The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Spring 2022

INTRODUCTION TO COMPUTER VISION

Atlas Wang
Assistant Professor, The University of Texas at Austin

Visual Informatics Group@UT Austin
https://vita-group.github.io/

Visual Degradation

Data Acquisition

Degradation before Data Acquisition

- Heavy Rain/Snow
- Underwater
- Low Light
- Haze/Sandstorm

Degradation in Data Acquisition

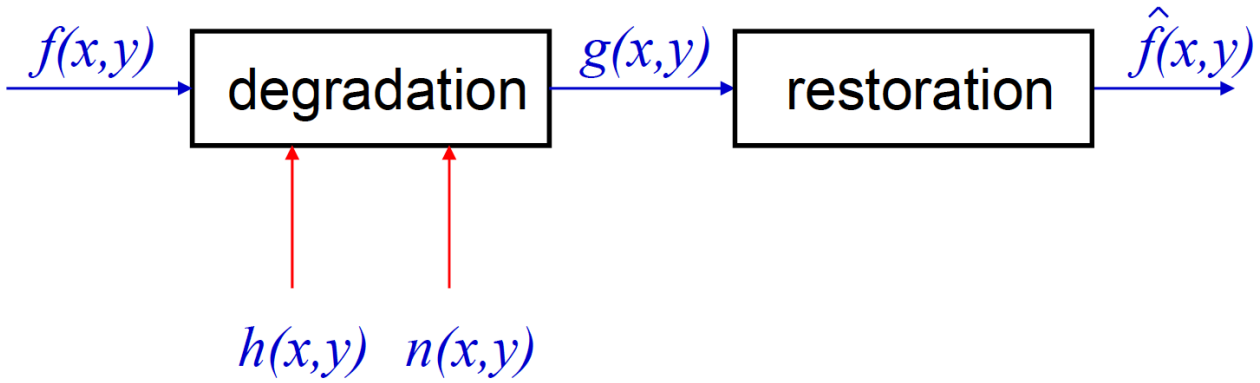
- Downsample
- Motion Blur
- System Noise
- Optical Distortion

Degradation after Data Acquisition

- Scratches
- Watermark
- Mildew
- Compression Loss

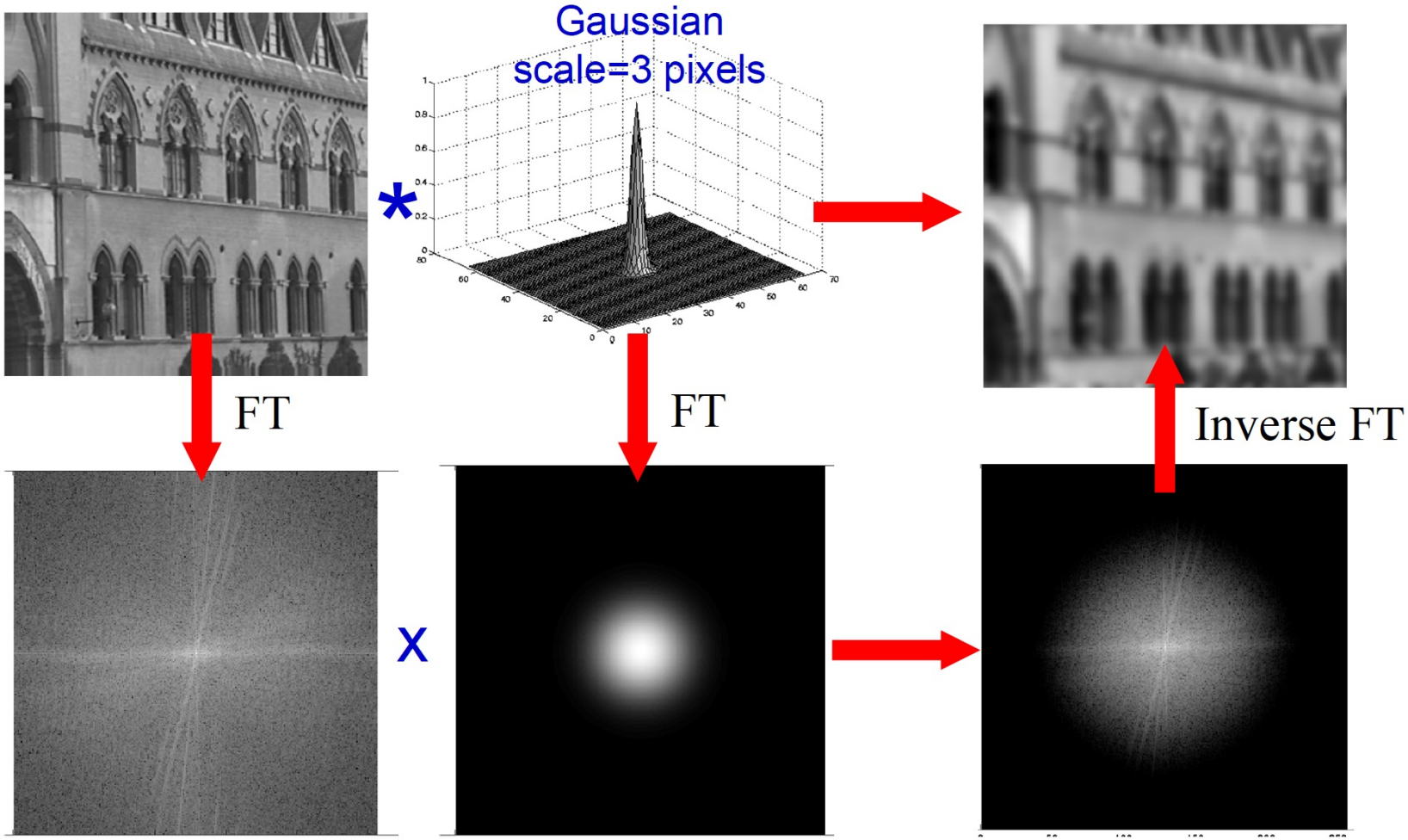
SIMPLIFIED Image Degradation Model

- $f(x,y)$ – image before degradation, 'true image'
- $g(x,y)$ – image after degradation, 'observed image'
- $h(x,y)$ – degradation filter
- $\hat{f}(x,y)$ – estimate of $f(x,y)$ computed from $g(x,y)$
- $n(x,y)$ – additive noise



$$g(x,y) = h(x,y) * f(x,y) + n(x,y) \leftrightarrow \quad G(u,v) = H(u,v)\,F(u,v) + N(u,v)$$

Example: Image Blur



Blurring acts as a low pass filter and attenuates higher spatial frequencies

Goal of Image Enhancement Diversified

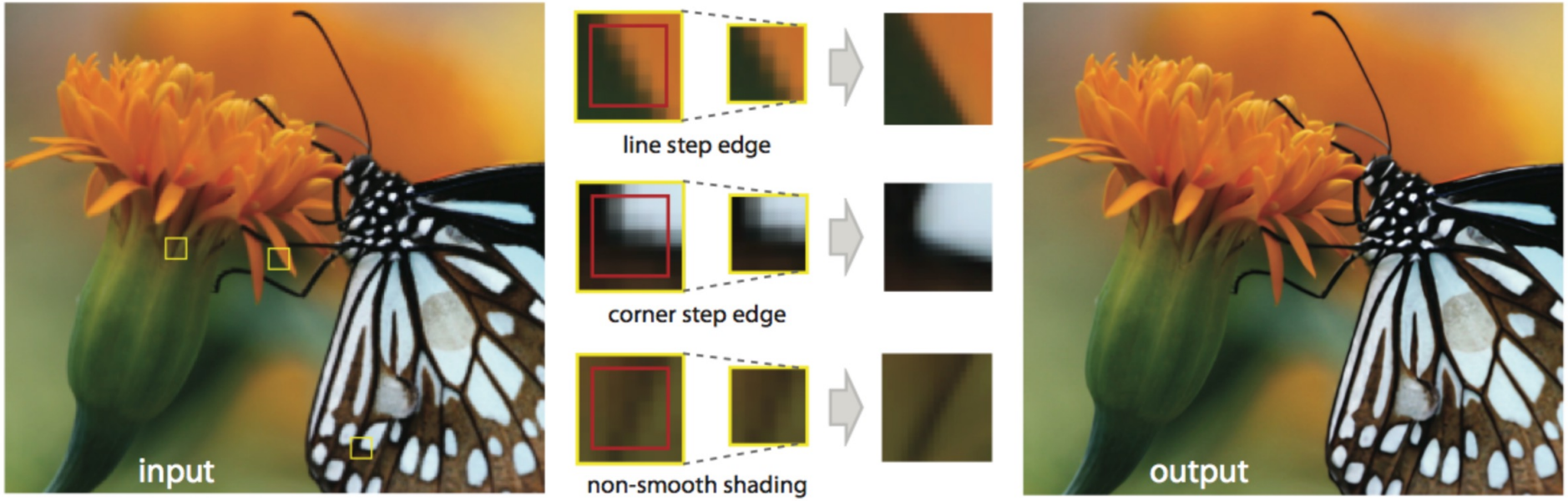
- From traditional signal processing (reconstruction) viewpoint
  - Full-reference metrics: PSNR, SSIM, etc.
- … to human perception (subjective quality)-based
  - No-reference metrics (e.g., NIQE), and human study
- … And to task-oriented, “end utility”-based
  - Typical examples: dehazing, deraining, (extreme) light, underwater …
  - **Representative datasets: RESIDE** dehazing (TIP’18), **MPID** deraining (CVPR’19)
  - **CVPR UG2+ Challenge:** http://www.ug2challenge.org

Discussion: Patch-Based v.s. Image-Level

- The term "**patch-based**" may be vague because it can refer to any algorithm that works with small **image patches**.
  - BM3D image denoising, sparse coding for image super-resolution, image compression algorithms such as JPEG…
- Traditional image processing works on patches
  - Efficiency (esp. when model learning capacity is limited)
  - A lot of natural image statistics and similarities to exploit
- Deep learning image processing works on whole images
  - Mostly obtain better results as they are more "global-view"
  - But often ignore some useful prior knowledge on patch-level

Discussion: Self v.s. External Similarity

- Natural images contain abundant **self-similarities**.
  - For every patch in a natural image, we can probably find many similar patches in the same image.
  - Nonlocal patch-based methods exploit this self-similarity by finding/collecting similar patches and processing them jointly.
  - Cross-scale self-similarity (*Example Below*)



Learning to Enhance Images

- Data-driven training of “end-to-end” models (usually assuming “pairs”)
- Prior/physical information can still be helpful

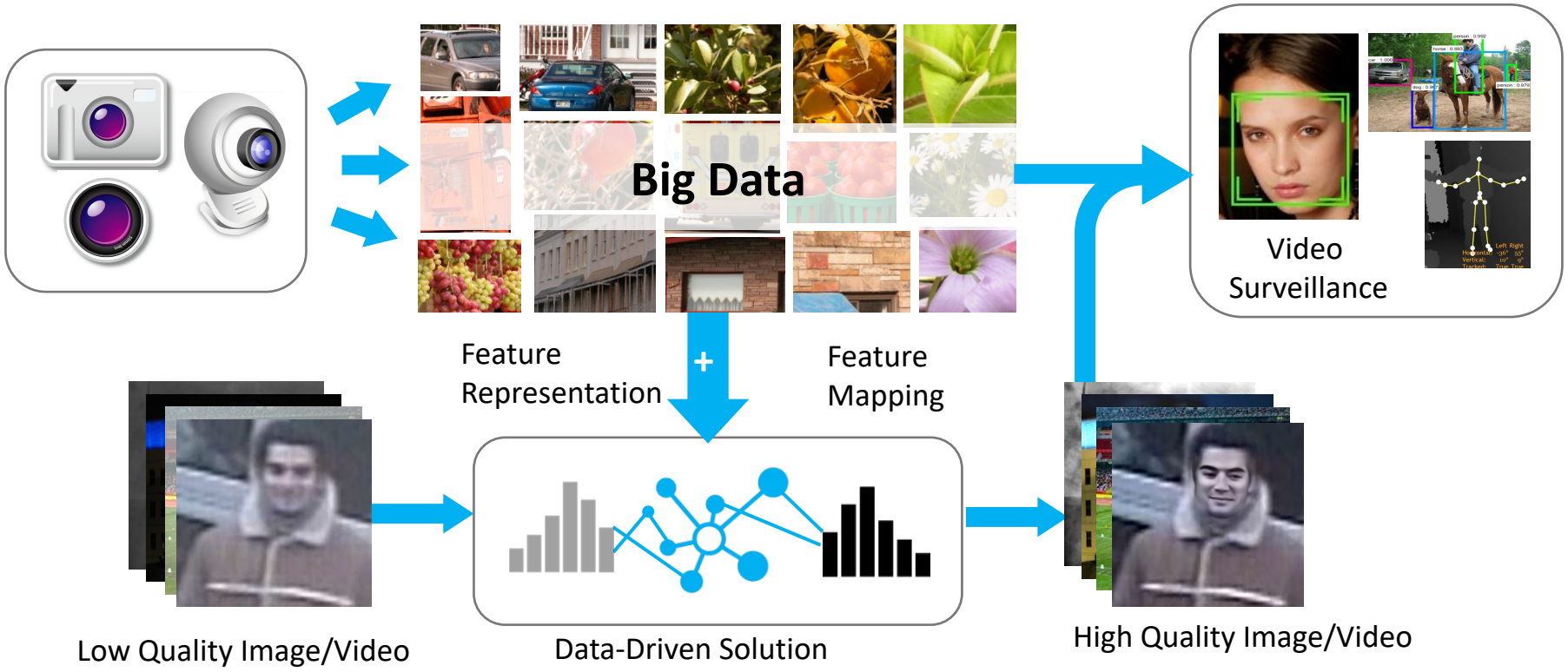


Image Denoising

- Simplest Low-Level Vision Problem

  - Noisy Measurement: $y = x + e$

=

+

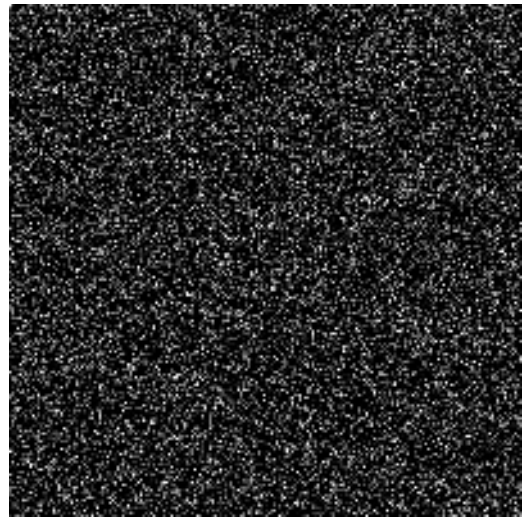


Image Denoising

- Simplest Low-Level Vision Problem

  - Estimate the clean image: $\hat{\boldsymbol{x}} = f(y)$

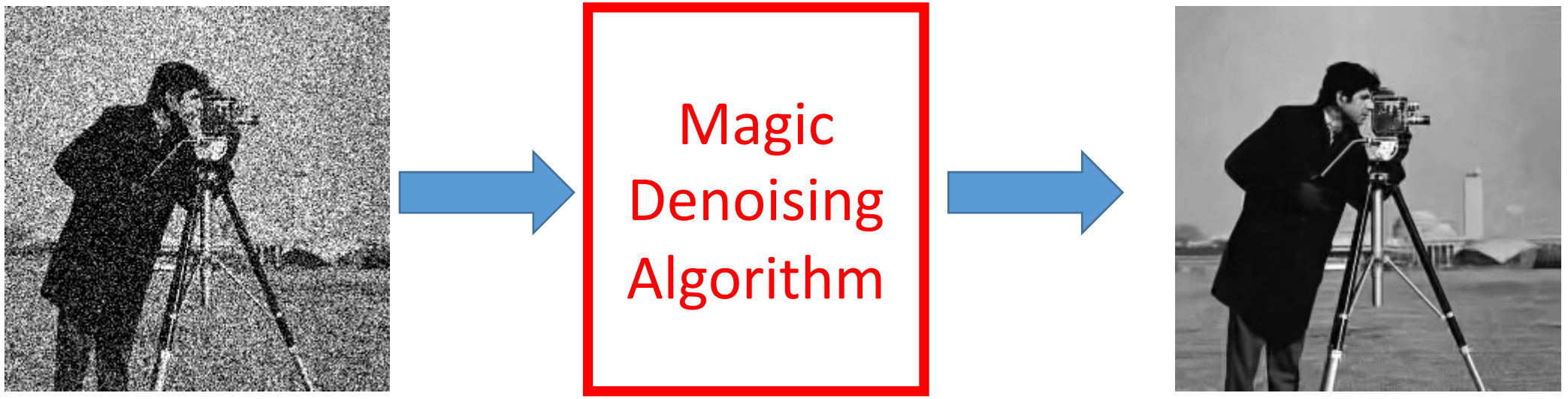


Image Denoising – Conventional Methods

- Collaborative Filtering
  - Non-local Mean, BM3D, etc

Classical Image Denoising: BM3D

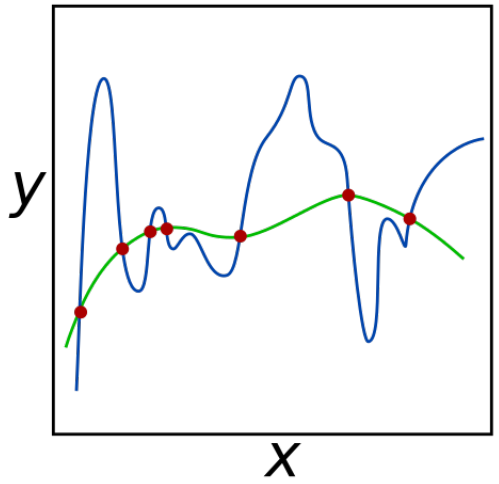
- BM3D = *Block-Matching and 3D filtering*, suggested first in 2007.
- Given a 2D square-block, finds all 2D similar blocks and “group” them together as a 3D array, then performs a *collaborative filtering* (method that the authors designed) of the group to obtain a noise-free 2D estimation.
- Averaging overlapping pixels estimations.
- Gives state of the art results.

Based on: K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. *Image denoising by sparse 3-D transform-domain collaborative filtering*. IEEE Transactions on Image Processing, 16(8):2080–2095, 2007.

Patch-based + Self-Similarity + Domain Expertise

Image Denoising – Conventional Methods

- Collaborative Filtering
  - Non-local Mean, BM3D, etc
- Piece-wise Smooth
  - Total Variation, Tikhonov Regularization, etc



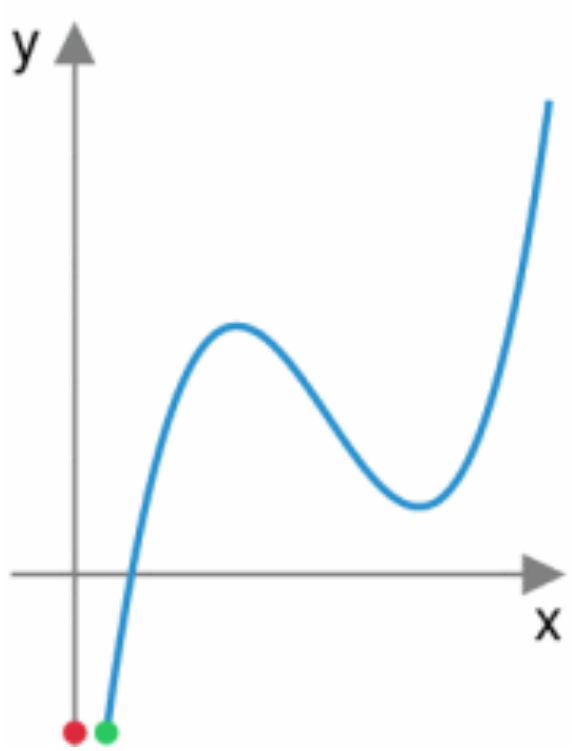
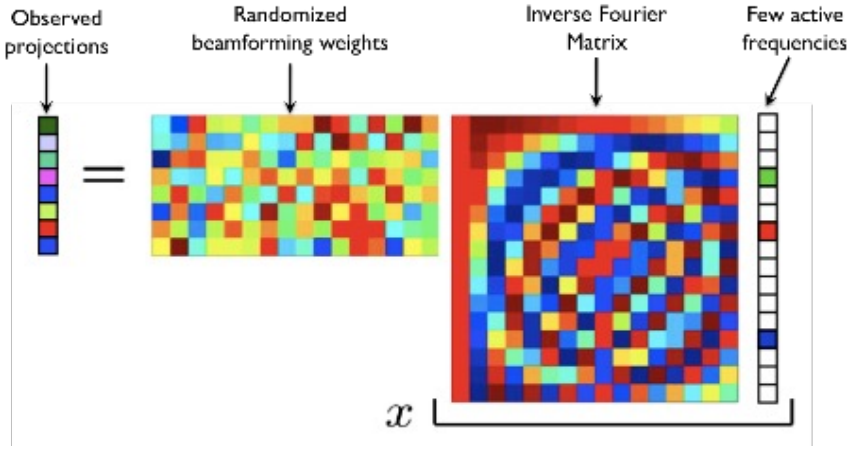
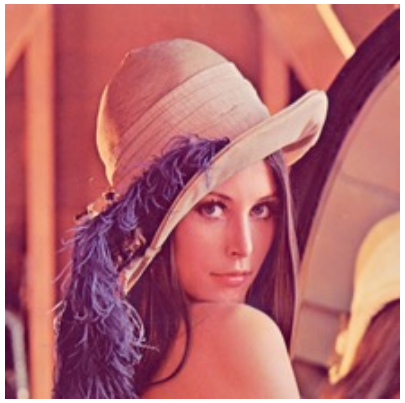
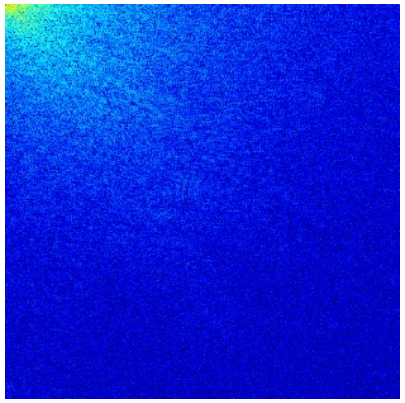


Image Denoising – Conventional Methods

- Collaborative Filtering
  - Non-local Mean, BM3D, etc
- Piece-wise Smooth
  - Total Variation, Tikhonov Regularization, etc
- Sparsity
  - Discrete Cosine Transform (DCT), Wavelets, etc
  - Dictionary Learning: KSVD, OMP, Lasso, etc
  - Analysis KSVD, Transform Learning, etc







It is all about good "prior"

Image Deblurring

- Blurred Measurement: $y = M \otimes \boldsymbol{x}$

=

$\otimes$

$$M = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Image Deblurring

- Estimate the stable image: $\hat{\boldsymbol{x}} = f(y)$

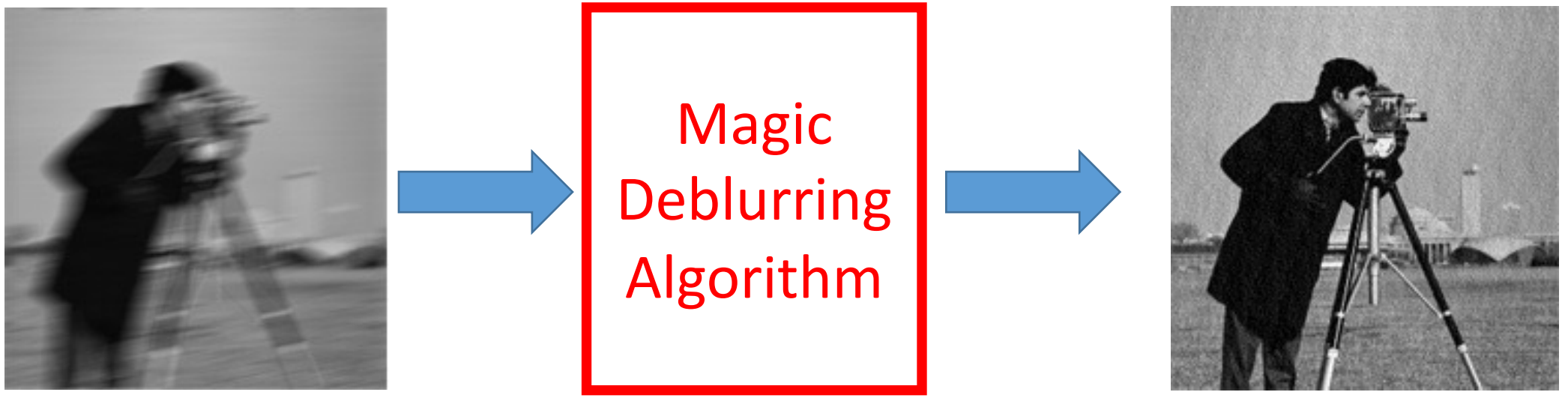


Image Deblurring

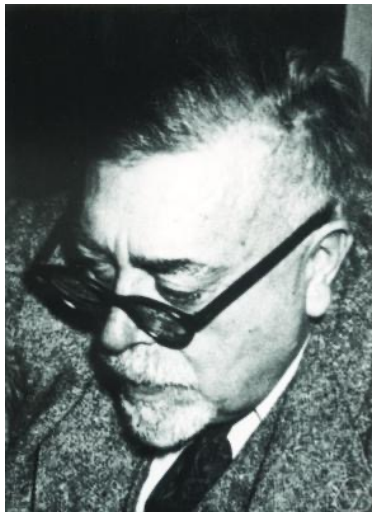
- Non-blind Image Deblurring
  - Suppose you know the blurring kernel, $\boldsymbol{M}$.
  - $\widehat{\boldsymbol{x}} = f(y, M)$
  - All training data need to have consistent $\boldsymbol{M}$, as the testing data

Image Deblurring

- Non-blind Image Deblurring
  - Suppose you know the blurring kernel, $\boldsymbol{M}$.
  - $\hat{\boldsymbol{x}} = f(y, M)$
  - All training data need to have consistent $\boldsymbol{M}$, as the testing data
- Blind Image Deblurring – More challenging yet practical problem
  - Estimate both the image, and the blurring kernel
  - $\{\hat{\boldsymbol{x}}, M\} = f(y)$

Wiener Filtering

Norbert Wiener
(1894-1964)
"Father of cybernetics"



$$\hat{F}(u,v) = W(u,v)\, G(u,v)$$

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)}$$

where

$$K(u,v) = S_\eta(u,v)/S_f(u,v)$$

$S_f(u,v) = |F(u,v)|^2$ power spectral density of $f(x,y)$

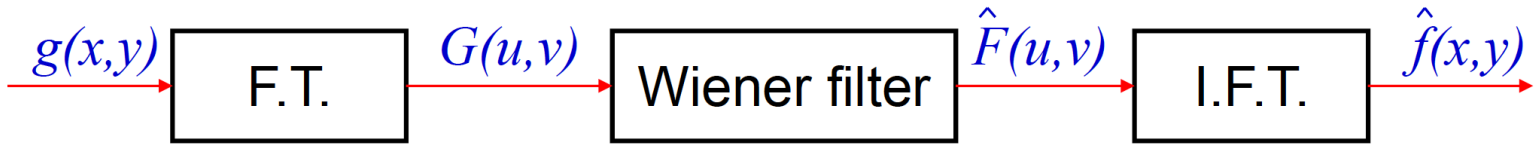
$S_\eta(u,v) = |N(u,v)|^2$ power spectral density of $\eta(x,y)$

Restoration with a Wiener filter

$G(u,v) = H(u,v)\, F(u,v) + N(u,v)$

$\hat{F}(u,v) = W(u,v)\, G(u,v)$

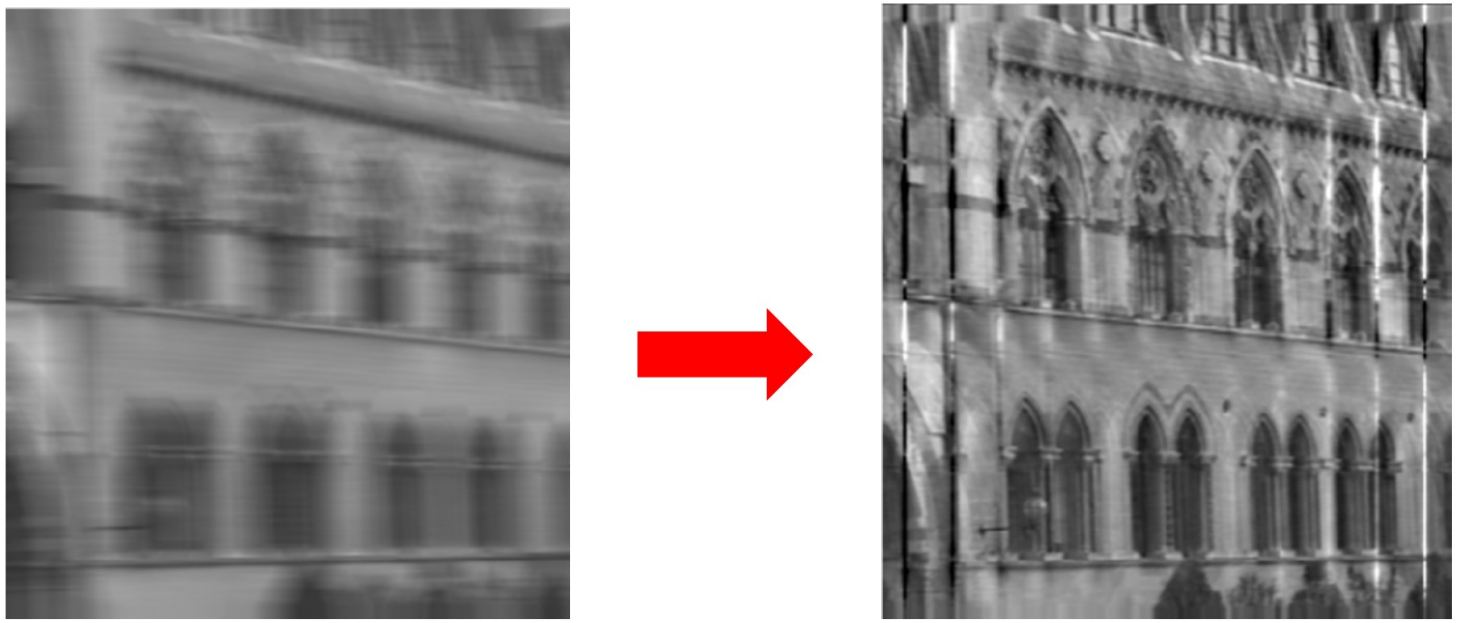
Limitation: Assuming known stationary signal and noise spectra, and additive noise



Example: Motion Deblurring by Wiener Filtering

blur = 20 pixels

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)}$$



1. Compute the FT of the blurred image
2. Multiply the FT by the Wiener filter
3. Compute the inverse FT

$$\hat{F}(u,v) = W(u,v)\, G(u,v)$$

Maximum a posteriori (MAP) Estimation

- original $f(x,y)$
- motion blur
- additive intensity noise

For an image with n pixels, write this process as

$$\hat{\mathbf{g}} = \mathbf{A}\mathbf{f} + \mathbf{n}$$

where $\hat{\mathbf{g}}$ and $\mathbf{f}$ are n-vectors, and $\mathbf{A}$ is an $n \times n$ matrix.

- Estimate $f(x,y)$ by optimizing a cost function:

observed image

generated image

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} (\mathbf{g} - \mathbf{A}\mathbf{f})^2 + \lambda p(\mathbf{f})$$

Likelihood/ loss function

prior/ regularization

Example

$$p(f) = (\nabla \mathbf{f})^2$$

to suppress high frequency noise

Blind Deblurring?

- Estimate *f(x,y)* and *h(x,y)* by optimizing a cost function:

observed image

generated image

$$\min_{\mathbf{f},\mathbf{h}} (\mathbf{g} - \mathbf{A}(\mathbf{h})\,\mathbf{f})^2 + \lambda p_f(\mathbf{f}) + \mu p_h(\mathbf{h})$$

Likelihood/ loss function

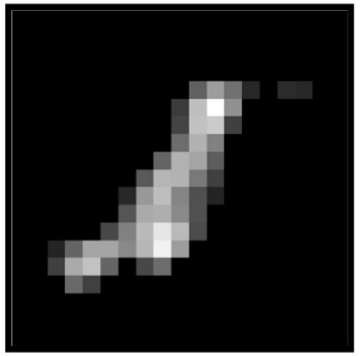
image prior

blur prior

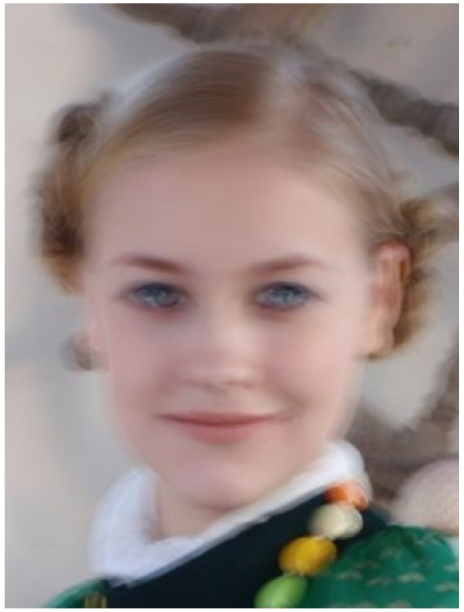
Blind Deblurring

blurred image

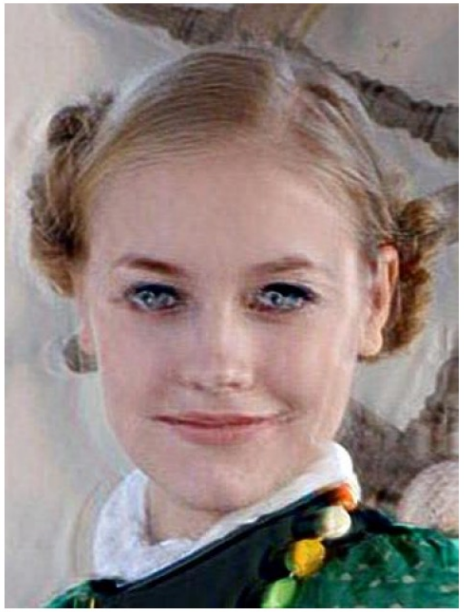
estimated blur filter



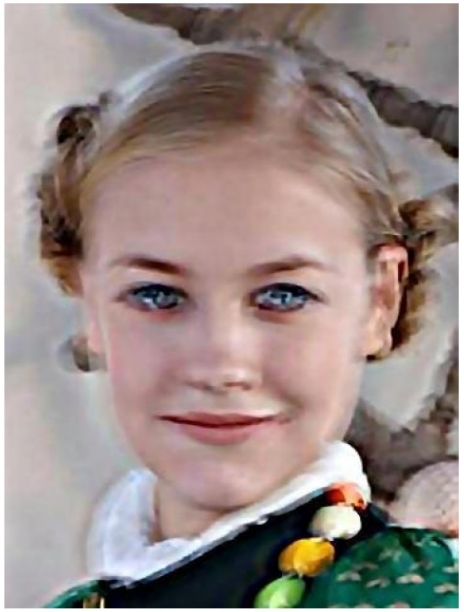
restored image



(a) Blurred photo



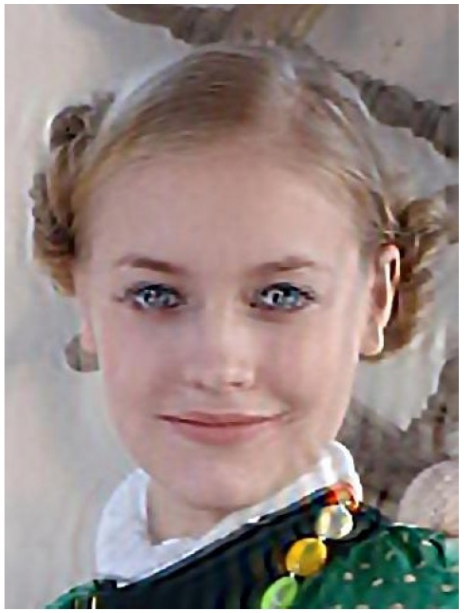
(b) Whyte *et al*. [40]



(c) Krishnan *et al*. [14]

(d) Sun *et al*. [18]

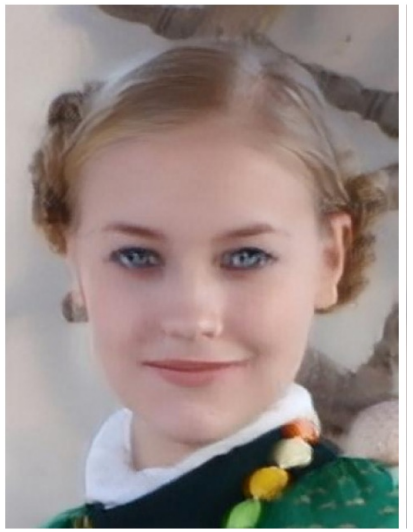
(e) Xu *et al*. [33]



(f) Pan *et al*. [15]

(g) DeepDeblur [2]

(h) SRN [3]



DeblurGAN V2 (2019)


MOTION DEBLURRING

Image Super-Resolution

- Low-Resolution Measurement: $y = D * M \bigotimes \boldsymbol{x}$

=

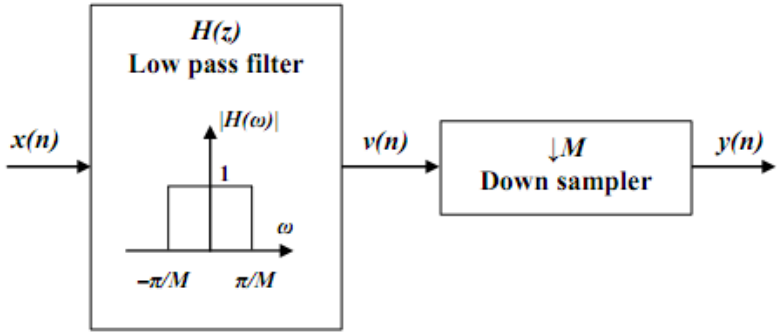
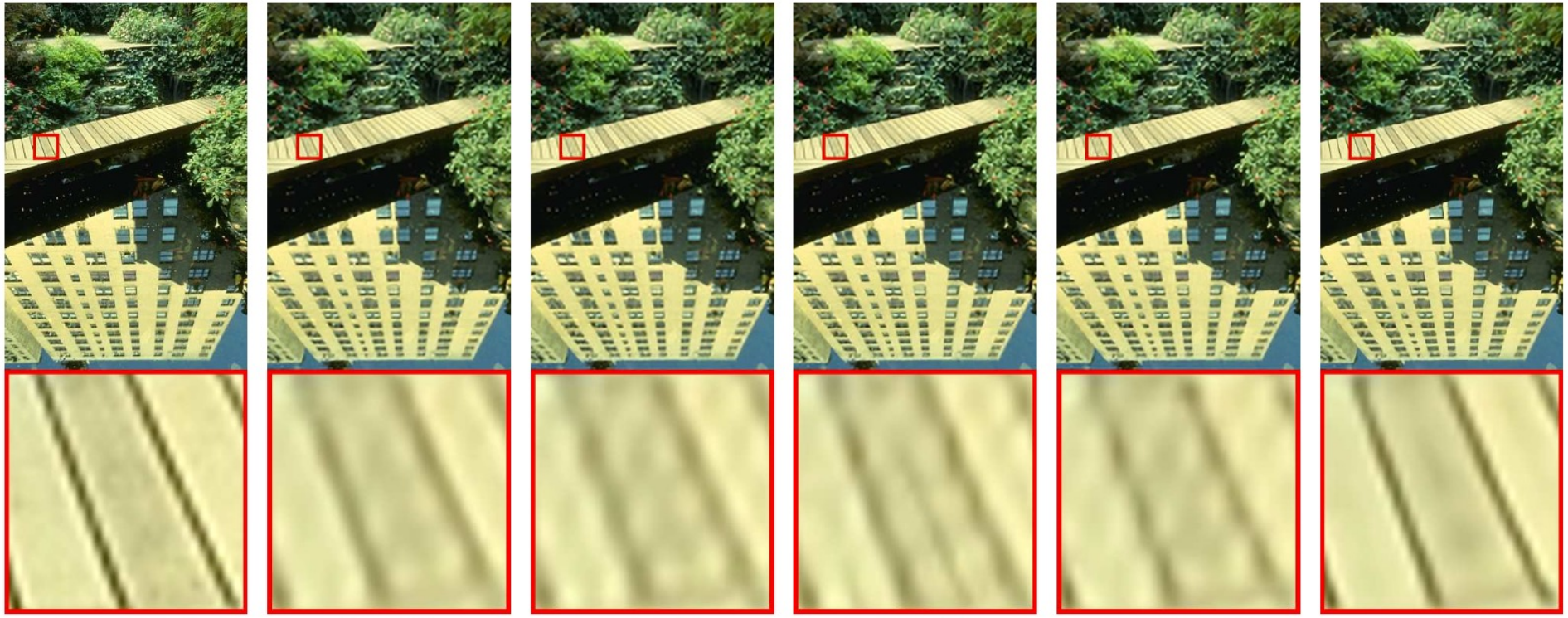


Image Super-Resolution

- Estimate the stable image: $\hat{\boldsymbol{x}} = f(y)$

Image Super Resolution by Deep Learning (2013 – 2017)

Super-resolution results of “148026” (*B100*) with scale factor ×3 (from VDSR paper)



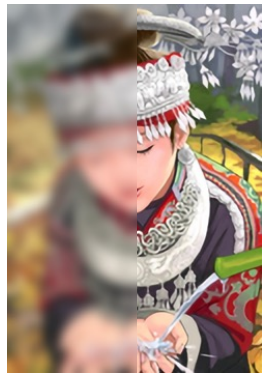
Ground Truth (PSNR, SSIM) | A+ [22] (22.92, 0.7379) | RFL [18] (22.90, 0.7332) | SelfEx [11] (23.00, 0.7439) | SRCNN [5] (23.15, 0.7487) | VDSR (Ours) (23.50, 0.7777)

Many More Tasks in the Real World!

Underwater Enhancement

Dehazing

Inpainting



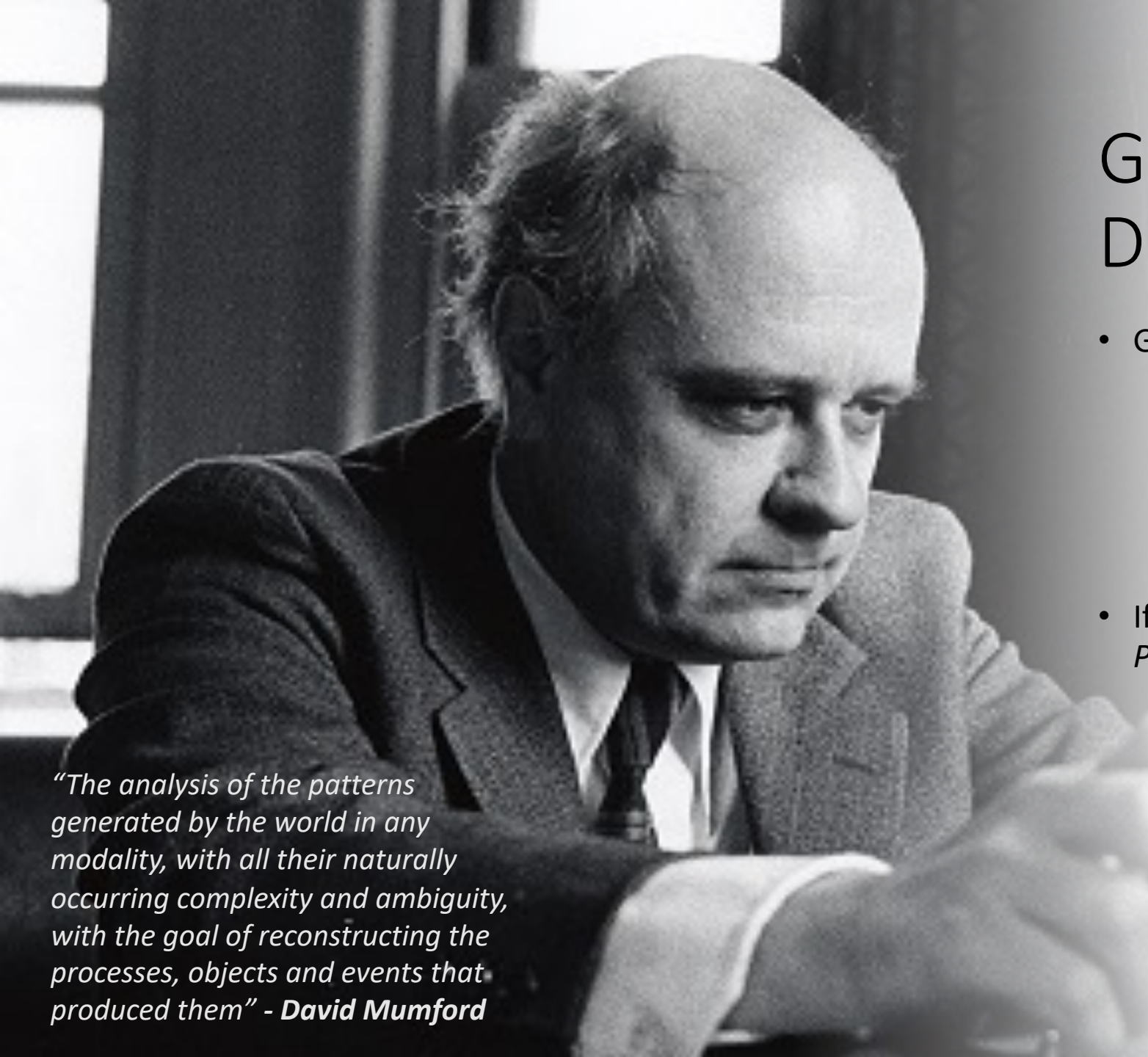
Super Resolution

Rain Removal

Denoising

Low Light Enhancement

. . . . . .

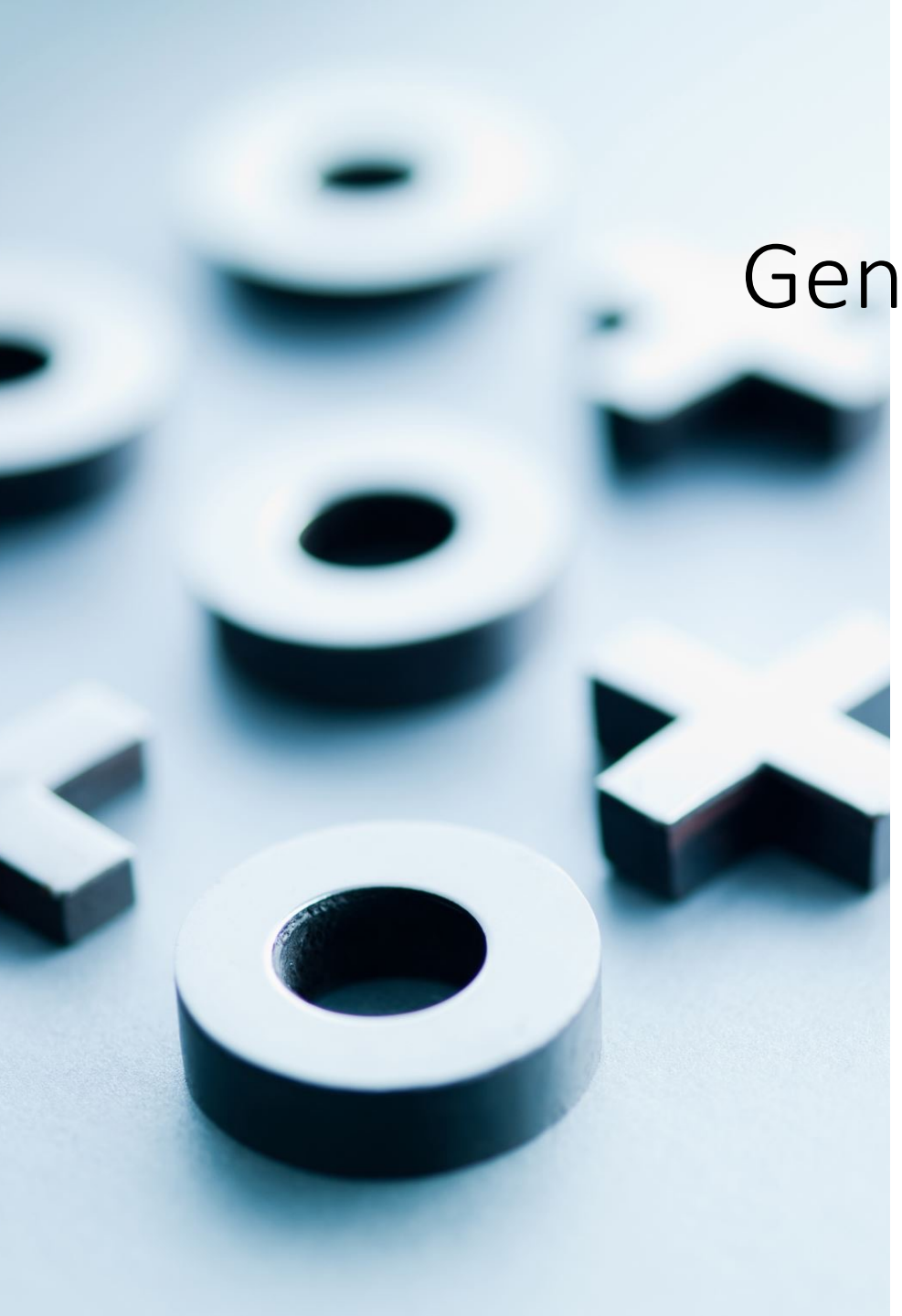


"The analysis of the patterns generated by the world in any modality, with all their naturally occurring complexity and ambiguity, with the goal of reconstructing the processes, objects and events that produced them" - ***David Mumford***

Generative vs. Discriminative Models

- Given a distribution of inputs *X* and labels *Y*
  - Discriminative: model the conditional distribution *P(Y | X).*
  - Generative networks model the joint distribution *P(X , Y).*

- If the model understands the joint distribution *P(X ,Y),* then it
  - can calculate *P(Y | X)* using Bayes rule
  - can perform other tasks like *P(X | Y), i.e.,* generating data from the label (called "conditional generation" in GANs)
  - understands the distribution better than a discriminative model, a scientific philosophy called "**analysis by synthesis"**



Generative vs. Discriminative Models

- Even if you only have *X*, you can still build a generative model
  - … making generative modeling amendable to unsupervised/semi-supervised representation learning
  - Not every problem is discriminative, but all problems could be generative!

- **However, generative modeling is harder!**
  - Map from *X* to *Y* is typically many to one
  - Map from *Y* to *X* is typically one to many
  - Dimensionality of *X* typically >> dimensionality of *Y*
  - Hence compared to estimating *P(Y | X),* the estimation of *P(X, Y)* usually runs into the “curse of dimensionality”

Setup of Generative Models

Discriminative model: given *n* examples $(x^{(i)}, y^{(i)})$
learn $h : X \to Y$

Generative model: given *n* examples $x^{(i)}$, recover $p(x)$

Maximum-likelihood objective: $\prod_i p_\theta(x) = \sum_i \log p_\theta(x)$

Generation: Sampling from $p_\theta(x)$

Autoregressive Models

Factorize dimension-wise:

$$p(x) = p(x_1)p(x_2|x_1)\dots p(x_n|x_1,\dots,x_{n-1})$$

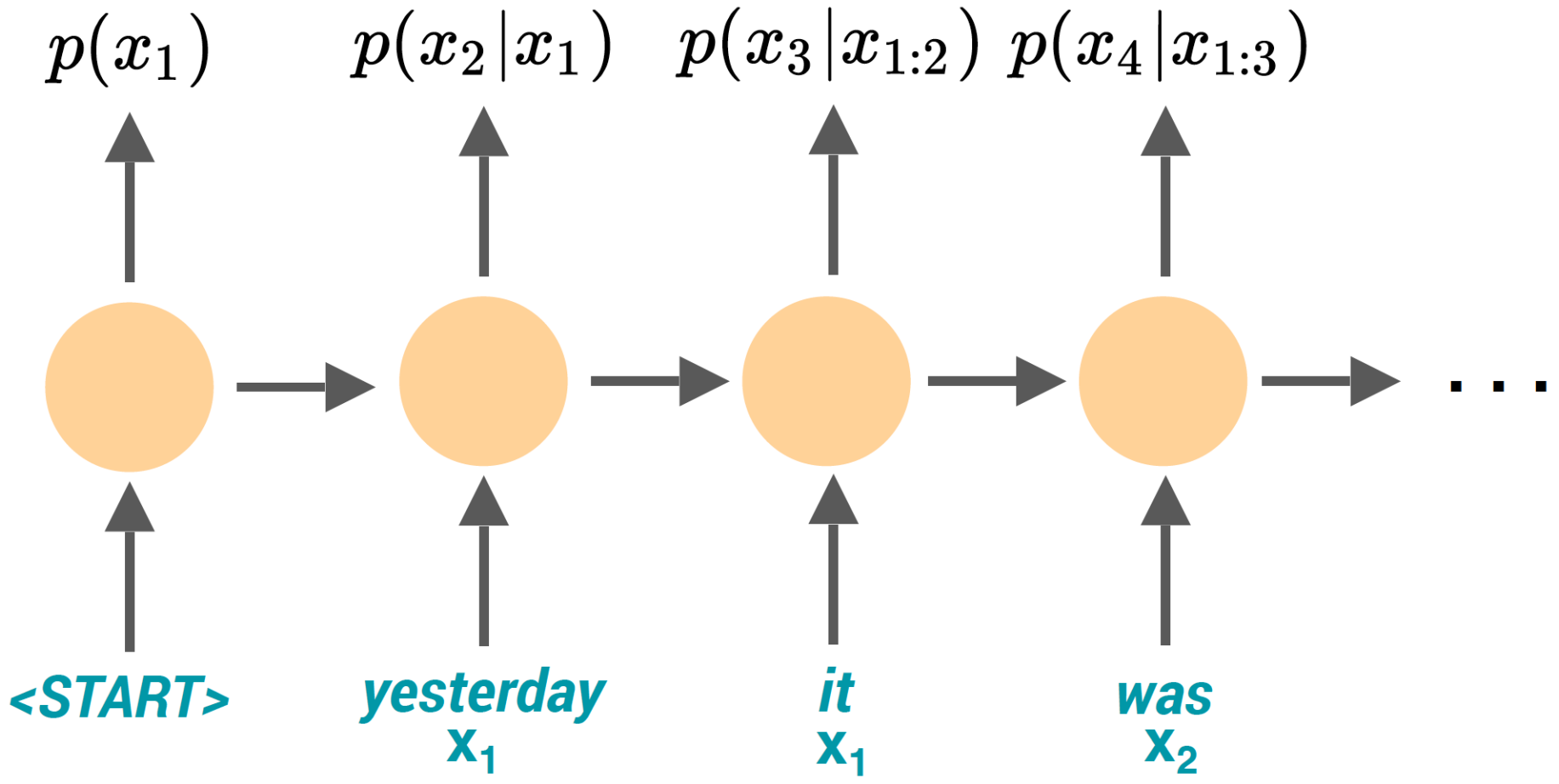
Build a "next-step prediction" model $p(x_n|x_1,\dots,x_{n-1})$

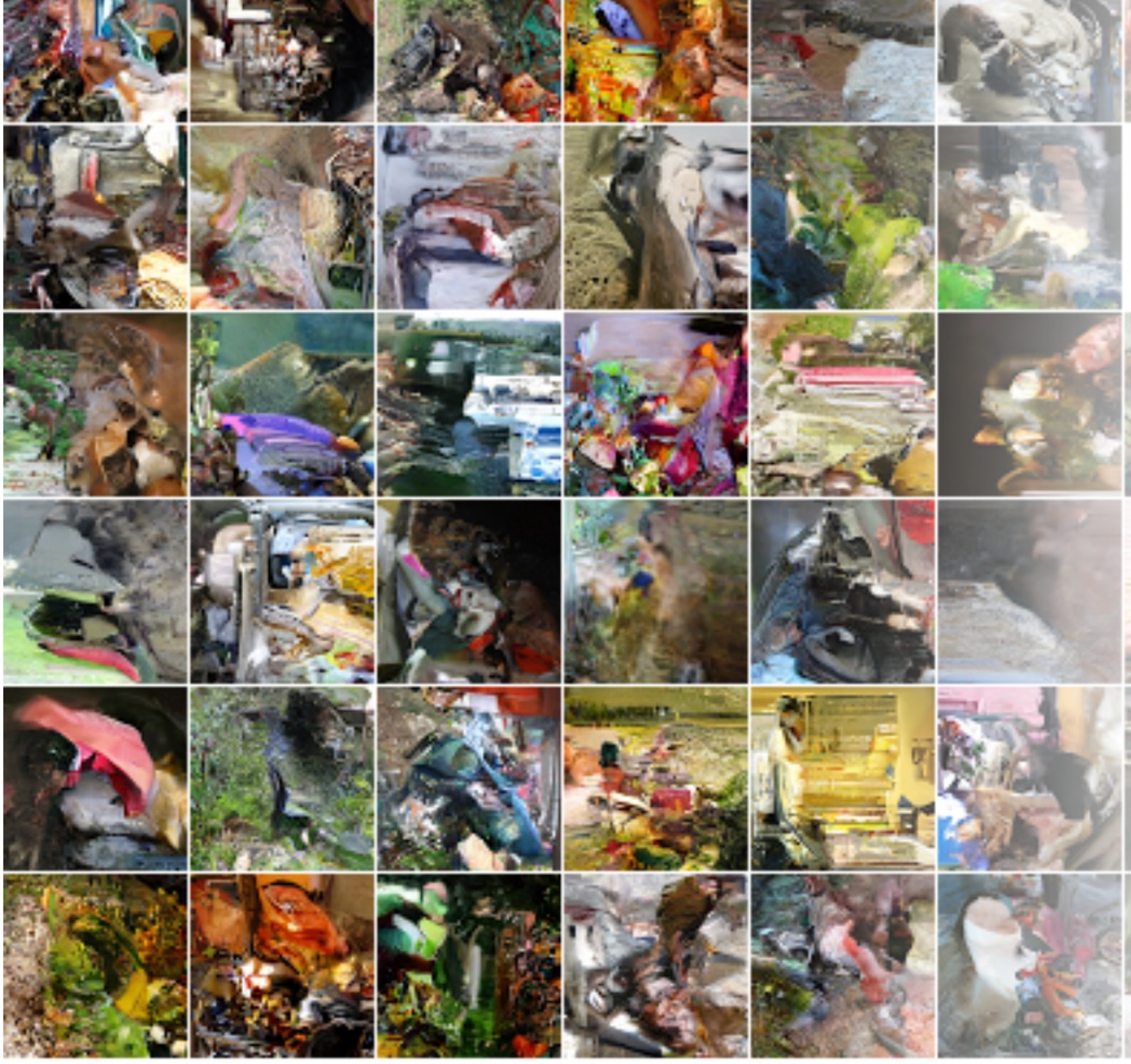
If x is **discrete**, network outputs a probability for each possible value

If x is **continuous**, network outputs parameters of a simple distribution (e.g. Gaussian mean and variance)… *or just discretize!*

Generation: sample one step at a time, conditioned on all previous steps

RNNs for Autoregressive Language Modeling



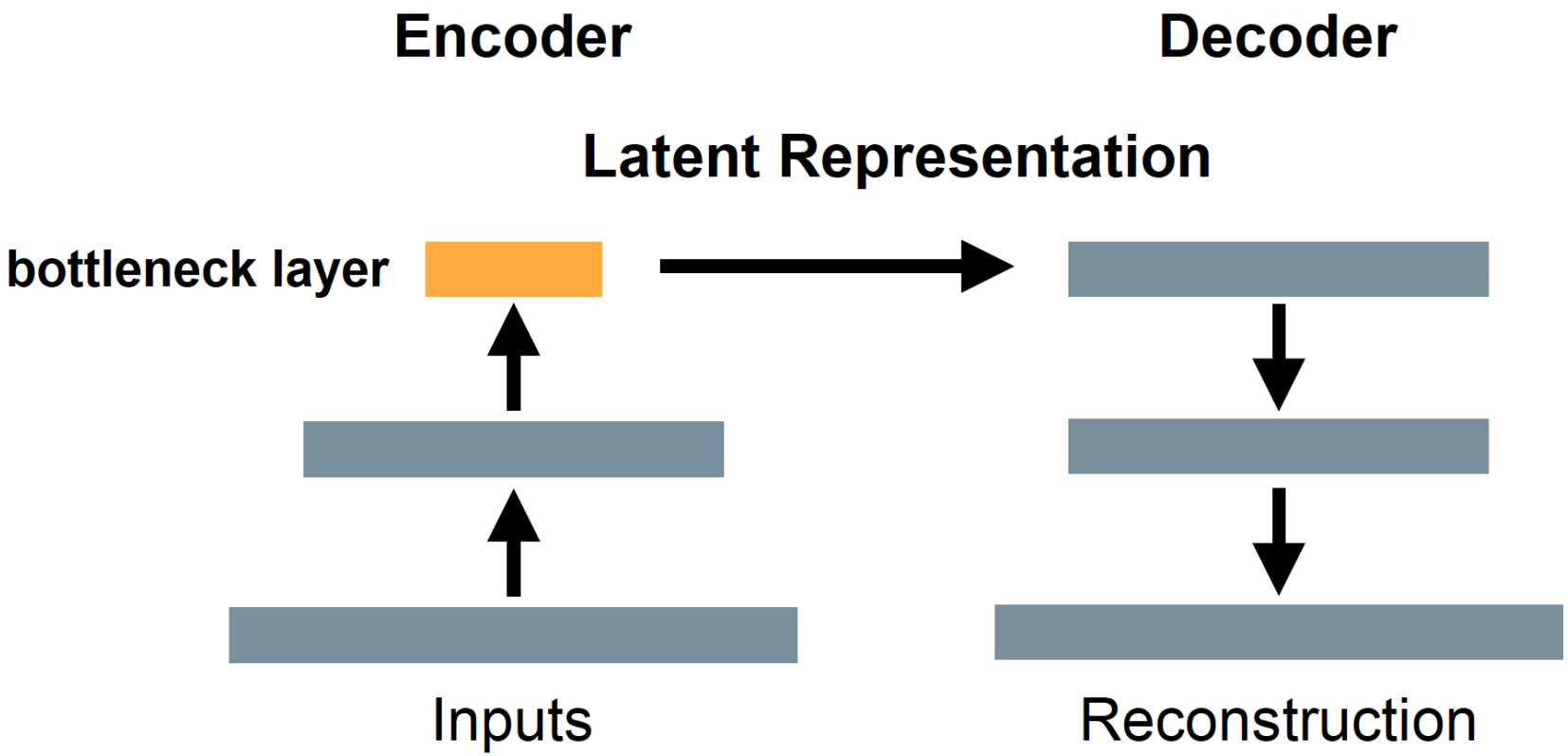


PixelRNN (van der Oord et al. 2016)

- Autoregressive RNN over pixels in an image
- Models pixel generation as discrete-value classification (256-way softmax at each step)

- Problems:
  - Sequential generation can be very slow
  - Not even close to the “true” image generating process

Autoencoders for Representation Learning



$$L = (x - \hat{x})^2$$

Idea: **extending** compression to implicit generative modeling, which allows for sampling!

Variational Autoencoders (VAEs)

$$
\begin{aligned}
\log p(X) &= \log \int_Z p(X, Z) \\
&= \log \int_Z p(X, Z) \frac{q(Z)}{q(Z)} \\
&= \log \left(\mathbb{E}_q \left[\frac{p(X, Z)}{q(Z)} \right] \right) \\
&\geq \mathbb{E}_q \left[\log \frac{p(X, Z)}{q(Z)} \right] \\
&= \mathbb{E}_q \left[\log p(X, Z) \right] + H[Z]
\end{aligned}
$$

- Similar to a typical autoencoder, but allowing for sampling!
  - **Goal:** generative model of P (X)
  - **Loss function:** reconstruct inputs X (in probabilistic sense)
  - **Encoder** models *P(Z | X)*
  - **Decoder** models *P(X | Z)*

- Hidden representation *Z* is to be learned by the model
  - We encourage the marginal distribution over Z to match **a prior *Q(Z)*** (**needs to be pre-chosen**)
  - During training: generated by encoder
  - During testing: sampled from *Q(Z),* assuming E_X *P(Z | X)* $\approx$ *Q(Z)*

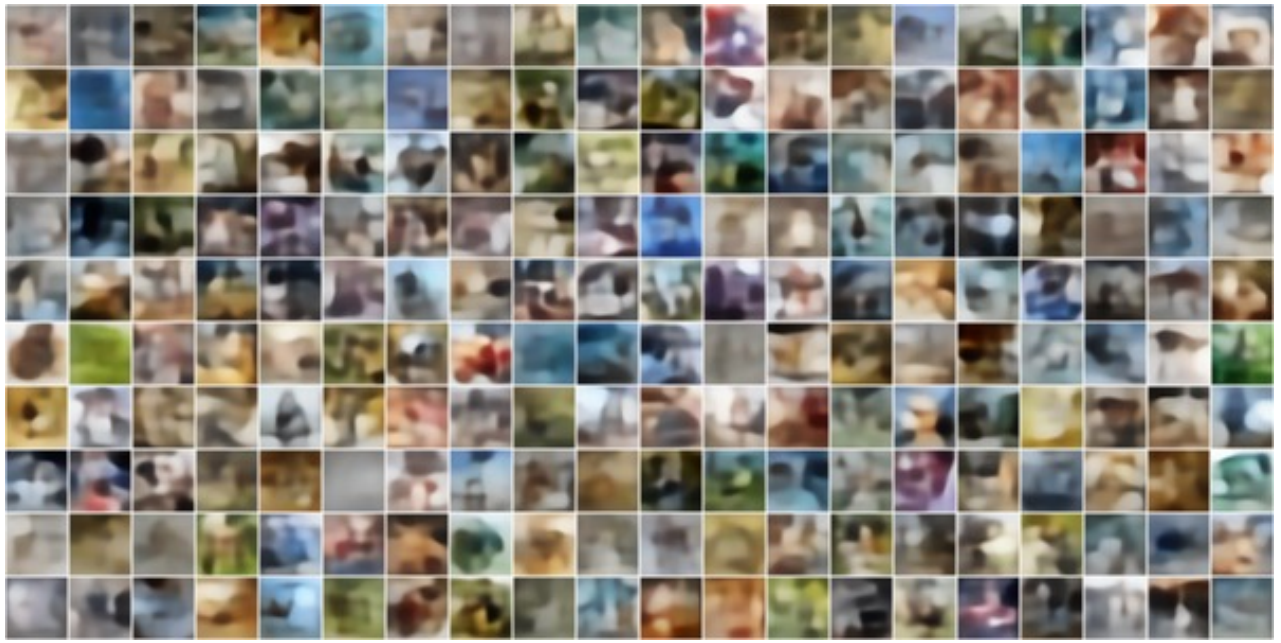
- **Algorithm:** maximizing log P(X)

  -> maximizing its **evidence lower bound** (ELBO)
  - can also be re-expressed as minimizing KL(*Q(Z)* || *P(Z | X))*
  - VAEs require an analytical understanding of the prior *Q(Z)*

VAE Results

Problems:

- Encoder and decoder's output distributions are typically limited (diagonal-covariance Gaussian or similar)
- This prevents the model from capturing fine details and leads to blurry generations

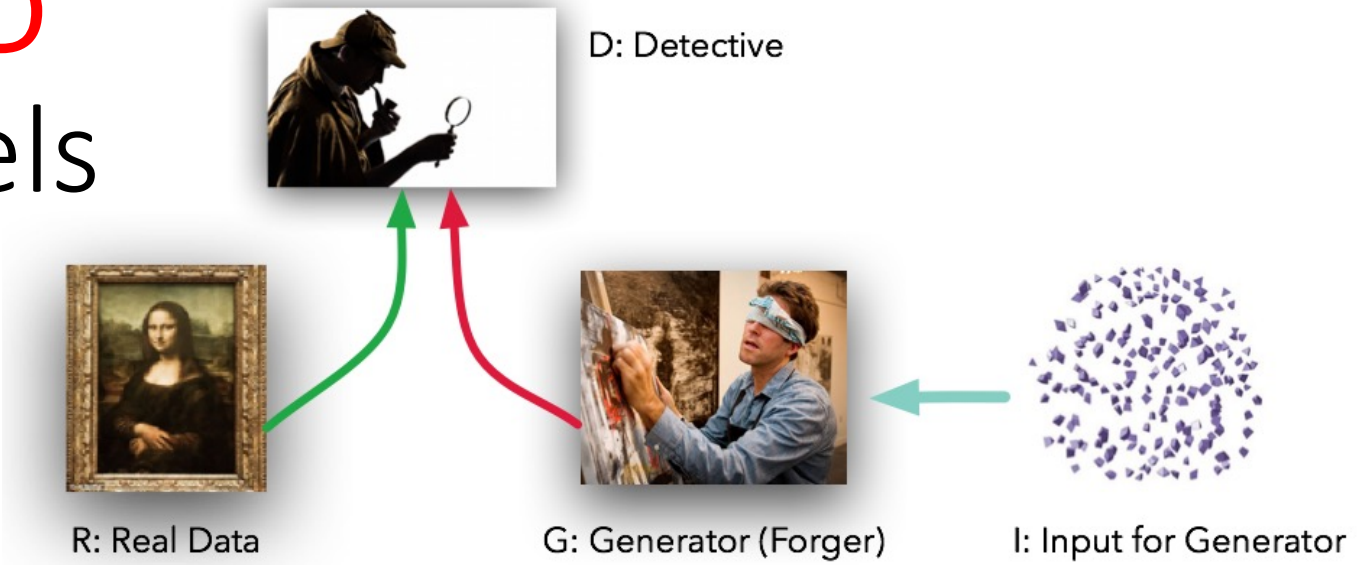


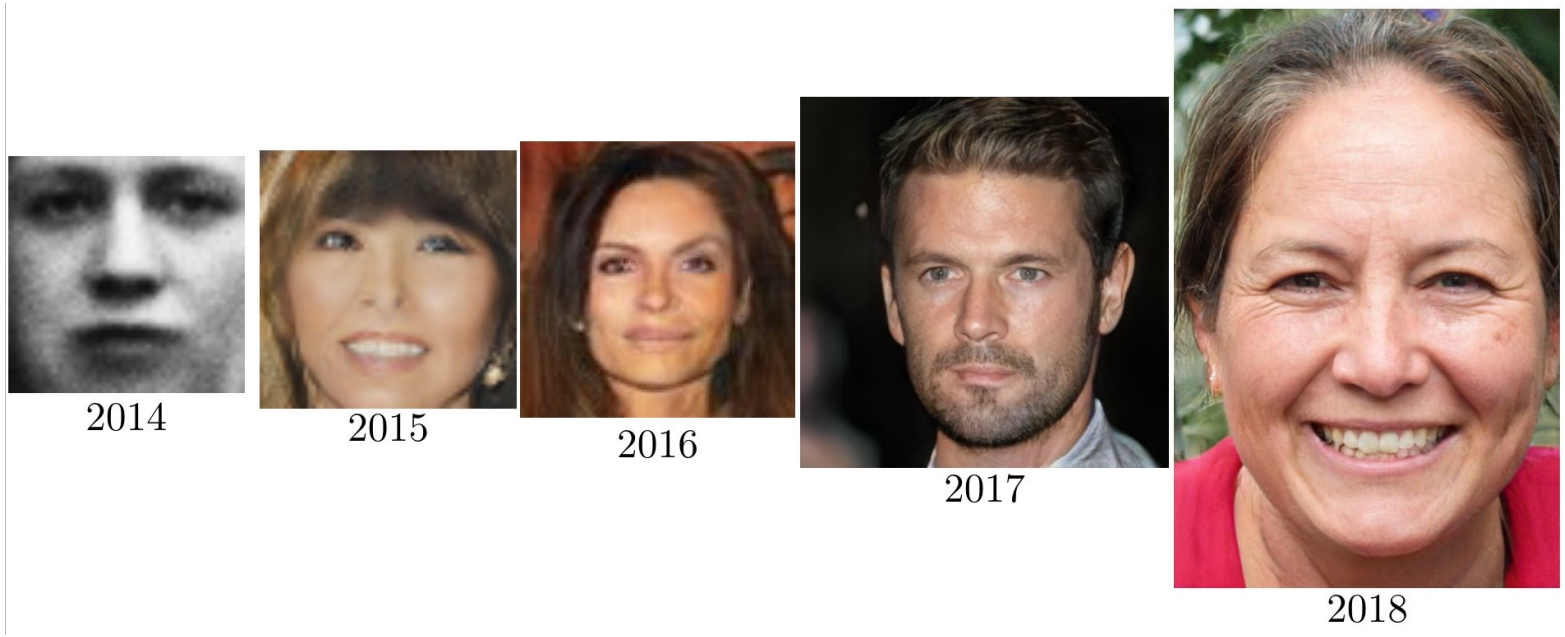
GANs: NEW WORLD of generative models

GAN (generative adversarial network)

= two competing modules:

G (generator) + ***D*** (discriminator)



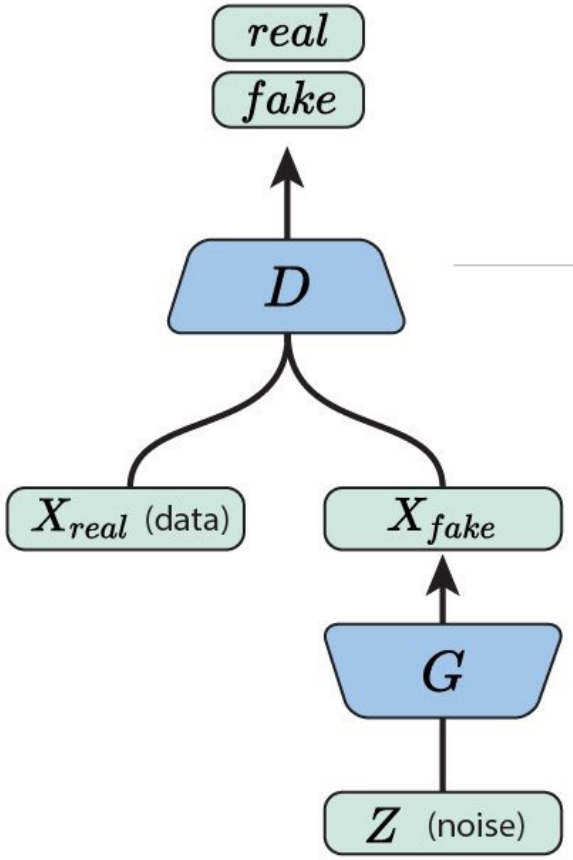


Conceptual Diagram of GANs

Quick Notes:

- **Same goal:** model *P(X)* !
- **G** learns *P(X | Z)*
- **Challenge:** no simple loss function available to measure the divergence
- **Solution:** learning it, using **D !**
  - From the perspective of the **G**, **D** is like an adaptive loss function
  - In most applications, **D** is an auxiliary and thrown away after training; only **G** is wanted

Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.



The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

The **generator** turns random noise into immitations of the data, in an attempt to fool the discriminator.


Donald J. Trump
@realDonaldTrump
You cannot trust CNN! They are FAKE!!!
RETWEETS
7,771
LIKES
2,094
11:07 AM - 21 Nov 2017
364
8K
2K
what people think he's referring to
what he's actually referring to
CNN
Convolution
Fully connected
L1
256x256
L2
128x128
L3
64x64
L4
32x32
F5
F6
(Output)

”DeepFake”

- A person in an existing image or video is replaced with someone else‘s likeness, usually by GAN (sometimes autoencoders)

https://github.com/deepfakes/faceswap

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering