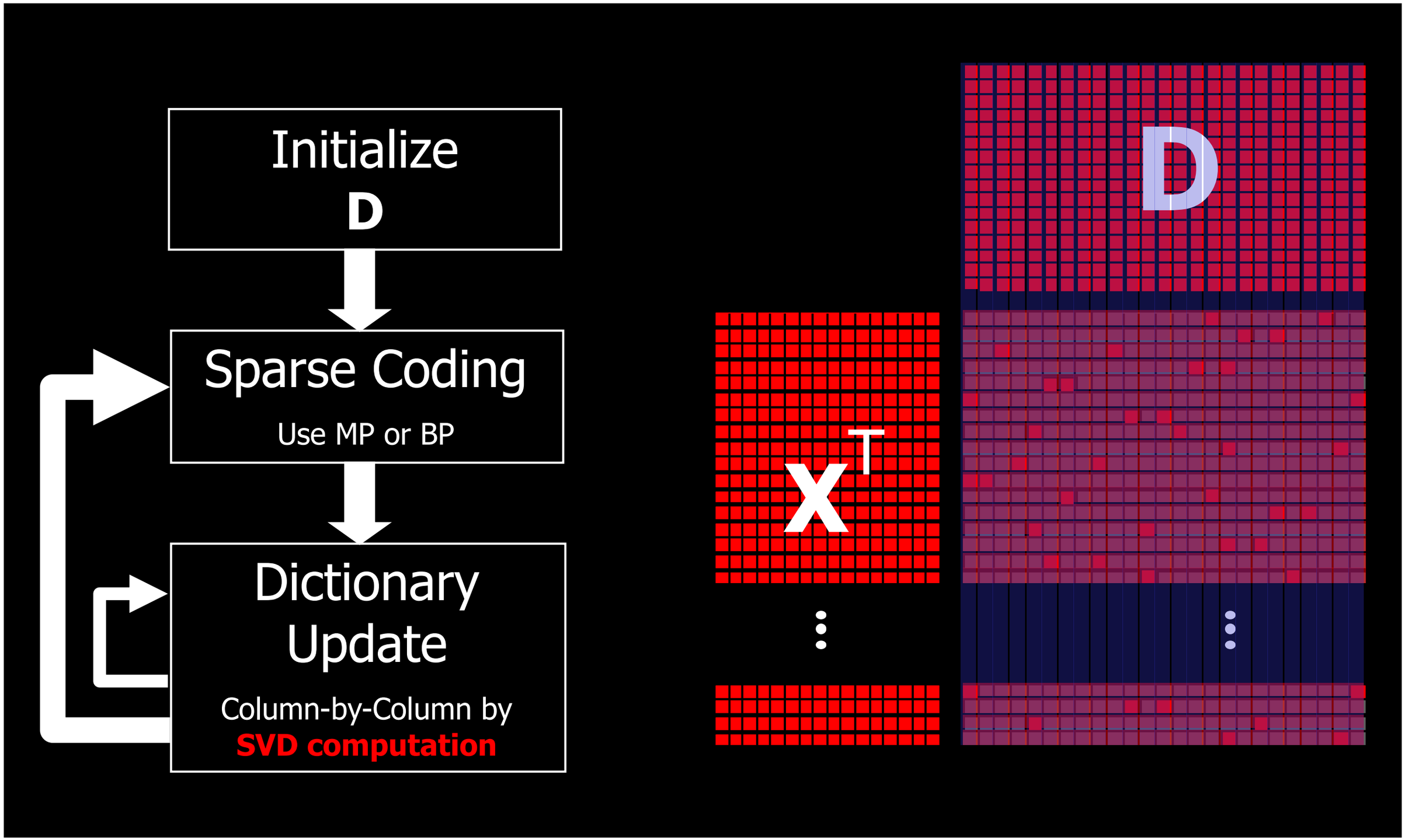
Elad, 2005

K-Means



Elad, 2005

K-SVD

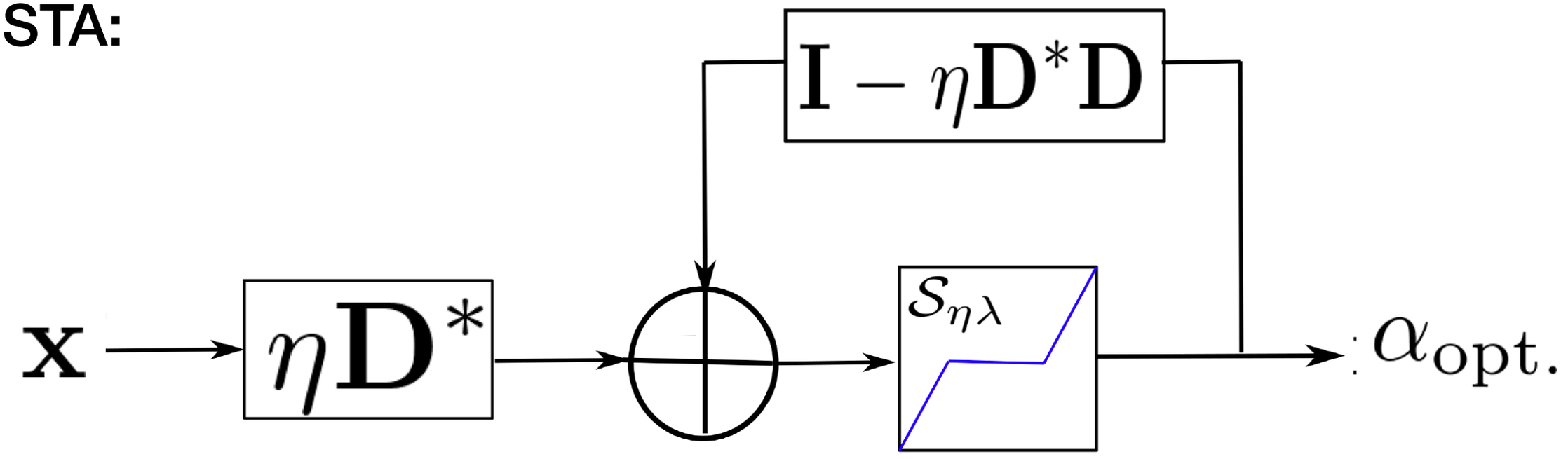

Initialize
D
Sparse Coding
Use MP or BP
Dictionary
Update
Column-by-Column by
SVD computation
D
X^T

K-SVD: Sparse coding

Goal: We know D, we need to get the sparse code.

Pursuit methods: OMP, Basis Pursuit

ISTA:



K-SVD: dictionary update

$$||\mathbf{X}-\mathbf{DA}||_F^2 = \left|\left|\mathbf{X}-\sum_{j=1}^{K} d_j\alpha_j\right|\right|_F^2 = \left|\left|\left(\mathbf{X}-\sum_{j\neq k} d_j\alpha_j\right)-d_k\alpha_k\right|\right|_F^2 = ||\mathbf{E}_k - d_k\alpha_k||_F^2$$

SVD?

We want to update kth column in D.
How to do it?

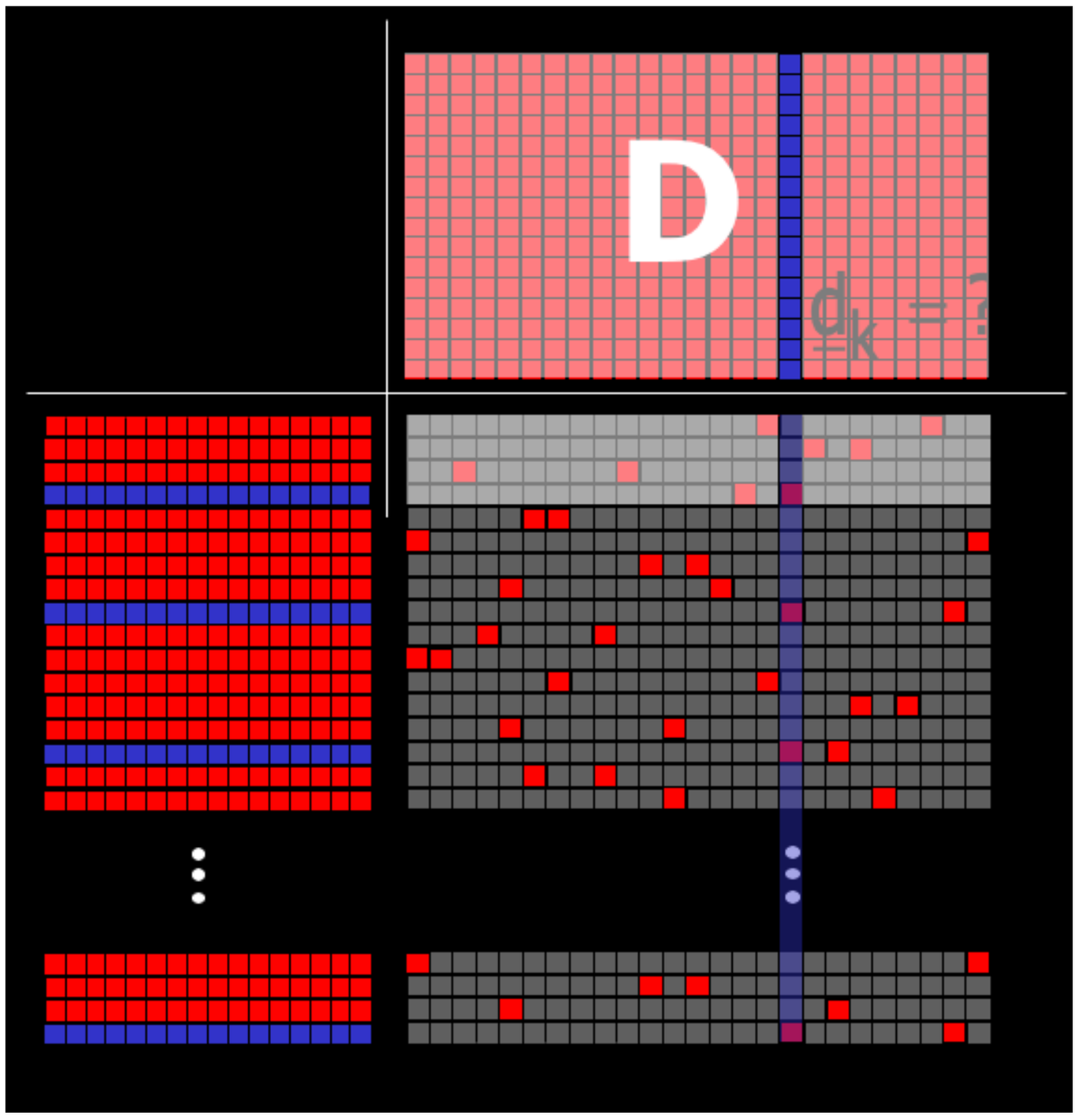
Possible solution: For each column in D,
let us treat the update step as a matrix factorisation problem — just do singular value decomposition and update the D and A.

Caveat?

This might be destroying the sparsity constraint.
Because, there's good chance that the resulting sparse code is not sparse.

Elad, 2005

K-SVD: Dictionary update



Do the following steps for each dictionary column d_k:

↓

Find the signals that use d_k

↓

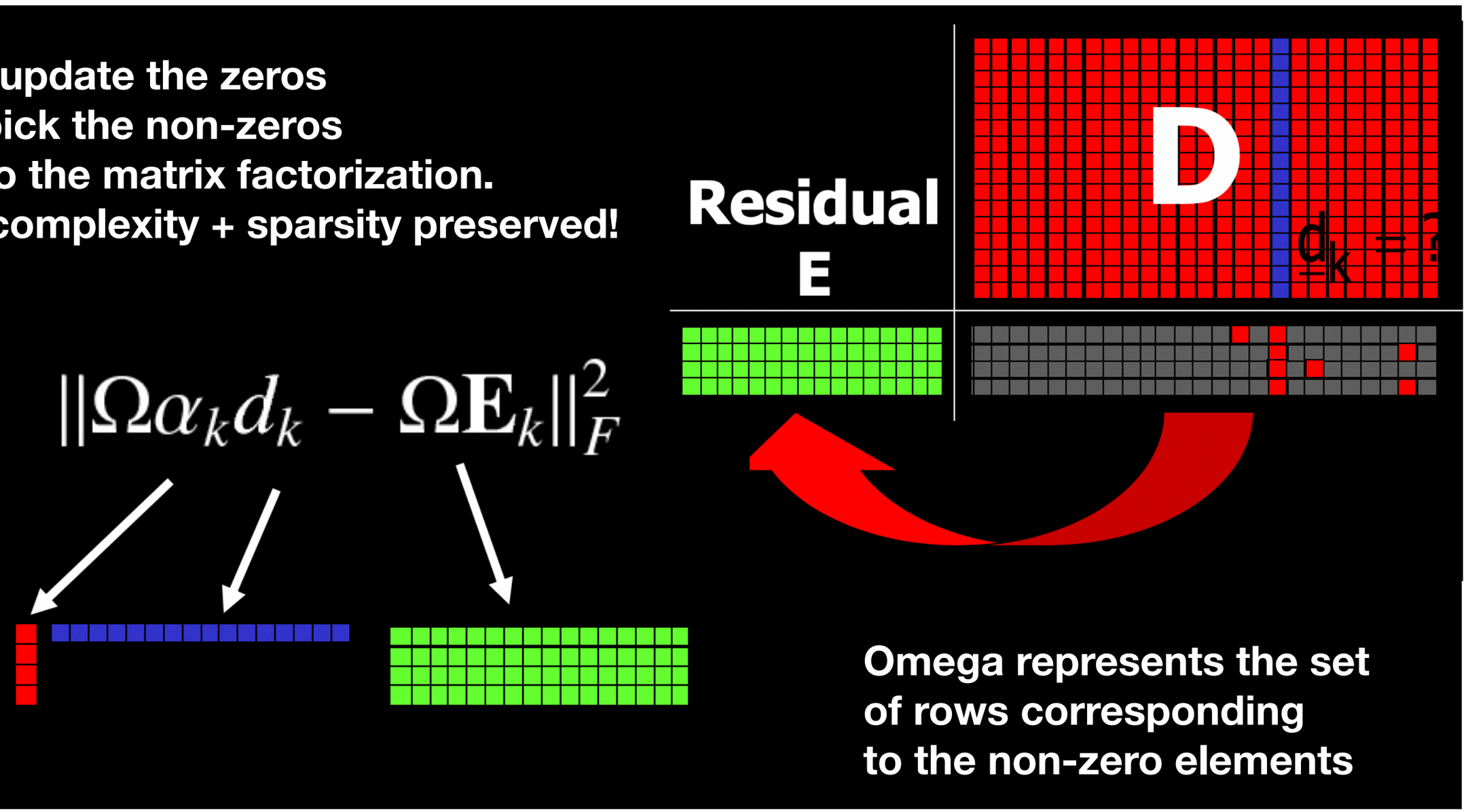
Fix the rest of the A and D apart from the kth column and seek both d_k and kth column in A to better fit the residual

$$\|\Omega\alpha_k d_k - \Omega \mathbf{E}_k\|_F^2$$

Elad, 2005

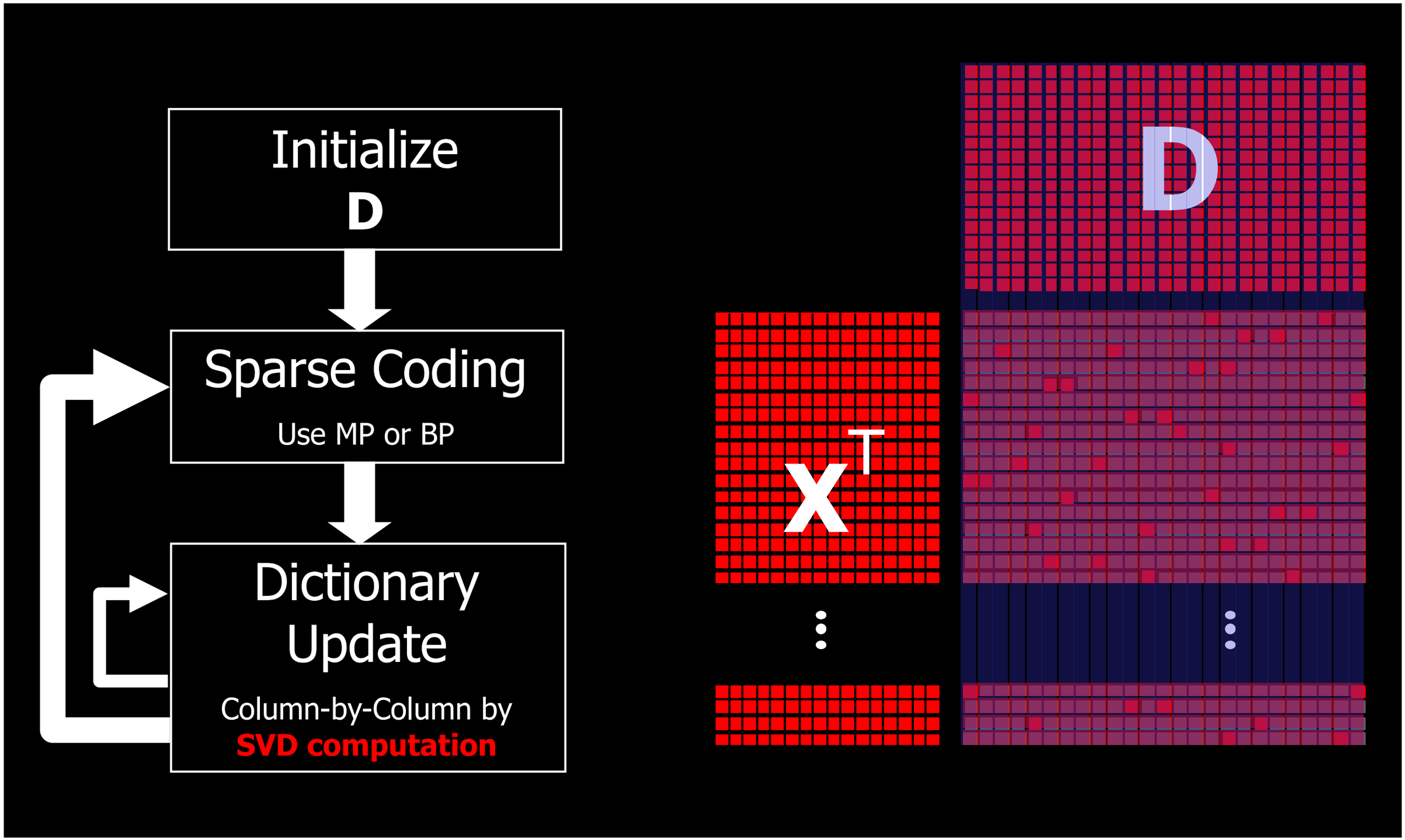
K-SVD: Dictionary update

Don’t update the zeros
Just pick the non-zeros
and do the matrix factorization.
Less complexity + sparsity preserved!



Elad, 2005

K-SVD - summarised

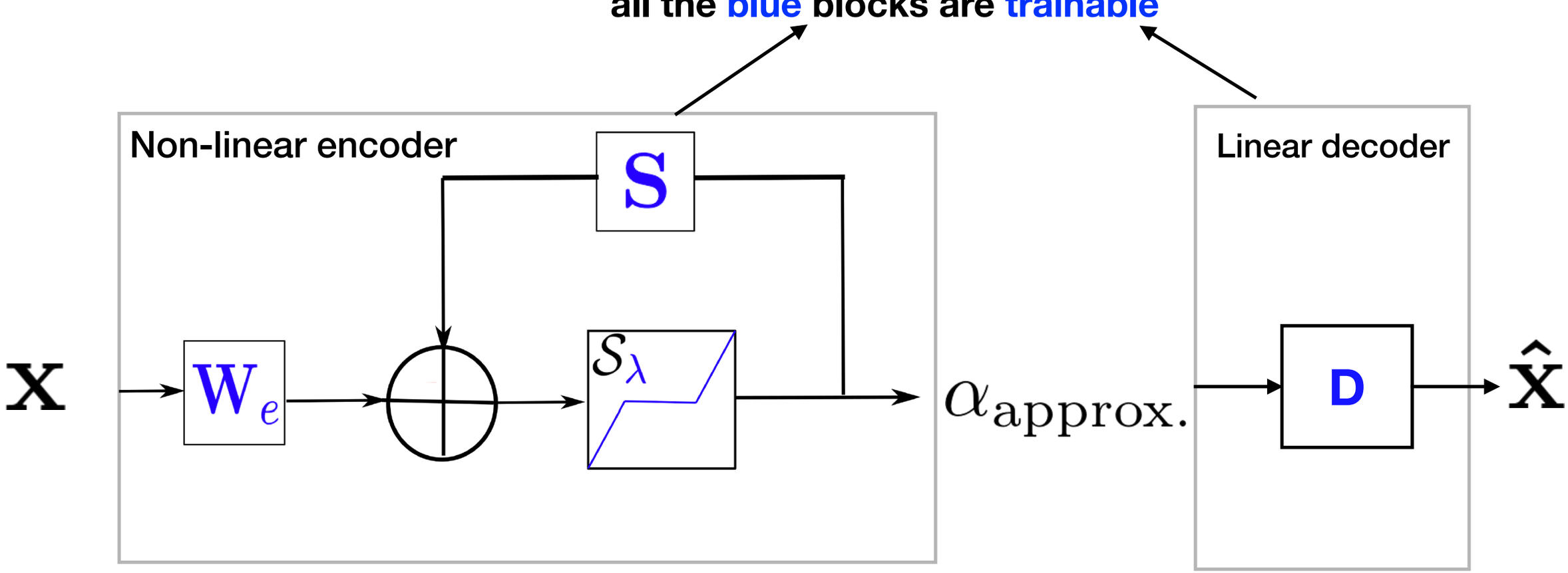


Unrolled networks

Denoising:
x - noisy input
x_hat - denoised

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = ||\hat{\mathbf{x}} - \hat{\mathbf{x}}||^2$$

optimized with gradient descent
and backpropagation



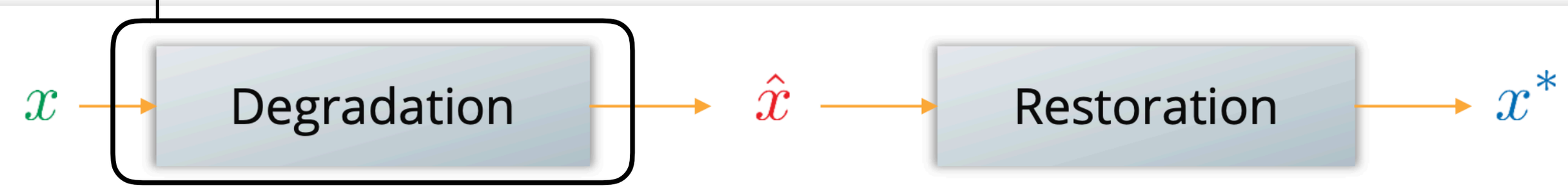
Notice the resemblance with neural networks — if the proximal operator
is non-negative and if it has a non-linear decoder

Ulyanov et al, 2018

Image Priors

- x – Clean image
- $\hat{x}$ – Corrupted image (observed)
- x^* – Restored image

denoising,
inpainting, blur etc.



MAP: $$x^* = \arg\max_x p(x|\hat{x})$$

$$p(x|\hat{x}) = \frac{p(\hat{x}|x)p(x)}{p(\hat{x})} \propto p(\hat{x}|x)p(x)$$

Likelihood **Prior**

Ulyanov et al, 2018

Image Priors

- x – Clean image
- $\hat{x}$ – Corrupted image (observed)
- x^* – Restored image

$$
\begin{aligned}
x^* &= \arg\max_x p(\hat{x}|x)p(x) \\
&= \arg\min_x -\log p(\hat{x}|x) - \log p(x) \\
&= \arg\min_x E(x;\hat{x}) + R(x)
\end{aligned}
$$

We've seen several instances of how to model priors.

Ulyanov et al, 2018

Parametrised Image Priors

- x – Clean image
- $\hat{x}$ – Corrupted image (observed)
- x^* – Restored image

Search in the image space

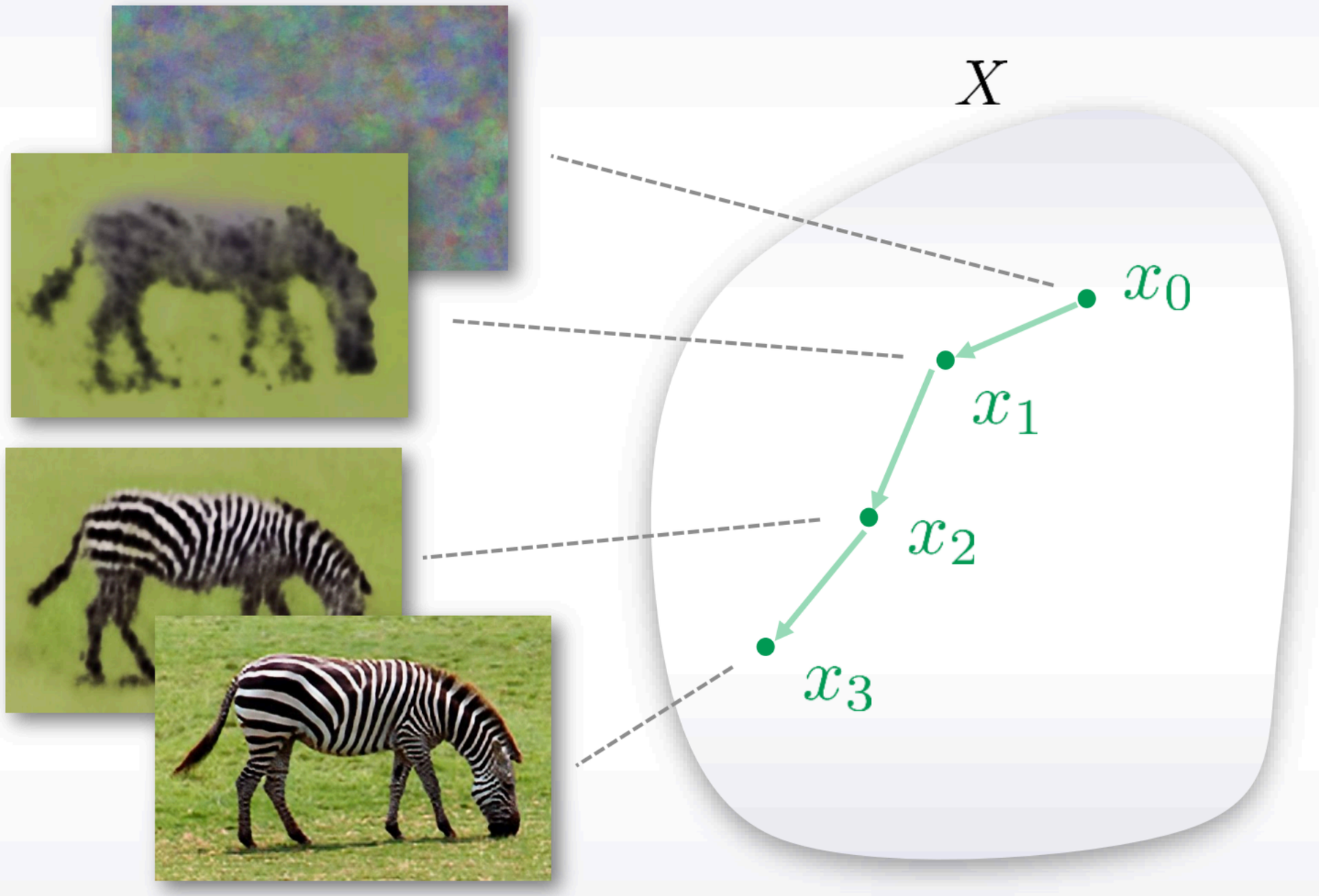
Regular: $\arg\min_{x} E(x; \hat{x}) + R(x)$

Parametrized: $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

Search in some other space

Ulyanov et al, 2018

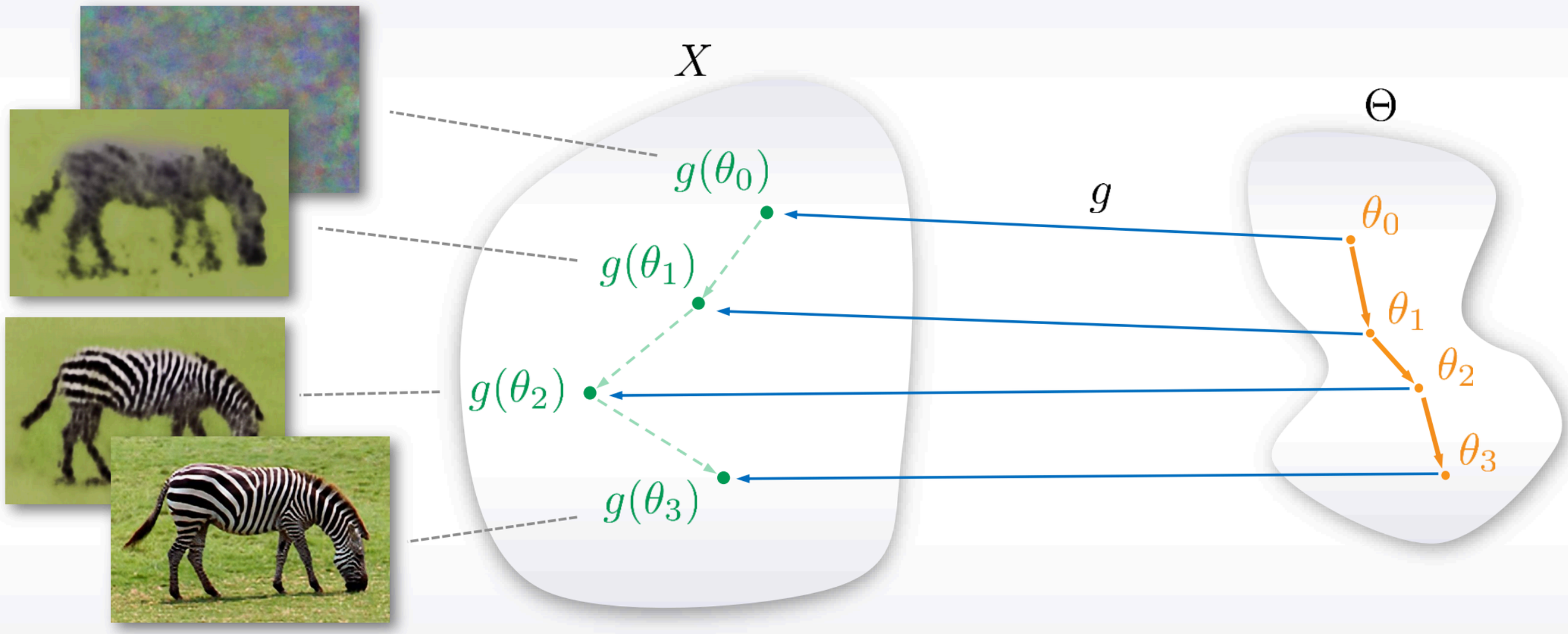
Regular Image Priors



Regular: $\arg\min_x E(x; \hat{x}) + R(x)$

Ulyanov et al, 2018

Parametrised Image Priors



Parametrized: $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

Ulyanov et al, 2018

Parametrised Image Priors

Example:

Parametrized: $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

Theta is sparse
D is a dictionary

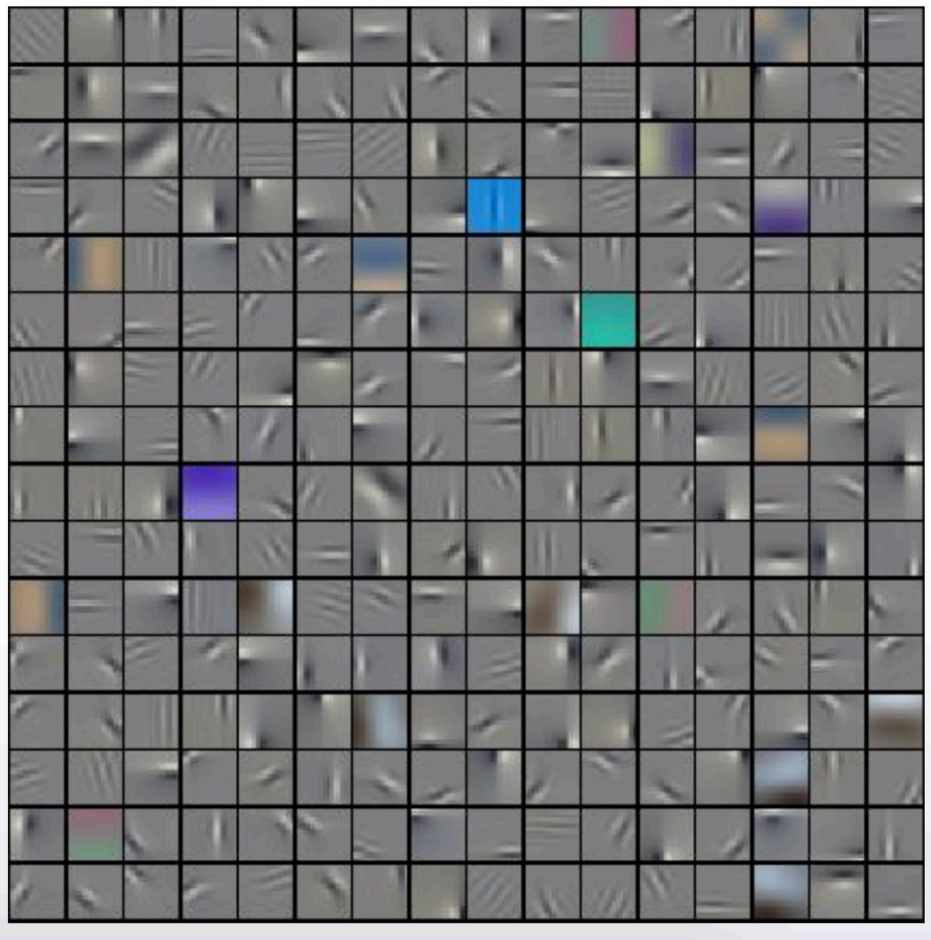
$g(\theta) = D\theta$

Coefficients

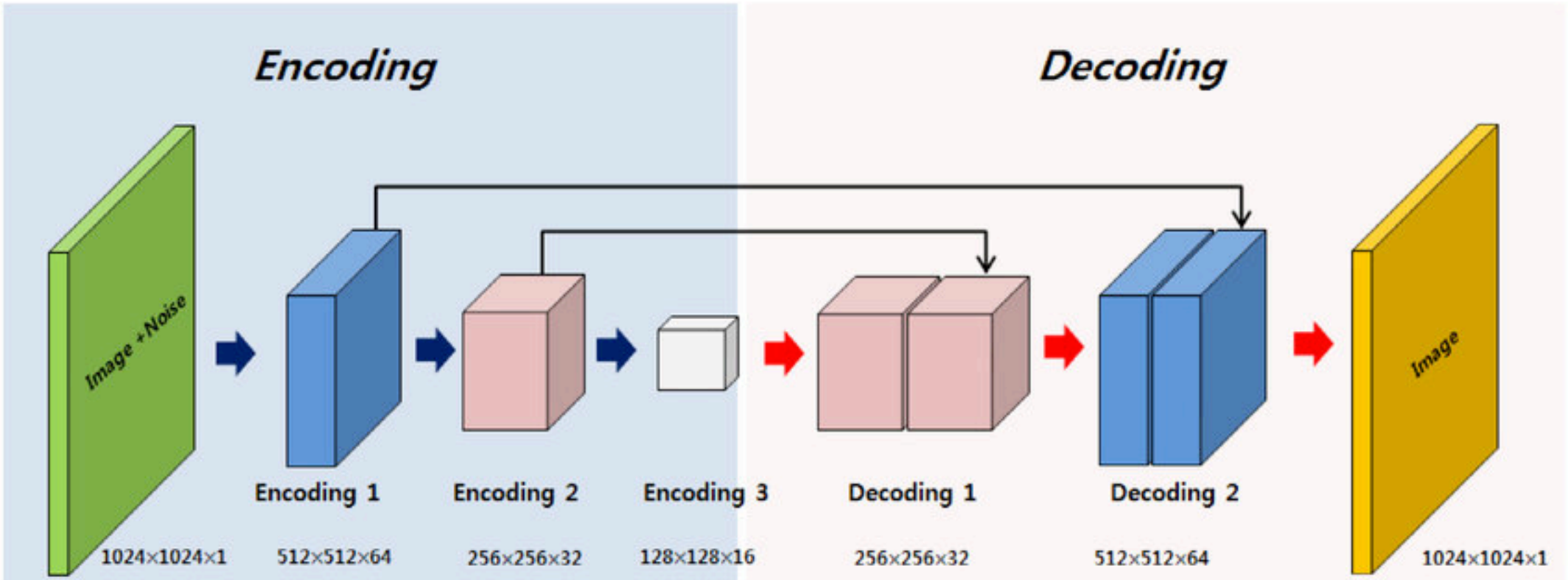
Dictionary

But this model is linear

Limited expressivity



Convolutional Neural Networks



Symbol	Description	Parameters
➡	Convolution and ReLu activation	Kernel size = 3 × 3 Stride = 2 × 2
➡	Deconvolution and ReLu activation	Kernel size = 3 × 3 Stride = 2 × 2
→	Concatenate	Concatenate Encoding 1, Decoding 2 Concatenate Encoding 2, Decoding 1

Picture from Lee et al, 2017

Ulyanov et al, 2018

Deep Image Prior

We know CNNs are excellent generative models!
Let's just model g(Theta) using a CNN and "learn" the prior instead of putting it by hand.

Parametrized: $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

$$g(\theta) \equiv f_\theta(z)$$

Fixed input

Convolutional network with parameters θ

g **itself is a prior**! And maybe it is sufficient to optimize only data term

Optimize only data term: $\arg\min_{\theta} E(f_\theta(z); \hat{x})$

Deep Image Prior: training

- $\hat{x}$ – Corrupted image (observed)

1. **Initialize** z
   - For example fill it with uniform noise U(-1, 1)

2. **Solve**

$$\arg\min_{\theta} E(f_\theta(z); \hat{x}))$$

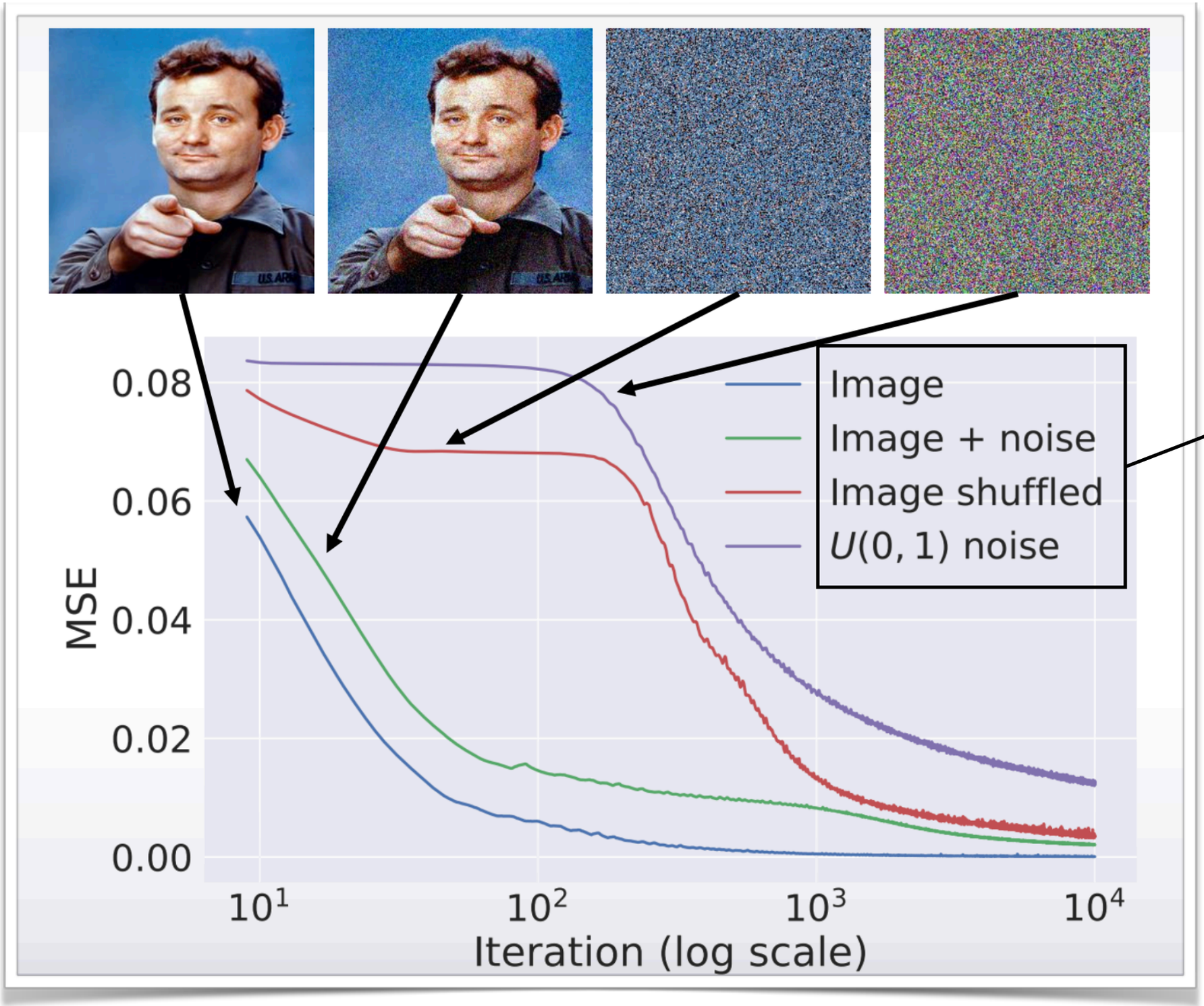
   - With your favorite gradient-based method

$$\theta^{k+1} = \theta^k - \alpha \frac{\partial E(f_\theta(z); \hat{x})}{\partial \theta}$$

3. **Get the solution**

$$x^* = f_{\theta^*}(z)$$

Why should it work?



Networks have very strong expressive capabilities

Experiments

- x – Clean image
- $\hat{x}$ – Corrupted image (observed)
- m – Binary mask

Objective: $\arg\min_\theta E(f_\theta(z); \hat{x}))$

- **Denoising:** $E(x; \hat{x}) = \|x - \hat{x}\|^2$ **Needs early stopping!**
- **Inpainting:** $E(x; \hat{x}) = \|(x - \hat{x}) \cdot m\|^2$
- **Super-Res.:** $E(x; \hat{x}) = \|d(x) - \hat{x}\|^2$

Results + Code

https://dmitryulyanov.github.io/deep_image_prior

Thanks!