Generative models for images I: Introduction and basics on image restoration

Bruno Galerne
bruno.galerne@univ-orleans.fr

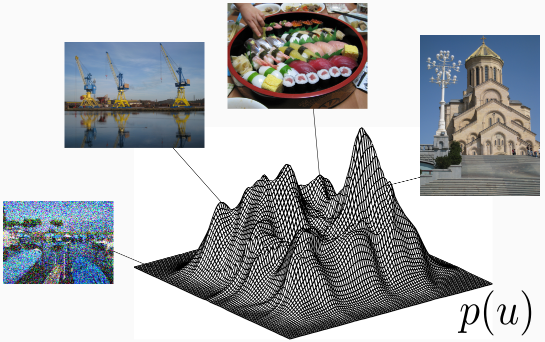
Master MVA 2025-26
Tuesday January 13, 2026

Institut Denis Poisson
Université d'Orléans, Université de Tours, CNRS
Institut universitaire de France (IUF)

Introduction on generative models

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.



(source: Charles Deledalle)

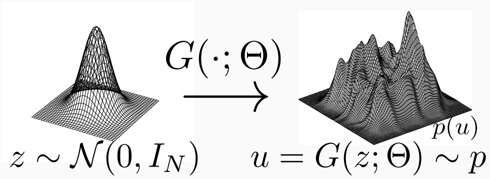
The images may represent:

- different instances of the same texture image,
- all images naturally described by a dataset of images,
- any image

2. Generate samples from this distribution.

Generative models

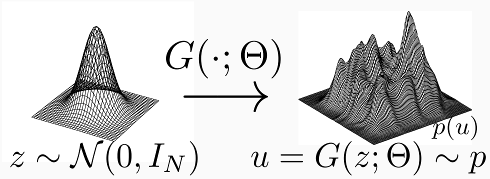
1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot;\Theta)$ is known, then $p = G(\cdot;\Theta)_{\#}\mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot;\Theta)$ is known, then $p = G(\cdot;\Theta)_{\#}\mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

The generator $G(\cdot;\Theta)$ can be:

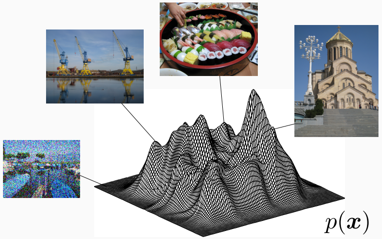
- A deterministic function (e.g. convolution operator),
- A neural network with learned parameter,
- An iterative optimization algorithm (gradient descent,...),
- A stochastic sampling algorithm (e.g. MCMC, Langevin diffusion,...).

Image generation: Gaussian model

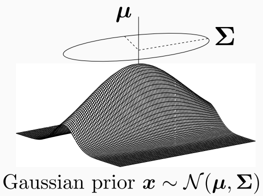
- Consider a **Gaussian model** for the distribution of images $\boldsymbol{x}$ with d pixels:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left[-(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right]$$

  - $\boldsymbol{\mu}$: mean image,
  - $\boldsymbol{\Sigma}$: covariance matrix of images.



$\approx$


$\boldsymbol{\mu}$
$\boldsymbol{\Sigma}$
Gaussian prior $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

(source: Charles Deledalle)

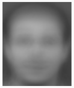
Image generation: Gaussian model

- Take a training dataset $\mathcal{D}$ of images:

$$\mathcal{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$$

$= \Big\{$ $, \dots \Big\}_{\times N}$

- Estimate the mean

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_i \boldsymbol{x}_i =$$ 

- Estimate the covariance matrix: $\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_i (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T = \hat{\boldsymbol{E}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{E}}^T$

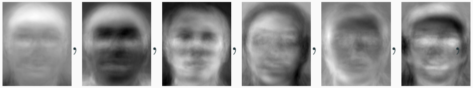
$\hat{\boldsymbol{E}} = \underbrace{\Big\{$  $, \dots \Big\}_{\times N}}_{\text{eigenvectors of } \hat{\boldsymbol{\Sigma}}, \textit{ i.e.}, \text{ main variation axis}}$

Image generation: Gaussian model

You now have learned a **generative model**:

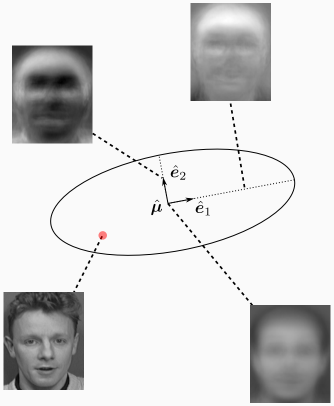
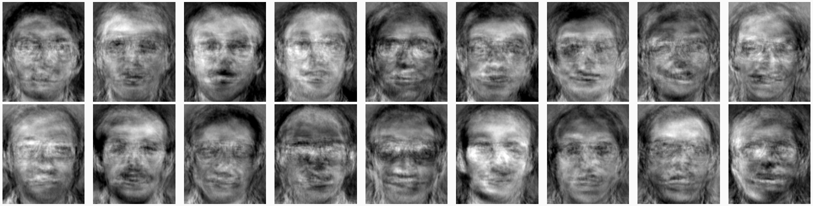


Image generation: Gaussian model

How to generate samples from $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$?

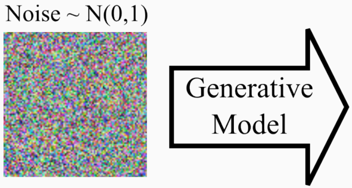
$$\begin{cases} \boldsymbol{z} & \sim \mathcal{N}(0, \boldsymbol{I}_d) \\ \boldsymbol{x} & = \hat{\boldsymbol{\mu}} + \hat{\boldsymbol{E}}\hat{\boldsymbol{\Lambda}}^{1/2}\boldsymbol{z} \end{cases} \leftarrow \text{Generate random latent variable}$$



The model does not generate realistic faces.

- The Gaussian distribution assumption is too simplistic.
- Each generated image is just a linear random combination of the eigenvectors (with independence !).
- The generator corresponds to a one layer liner neural network (without non-linearities).

Image generation: Gaussian model



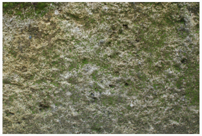
- Deep generative modeling consists in learning non-linear generative models to reproduce complex data such as realistic images.
- It relies on deep neural networks and several solutions have been proposed since the “Deep learning revolution” (2012).

Generative models: Examples

Texture synthesis with a stationary Gaussian model: (Galerne et al., 2011)

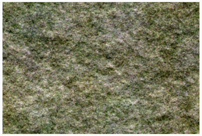
- Data: A single texture image $\boldsymbol{h}$.
- Inferred distribution: p is the stationary Gaussian distribution with similar mean and covariance statistics.
- z is a Gaussian white noise image (each pixel is iid with standard normal distribution).
- G is a convolution operator with know parameters Θ.

Data

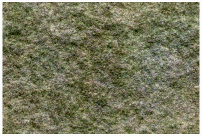


Spot $\boldsymbol{h}$

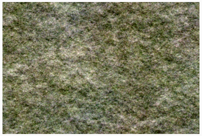
Generated images



$G(z_1; \Theta)$



$G(z_2; \Theta)$



$G(z_3; \Theta)$

Generative models: Examples

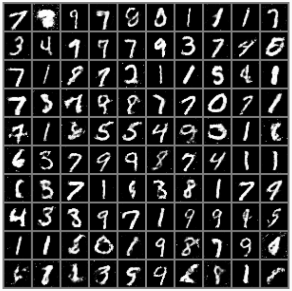
Generative Adversarial Networks: (Goodfellow et al., 2014)

- Data: A database of images.
- Inferred distribution: Not explicit, push-forward measure given by generator.
- z is a Gaussian array in a latent space.
- $G(\cdot;\Theta)$ is a (convolutional) neural network with parameters Θ learned using an adversarial discriminator network $D(\cdot;\Theta_D)$.

Data

MNIST: handwritten digits

Generated images



Fake images (100 epochs)

Image size: 28×28 px

Generative models: Examples

Generative Adversarial Networks: Style GAN (Karras et al., 2019)

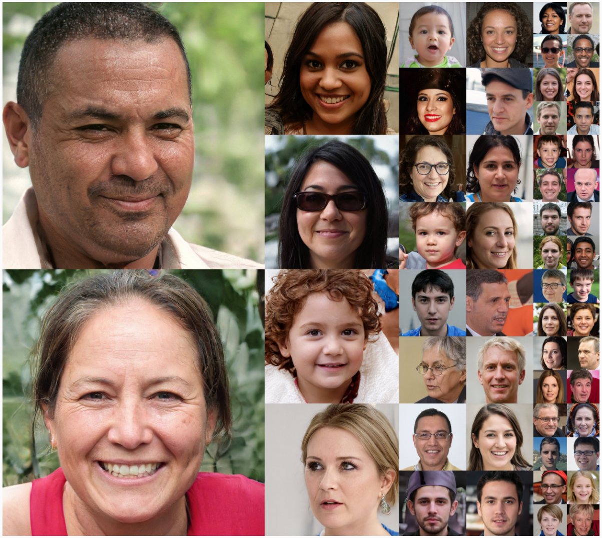
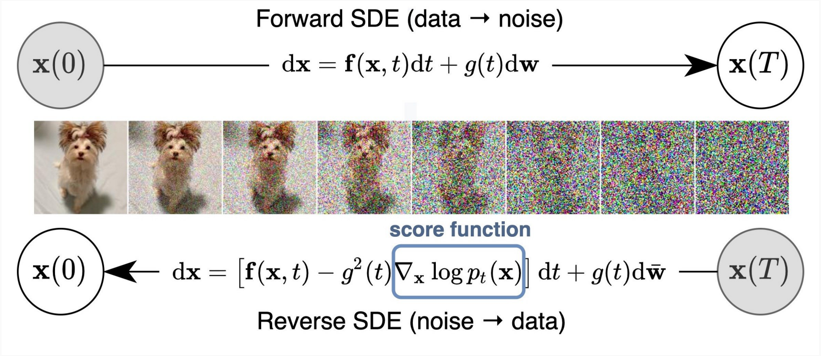


Image size:
1024× 1024 px
(source: Karras et al.)

Denoising diffusion probabilistic models

- Learn to revert a degradation process: Add more and more noise to an image.
- First similar model (Sohl-Dickstein et al., 2015)


Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

(source: Yang Song)

- Probably the most promising framework these days... but things change very quickly in this field!

Diffusion models

(Ho et al., 2020): Denoising Diffusion Probabilistic Models (DDPM): One of the first paper producing images with reasonable resolution.

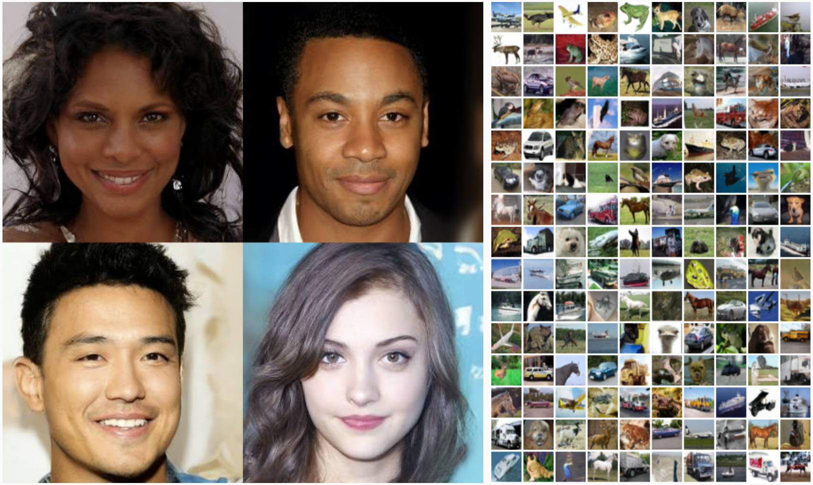


Figure 1: Generated samples on CelebA-HQ 256 $\times$ 256 (left) and unconditional CIFAR10 (right)

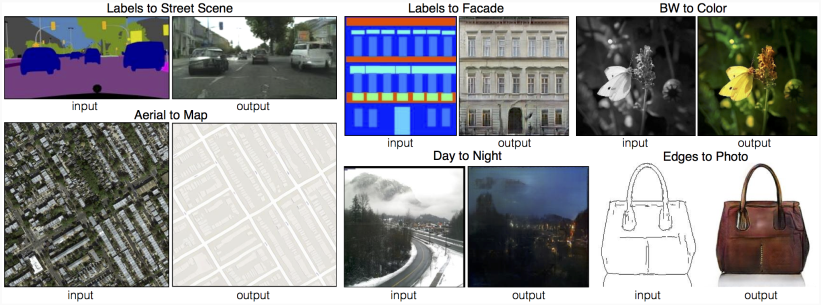
Generative models: Motivations

Why generative models are interesting ?

- **Generating realistic images is important by itself** for entertainment industry (visual effects, video games, augmented reality...), design, advertising industry,...
- **Good image model leads to good image processing:** Generative models can be used as a parametric space for solving inverse problems. Example: Inpainting of a portrait image.
- Also generative models opens the way to **non trivial image manipulation** using **conditional generative models**.

Conditional generative models: Examples

Pix2pix: Image-to-Image Translation with Conditional Adversarial Nets (Isola et al., 2017)



- GAN conditioned on input image.
- Generator: U-net architecture
- Discriminator: Patch discriminator applied to each patch
- Opens the way for new creative tools

(source: Isola et al.)

Conditional generative models: Examples

Latest trends using **diffusion models**: Text to image generation

- DALL·E 1 & 2: CreatingImages from Text (Open AI, January 2021 and April 2022)
- Imagen, Google research (May 2022)

DALL·E 2 (Open AI)

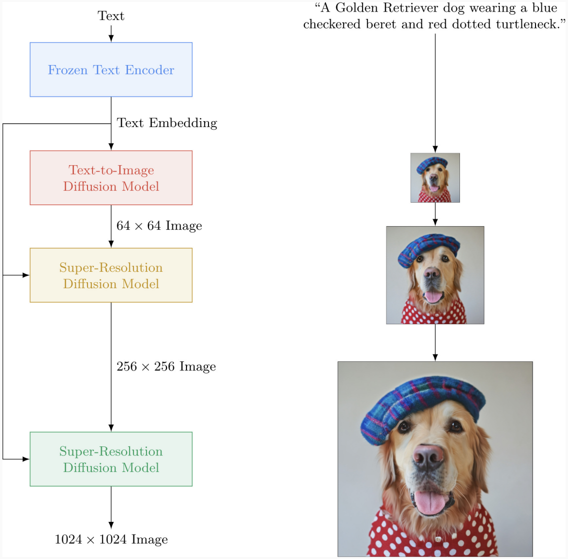
Input: An astronaut riding a horse in a photorealistic style

Imagen (Google)

Input: A dog looking curiously in the mirror, seeing a cat.

Conditional generative models: Examples

Imagen pipeline:


Text
Frozen Text Encoder
Text Embedding
Text-to-Image Diffusion Model
64 × 64 Image
Super-Resolution Diffusion Model
256 × 256 Image
Super-Resolution Diffusion Model
1024 × 1024 Image
"A Golden Retriever dog wearing a blue checkered beret and red dotted turtleneck."

(source: (Saharia et al., 2022))

Conditional generative models: Examples

In August 2022, StableDiffusion was released:

- Based on the paper (Rombach et al., 2022)
- **Open source!**

futuristic tree house, hyper realistic, epic composition, cinematic, landscape vista photography by Carr Clifton & Galen Rowell, Landscape veduta photo by Dustin Lefevre & tdraw, detailed landscape painting by Ivan Shishkin, rendered in Enscape, Miyazaki, Nausicaa Ghibli, 4k detailed post processing, unreal engine
Steps: 50, Sampler: PLMS, CFG scale: 9, Seed: 2937258437

Diffusion models in 2023

Diffusion models are considered mature models and have been used in a large variety of frameworks.

- Diffusion models **beyond image generation**: Text to video, motion generation, proteins, soft robots,...
- **Control of (latent) diffusion models**((Ruiz et al., 2023), (Zhang et al., 2023),...)
- **Diffusion models as priors for imaging inverse problems** ((Chung et al., 2023), (Song et al., 2023), lot of applications in medical imaging, etc.)

Diffusion models in 2023

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

Nataniel Ruiz[*,1,2] Yuanzhen Li[1] Varun Jampani[1]
Yael Pritch[1] Michael Rubinstein[1] Kfir Aberman[1]
[1] Google Research [2] Boston University

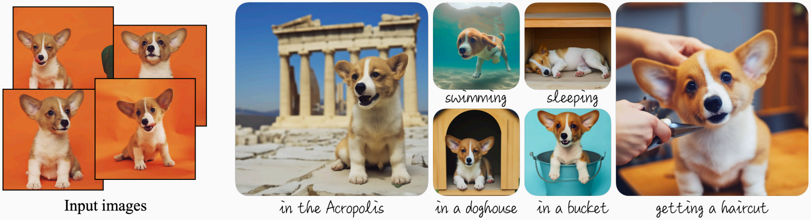


Figure 1. With just a few images (typically 3-5) of a subject (left), *DreamBooth*—our AI-powered photo booth—can generate a myriad of images of the subject in different contexts (right), using the guidance of a text prompt. The results exhibit natural interactions with the environment, as well as novel articulations and variation in lighting conditions, all while maintaining high fidelity to the key visual features of the subject.

(source: (Ruiz et al., 2023))

Diffusion models in 2023

Adding Conditional Control to Text-to-Image Diffusion Models

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala
Stanford University
`{lvmin, anyirao, maneesh}@cs.stanford.edu`

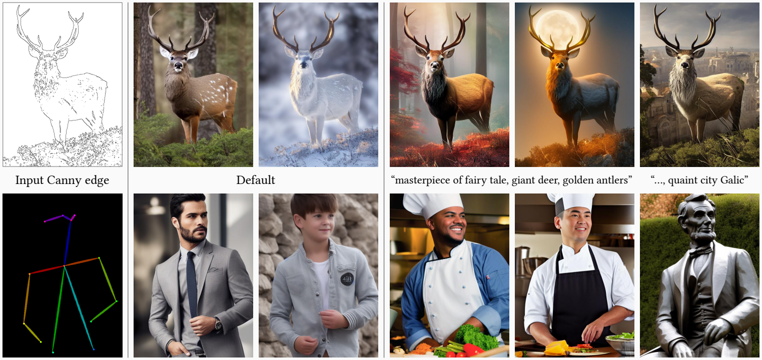


Figure 1: Controlling Stable Diffusion with learned conditions. ControlNet allows users to add conditions like Canny edges (top), human pose (bottom), *etc*., to control the image generation of large pretrained diffusion models. The default results use the prompt "a high-quality, detailed, and professional image". Users can optionally give prompts like the "chef in kitchen".

(source: ControlNet (Zhang et al., 2023))

Diffusion models in 2023

Diffusion posterior sampling for general noisy inverse problems (Chung et al., 2023)

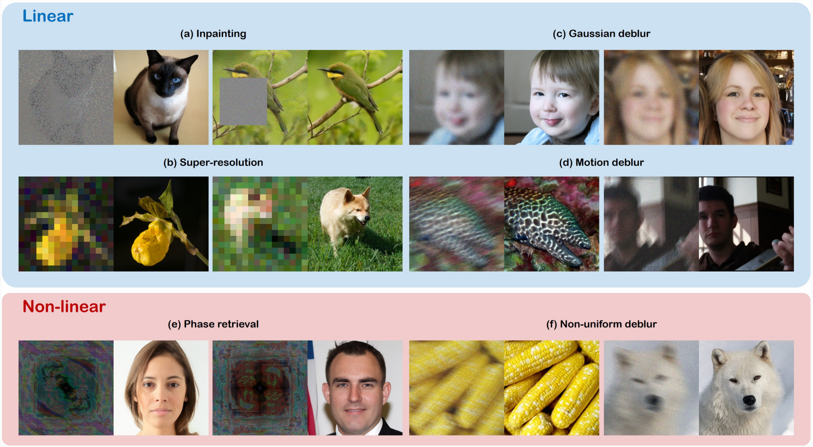


Figure 1: Solving noisy linear, and nonlinear inverse problems with diffusion models. Our reconstruction results (right) from the measurements (left) are shown.

(source: (Chung et al., 2023))

Diffusion models in 2024

- Text-to-video generation.
- Image generation broadly available via conversational agent (ChatGPT, Le Chat by Mistral (based on Flux Pro),...).
- Diffusion models used as “world model”.

DiffusionLight: Light Probes for Free by Painting a Chrome Ball

Pakkapon Phongthawee*[1] Worameth Chinchuthakun*[1,2] Nontaphat Sinsunthithet[1]
Amit Raj[3] Varun Jampani[4] Pramook Khungurn[5] Supasorn Suwajanakorn[1]
[1] VISTEC [2] Tokyo Tech [3] Google Research [4] Stability AI [5] Pixiv
https://diffusionlight.github.io/

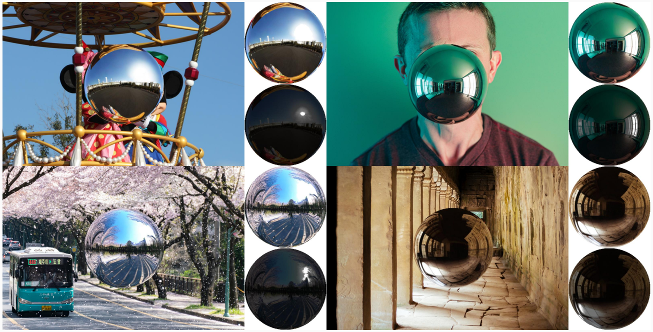


Figure 1. We leverage a pre-trained diffusion model (Stable Diffusion XL) for light estimation by rendering an HDR chrome ball. In each scene, we show our normally exposed chrome ball on top and our underexposed version, which reveals bright light sources, on the bottom.

(source: (Phongthawee et al., 2024))

Generative models for images: Plan of the course 2025-26

Courses shared with **Arthur Leclaire (Telecom Paris)**.

Warm up (today):

- Introduction to generative models for images (done)
- Basics on image restoration/inverse problems (end of today session)

Part I: Established generative models frameworks based on CNN

1. Generative models based on likelihood maximization (**BG**, 1 course)
    - Variational AutoEncoders (VAEs)
    - Normalizing Flows (NFs)
2. Generative Adversarial Networks (GANs) (**AL**, 2 courses)
    - Training of GANs
    - Link with optimal transport
3. Application of generative models for imaging inverse problems (**AL**, 1 course)
    - Classical imaging problems: Super-Resolution, inpainting,...
    - Image-to-image translations

Generative models for images: Plan of the course 2025-26

Part II: Plug-and-Play methods for imaging inverse problems
(**AL**, 2 courses)

1. Proximal splitting and PnP methods
2. Convergence of PnP methods via fixed point
    - Fixed point theorem for “averaged” operators
    - Convergence of PnP with an “averaged” denoiser
    - Learning a denoiser with Lipschitz constraint
3. Convergence of PnP methods via non-convex optimization
    - Convergence of proximal splitting in the non-convex case
    - Gradient-step regularization, and convergence of PnP
    - Applications to several inverse problems

Generative models for images: Plan of the course 2025-26

Part III: Diffusion or Score-Based Generative Models (SGM)
(**BG**, 2 courses)

1. Diffusion models in pixel space
    - Time reversal of stochastic processes
    - Denoising Diffusion Probabilistic Models (DDPM)
    - Denoising Implicit Diffusion Models (DDIM)
    - Application to imaging inverse problems
2. Latent Diffusion Models (LDM)
    - Stable diffusion pipeline
    - Text-to-image synthesis
    - Controllable diffusion

Some courses are practice sessions. Bring your laptop!

Generative models for images: Validation

Validation (Master MVA)

- Un devoir sur table rapide (30 min) en février.
- Un projet final.
- Merci de vous inscrire via le google form.
- Email: generative.modeling.mva@gmail.com

Projet final : Détails

- Etude d'un article en groupe de 2 étudiants
- Rendu de rapport et soutenance orale à Telecom Paris (semaine du 30 mars 2026)
- Choix d'un article parmi une liste proposée
- Discussion critique de l'article
- Comparaison avec les méthodes vues en cours
- Expériences numériques et comparaison avec les résultats vus en TP

Inverse Problems

Inverse problem with additive noise:

$$v = \mathcal{A}u_0 + w$$

where

- $u_0 \in \mathbb{R}^d$ is the clean image to recover
- $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^m$
- w is a noise

In many cases, the degradation operator $\mathcal{A}$ can be approximated with a linear operator A, and the noise model w is assumed to be Gaussian.

But, there are also inverse problems with non-linear $\mathcal{A}$ and non-Gaussian noise (e.g. Poisson noise).

Classical Inverse Problems

Application	Forward model	Notes
Denoising [58]	$A = I$	I is the identity matrix
Deconvolution [58,59]	$\mathcal{A}(\boldsymbol{x}) = \boldsymbol{h} * \boldsymbol{x}$	$\boldsymbol{h}$ is a known blur kernel and $*$ denotes convolution. When $\boldsymbol{h}$ is unknown the reconstruction problem is known as blind deconvolution.
Superresolution [60,61]	$A = SB$	S is a subsampling operator (identity matrix with missing rows) and B is a blurring operator cooresponding to convolution with a blur kernel
Inpainting [62]	$A = S$	S is a diagonal matrix where $S_{i,i} = 1$ for the pixels that are sampled and $S_{i,i} = 0$ for the pixels that are not.
Compressive Sensing [63,64]	$A = SF$ or $A =$ Gaussian or Bernoulli ensemble	S is a subsampling operator (identity matrix with missing rows) and F discrete Fourier transform matrix.
MRI [3]	$A = SFD$	S is a subsampling operator (identity matrix with missing rows), F is the discrete Fourier transform matrix, and D is a diagonal matrix representing a spatial domain multiplication with the coil sensitivity map (assuming a single coil aquisition with Cartesian sampling in a SENSE framework [65]).
Computed tomography [58]	$A = R$	R is the discrete Radon transform [66].
Phase Retrieval [67–70]	$\mathcal{A}(\boldsymbol{x}) = \lvert A\boldsymbol{x}\rvert^2$	$\lvert\cdot\rvert$ denotes the absolute value, the square is taken elementwise, and A is a (potentially complex-valued) measurement matrix that depends on the application. The measurement matrix A is often a variation on a discrete Fourier transform matrix.

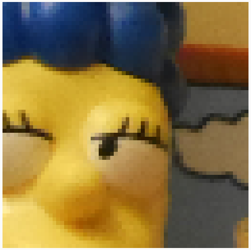
(source: (Ongie et al., 2020))

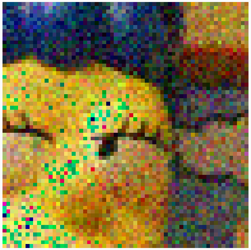
Gaussian denoising

Let's start with the case $A = \mathrm{Id}$, i.e. **image denoising**:

$$v = u_0 + w \quad \text{where} \quad w \sim \mathcal{N}(0, \sigma^2 \mathrm{Id}).$$

- We want to estimate u_0 from a single realization of v.
- We need to add some prior knowledge on the solution (e.g. regularity assumption).


u_0

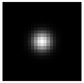

v

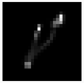
Deblurring

- A spatially invariant blur can be modeled by a convolution operator $Au = k * u$
- Several types of blur exist (motion, defocus)
- **Non-blind deblurring** consists in recovering u_0 from

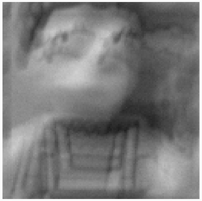
$$v = k * u_0 + w.$$

- Non-blind means that the blur kernel k is known.
- We won't tackle blind deblurring here.


Isotropic blur


Motion blur

Original


Blurred

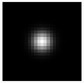
Deblurring

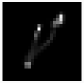
- A spatially invariant blur can be modeled by a convolution operator $Au = k * u$
- Several types of blur exist (motion, defocus)
- **Non-blind deblurring** consists in recovering u_0 from

$$v = k * u_0 + w.$$

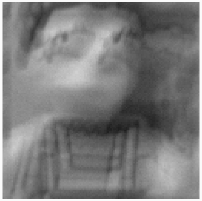
- Non-blind means that the blur kernel k is known.
- We won't tackle blind deblurring here.

Which blur kernel was used ?


Isotropic blur


Motion blur

Original


Blurred

Super-Resolution

Super-resolution consists in finding another version of v at higher resolution.

This is an inverse problem involving the subsampling operator with stride $s \in \mathbb{N}^*$:

$$u_{\downarrow s}(x, y) = u(sx, sy).$$

In practice, to create a low resolution version of an image we need to apply an (*anti-aliasing*) filter before subsampling!

With prefiltering, we obtain the zoom-out operator

$$Au = (k * u)_{\downarrow s}.$$

Super-resolution consists in recovering u_0 from

$$v = (k * u)_{\downarrow s} + w.$$

The degraded image v is defined on a subgrid of stride s.

Inpainting

Inpainting consists in filling missing regions in images

The degradation operator then writes

$$Au = u\mathbf{1}_\omega$$

where $\omega \subset \Omega$ is the set of known pixels and $\Omega \setminus \omega$ the mask.

Inverse problem

We wish to recover u_0 from

$$v = Au_0 + w.$$

The problem is said ill-posed when A is not invertible or with unstable inverse.

Example : For deblurring, $Au = k * u$, we can invert A directly in Fourier domain:

Inverse problem

We wish to recover u_0 from

$$v = Au_0 + w.$$

The problem is said ill-posed when A is not invertible or with unstable inverse.

Example : For deblurring, $Au = k * u$, we can invert A directly in Fourier domain:

$$u = \mathcal{F}^{-1}\left(\frac{\hat{v}}{\hat{k}}\right) = \mathcal{F}^{-1}\left(\hat{u_0} + \frac{\hat{w}}{\hat{k}}\right) \quad \longrightarrow \quad \text{but noise explodes !}$$

Inverse problem

We wish to recover u_0 from

$$v = Au_0 + w.$$

The problem is said ill-posed when A is not invertible or with unstable inverse.

Example : For deblurring, $Au = k * u$, we can invert A directly in Fourier domain:

$$u = \mathcal{F}^{-1}\left(\frac{\hat{v}}{\hat{k}}\right) = \mathcal{F}^{-1}\left(\hat{u_0} + \frac{\hat{w}}{\hat{k}}\right) \quad \longrightarrow \quad \text{but noise explodes !}$$

When the problem is ill-posed, there may be multiple solutions or erroneous solutions.

It is thus useful to adopt an *a priori* on the solution, e.g. imposing some kind of regularity.

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2}\|Au - v\|_2^2 + \lambda R(u)$$

where $R(u)$ imposes some kind of regularity of u, and $\lambda \geq 0$ is a parameter.

The problem $\operatorname{argmin}_{u\in\mathbb{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Simple (nearly useless) regularization: Consider $R(u) = \frac{\lambda}{2}\|u\|_2^2$. Then $u_\lambda \in \operatorname{argmin}_F$ is given by

$$A^T(Au_\lambda - v) + \lambda u_\lambda = 0 \quad \text{i.e.} \quad u_\lambda = (A^TA + \lambda I)^{-1}A^T v$$

Example: for denoising ($A = \text{Id}$),

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2}\|Au - v\|_2^2 + \lambda R(u)$$

where $R(u)$ imposes some kind of regularity of u, and $\lambda \geq 0$ is a parameter.

The problem $\mathrm{argmin}_{u\in\mathbb{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Simple (nearly useless) regularization: Consider $R(u) = \frac{\lambda}{2}\|u\|_2^2$. Then $u_\lambda \in \mathrm{argmin}_F$ is given by

$$A^T(Au_\lambda - v) + \lambda u_\lambda = 0 \quad \text{i.e.} \quad u_\lambda = (A^TA + \lambda I)^{-1}A^Tv$$

Example: for denoising ($A = \mathrm{Id}$), it just divides all values by $1 + \lambda$...

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2}\|Au - v\|_2^2 + \lambda R(u)$$

where $R(u)$ imposes some kind of regularity of u, and $\lambda \geq 0$ is a parameter.

The problem $\operatorname{argmin}_{u\in\mathbb{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Simple (nearly useless) regularization: Consider $R(u) = \frac{\lambda}{2}\|u\|_2^2$. Then $u_\lambda \in \operatorname{argmin}_F$ is given by

$$A^T(Au_\lambda - v) + \lambda u_\lambda = 0 \quad \text{i.e.} \quad u_\lambda = (A^TA + \lambda I)^{-1}A^Tv$$

Example: for denoising ($A = \text{Id}$), it just divides all values by $1+\lambda$...

For differentiable F, we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2}\|Au - v\|_2^2$ is

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2}\|Au - v\|_2^2 + \lambda R(u)$$

where $R(u)$ imposes some kind of regularity of u, and $\lambda \geq 0$ is a parameter.

The problem $\operatorname{argmin}_{u\in\mathbb{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Simple (nearly useless) regularization: Consider $R(u) = \frac{\lambda}{2}\|u\|_2^2$. Then $u_\lambda \in \operatorname{argmin}_F$ is given by

$$A^T(Au_\lambda - v) + \lambda u_\lambda = 0 \quad \text{i.e.} \quad u_\lambda = (A^T A + \lambda I)^{-1} A^T v$$

Example: for denoising ($A = \mathrm{Id}$), it just divides all values by $1 + \lambda$...

For differentiable F, we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2}\|Au - v\|_2^2$ is $\nabla f(u) = A^T(Au - v)$.

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2}\|Au - v\|_2^2 + \lambda R(u)$$

where $R(u)$ imposes some kind of regularity of u, and $\lambda \geq 0$ is a parameter.

The problem $\operatorname{argmin}_{u \in \mathbb{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Simple (nearly useless) regularization: Consider $R(u) = \frac{\lambda}{2}\|u\|_2^2$. Then $u_\lambda \in \operatorname{argmin}_F$ is given by

$$A^T(Au_\lambda - v) + \lambda u_\lambda = 0 \quad \text{i.e.} \quad u_\lambda = (A^TA + \lambda I)^{-1}A^Tv$$

Example: for denoising ($A = \text{Id}$), it just divides all values by $1 + \lambda$...

For differentiable F, we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2}\|Au - v\|_2^2$ is $\nabla f(u) = A^T(Au - v)$.

If $Au = k * u$ (periodic convolution), then $A^Tu =$

Image Restoration by Optimization

We will therefore try to solve

$$F(u) = \frac{1}{2}\|Au - v\|_2^2 + \lambda R(u)$$

where $R(u)$ imposes some kind of regularity of u, and $\lambda \geq 0$ is a parameter.

The problem $\operatorname{argmin}_{u\in\mathbb{R}^\Omega} F(u)$ is very high-dimensional, and we need efficient algorithms.

Simple (nearly useless) regularization: Consider $R(u) = \frac{\lambda}{2}\|u\|_2^2$. Then $u_\lambda \in \operatorname{argmin}_F$ is given by

$$A^T(Au_\lambda - v) + \lambda u_\lambda = 0 \quad \text{i.e.} \quad u_\lambda = (A^TA + \lambda I)^{-1}A^Tv$$

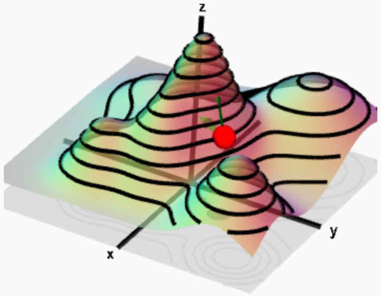
Example: for denoising ($A = \mathrm{Id}$), it just divides all values by $1+\lambda$...

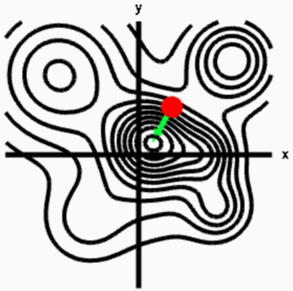
For differentiable F, we can always consider simple gradient descent.

Example: The gradient of $f(u) = \frac{1}{2}\|Au - v\|_2^2$ is $\nabla f(u) = A^T(Au - v)$.

If $Au = k * u$ (periodic convolution), then $A^Tu = \tilde{k} * u$ with $\tilde{k}(\mathbf{x}) = \overline{k(-\mathbf{x})}$.

The Steepest Descent





https://mathinsight.org/directional_derivative_gradient_introduction

Descent Lemma

Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with L-Lipschitz gradient. Then, for any $x, y \in \mathbb{R}^d$,

$$\begin{aligned}
f(y) &= f(x) + \int_0^1 \nabla f(x + t(y-x)) \cdot (y-x) dt \\
&= f(x) + \nabla f(x) \cdot (y-x) + \int_0^1 \big(\nabla f(x + t(y-x)) - \nabla f(x)\big) \cdot (y-x) dt \\
&\leq f(x) + \nabla f(x) \cdot (y-x) + \int_0^1 \|\nabla f(x + t(y-x)) - \nabla f(x)\| \|y-x\| dt \\
&\leq f(x) + \nabla f(x) \cdot (y-x) + \int_0^1 Lt\|y-x\|^2 dt \\
&\leq f(x) + \nabla f(x) \cdot (y-x) + \frac{L}{2}\|y-x\|^2.
\end{aligned}$$

Consequence: If we choose $\tau \in [0, \frac{2}{L}]$, then

$$f(x - \tau \nabla f(x)) \leq f(x) - \tau\left(1 - \frac{\tau L}{2}\right)\|\nabla f(x)\|^2 \leq f(x).$$

Gradient Descent

We consider here the gradient descent method:

$$x_{n+1} = x_n - \tau_n \nabla f(x_n) \ ,$$

where $\tau_n > 0$ is a sequence of step sizes.

- For $\tau_n = \tau$ constant, we speak of fixed step size.
- We speak of optimal step size if, at each iteration n, we choose

$$\tau_n \in \operatorname{argmin}_{t\in\mathbb{R}} f(x_n - t\nabla f(x_n)).$$

The descent lemma gives that for f differentiable with L-Lipschitz gradient and $\underline{\tau \leqslant \frac{2}{L}}$,

$$f(x_{n+1}) \leqslant f(x_n)$$

Thus, if f is lower bounded, $f(x_n)$ converges.

Convexity and Minimum

The function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^d$,

$$\forall t \in (0,1), \quad f((1-t)x + ty) \leqslant (1-t)f(x) + tf(y).$$

It is said strictly convex if the inequality is strict.

If f is convex and differentiable, one can show that for any $x, y \in \mathbb{R}^d$,

$$f(y) \geqslant f(x) + \nabla f(x) \cdot (y - x).$$

Consequence : If f is convex and differentiable, then

$$x \in \operatorname{argmin} f \quad \Longleftrightarrow \nabla f(x) = 0.$$

The argmin is unique as soon as f is strictly convex.

Strong Convexity

We say that f is α-convex (with $\alpha \in \mathbb{R}$) if $f - \frac{\alpha}{2}\|\cdot\|^2$ is convex.

When $\alpha > 0$, we say that f is **strongly convex**.

Remark : The convexity and the gradient Lipschitz constant can be read on the Hessian:

If $A, B \in \mathbb{R}^{d\times d}$ are symmetric, we write $A \succeq B$ if $A - B$ if semi-definite positive, i.e.

$$\forall x \in \mathbb{R}^d, \quad Ax\cdot x \geq Bx \cdot x.$$

For $f : \mathbb{R}^d \to \mathbb{R}$ of class $\mathscr{C}^2$,

∇f is L-Lipschitz iff $\quad \forall x \in \mathbb{R}^d,\ -L\mathrm{Id} \preceq \nabla^2 f(x) \preceq L\mathrm{Id}$.
i.e. $\forall x$ the eigenvalues of $\nabla^2 f(x)$ have moduli $\leq L$.

f is α-convex iff $\quad \forall x \in \mathbb{R}^d,\ \nabla^2 f(x) \succeq \alpha\mathrm{Id}$
i.e. $\forall x$ the eigenvalues of $\nabla^2 f(x)$ are all $\geq \alpha$.

Convergence Guarantees, Convex Case

Theorem
Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex differentiable with ∇f L-Lipschitz. Assume that $\operatorname{argmin} f$ is non-empty.

Let $\tau \in (0, \frac{2}{L})$, $x_0 \in \mathbb{R}^d$ and (x_n) the sequence defined by

$$x_{n+1} = x_n - \tau \nabla f(x_n) \ .$$

Then (x_n) converges towards an element of $\operatorname{argmin} f$.

Theorem
Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and α-strongly convex with L-Lipschitz gradient.

Then there exists a unique $x_* \in \operatorname{argmin} f$, and for $\tau < \frac{1}{L} \leqslant \frac{1}{\alpha}$, we have

$$\|x_n - x_*\|^2 \leqslant (1 - \tau\alpha)^n \|x_0 - x_*\|^2.$$

Optimization for Inverse Problems

To solve the inverse problem $v = Au_0 + w$, we can thus minimize

$$F(u) = f(u) + g(u)$$

with $f(u) = \frac{1}{2}\|Au - v\|^2$ and $g(u) = \lambda R(u)$, $\lambda > 0$.

Consider here $R(u) = \frac{1}{2}\|Bu\|_2^2$ with $B \in \mathbb{R}^{p\times d}$, F is convex and differentiable.

Solutions are characterized by $\nabla F(u) = 0$ i.e. $A^T(Au - v) + \lambda B^T Bu = 0$.

Also, we can minimize F by gradient descent with $\tau < \frac{2}{L}$ where $L = \|A^TA + \lambda B^TB\|$.

- For $Au = k * u$, $A^TAu = \mathcal{F}^{-1}(|\hat{k}|^2\hat{u})$.
  If $|\hat{k}| \leq 1$, it follows that $\|A^TA\| \leq 1$.
- For $Au = \mathbf{1}_\omega u$, $A^TA = A^2 = A$ and $\|A\| = 1$.

Optimization for Inverse Problems

To solve the inverse problem $v = Au_0 + w$, we can thus minimize

$$F(u) = f(u) + g(u)$$

with $f(u) = \frac{1}{2}\|Au - v\|^2$ and $g(u) = \lambda R(u)$, $\lambda > 0$.

Consider here $R(u) = \frac{1}{2}\|Bu\|_2^2$ with $B \in \mathbb{R}^{p \times d}$, F is convex and differentiable.

Solutions are characterized by $\nabla F(u) = 0$ i.e. $A^T(Au - v) + \lambda B^T Bu = 0$.

Also, we can minimize F by gradient descent with $\tau < \frac{2}{L}$ where $L = \|A^T A + \lambda B^T B\|$.

- For $Au = k * u$, $A^T Au = \mathcal{F}^{-1}(|\hat{k}|^2 \hat{u})$.
  If $|\hat{k}| \leq 1$, it follows that $\|A^T A\| \leq 1$.
- For $Au = \mathbf{1}_\omega u$, $A^T A = A^2 = A$ and $\|A\| = 1$.

Good news: By automatic differentiation you need only coding $F(u)$...

Optimization for Inverse Problems

To solve the inverse problem $v = Au_0 + w$, we can thus minimize

$$F(u) = f(u) + g(u)$$

with $f(u) = \frac{1}{2}\|Au - v\|^2$ and $g(u) = \lambda R(u)$, $\lambda > 0$.

Consider here $R(u) = \frac{1}{2}\|Bu\|_2^2$ with $B \in \mathbb{R}^{p\times d}$, F is convex and differentiable.

Solutions are characterized by $\nabla F(u) = 0$ i.e. $A^T(Au - v) + \lambda B^T Bu = 0$.

Also, we can minimize F by gradient descent with $\tau < \frac{2}{L}$ where $L = \|A^TA + \lambda B^TB\|$.

- For $Au = k * u$, $A^TAu = \mathcal{F}^{-1}(|\hat{k}|^2\hat{u})$.
  If $|\hat{k}| \leq 1$, it follows that $\|A^TA\| \leq 1$.
- For $Au = \mathbf{1}_\omega u$, $A^TA = A^2 = A$ and $\|A\| = 1$.

Good news: By automatic differentiation you need only coding $F(u)$...

But ! in order to avoid instability problems, you'd better know what F does...

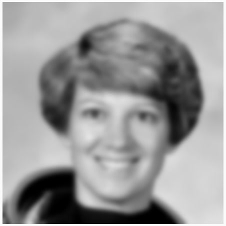
Let us start with zero regularization!

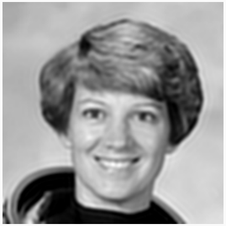
Consider here
$$f(u) = \frac{1}{2}\|Au - v\|^2.$$

- We have an orthogonal decomposition $\mathbb{R}^d = K \oplus K^\perp$ with $K = \mathrm{Ker}[A]$ and $K^\perp = \mathrm{Im}[A^T]$
- Therefore $\mathrm{argmin}_{\mathbb{R}^d} f$ is non-empty and we can define
$$A^+ v = \min_{u \in \mathrm{argmin} f} \|u\|_2^2.$$
It defines a linear operator A^+, called Moore-Penrose pseudo-inverse.
- The Moore-Penrose pseudo-inverse has a zero component in $\mathrm{Ker}[A]$.
- $A_{K^\perp} : K^\perp \to \mathrm{Im}(A)$ is invertible. Thus $A^+ = A_{K^T}^{-1} P$ (with P the orthogonal projection on $\mathrm{Im}(A^T)$).
- Actually, one can show that $A^+ v = \lim_{\lambda \to 0} (A^T A + \lambda I)^{-1} A^T v$.
- Gradient descent on $f(u)$ converges to $A^+ v$, as soon as initialization has null component on K.
- But $A^+ v$ is generally a bad solution for inverse problems because of bad conditioning.

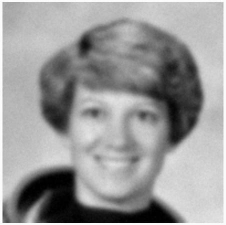
Gaussian deblurring and pseudo-inverse

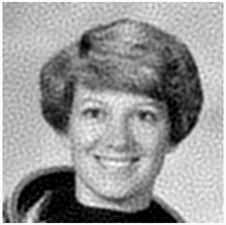
Original


Blurred
no noise


Pseudo-inverse

Original


Blurred
noise $\sigma = 4$


Pseudo-inverse

Explicit Regularizations

We define the discrete derivatives of u by

$$\nabla u(x,y) = \begin{pmatrix} \partial_1 u(x,y) \\ \partial_2 u(x,y) \end{pmatrix} \quad \text{avec} \quad \begin{cases} \partial_1 u(x,y) = d_1 * u(x,y) = u(x+1,y) - u(x,y) \\ \partial_2 u(x,y) = d_2 * u(x,y) = u(x,y+1) - u(x,y) \end{cases} .$$

We define **Tychonov regularization** by

$$\|\nabla u\|_2^2 = \sum_{\mathbf{x}\in\Omega} \|\nabla u(\mathbf{x})\|^2 = \sum_{\mathbf{x}\in\Omega} |\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2.$$

We define the **total variation** by

$$\mathsf{TV}(u) = \|\nabla u\|_1 = \sum_{\mathbf{x}\in\Omega} \|\nabla u(\mathbf{x})\| = \sum_{\mathbf{x}\in\Omega} \sqrt{|\partial_1 u(\mathbf{x})|^2 + |\partial_2 u(\mathbf{x})|^2}.$$

Back to denoising

Let us minimize

$$F(u) = \frac{1}{2}\|u - v\|^2 + \lambda R(u)$$

where R is a regularization and $\lambda > 0$.

Consider first Tychonov regularization $R(u) = \frac{1}{2}\|\nabla u\|_2^2$**.**

Back to denoising

Let us minimize

$$F(u) = \frac{1}{2}\|u - v\|^2 + \lambda R(u)$$

where R is a regularization and $\lambda > 0$.

Consider first Tychonov regularization $R(u) = \frac{1}{2}\|\nabla u\|_2^2$.

We have $\nabla R(u) = \nabla^T \nabla u$.

Back to denoising

Let us minimize

$$F(u) = \frac{1}{2}\|u - v\|^2 + \lambda R(u)$$

where R is a regularization and $\lambda > 0$.

Consider first Tychonov regularization $R(u) = \frac{1}{2}\|\nabla u\|_2^2$**.**

We have $\nabla R(u) = \nabla^T \nabla u$. As F is convex,

$$u \in \operatorname{argmin} F \iff \nabla F(u) = 0 \iff u - v + \lambda \nabla^T \nabla u = 0 \iff u = (I + \lambda \nabla^T \nabla)^{-1} v$$

Back to denoising

Let us minimize

$$F(u) = \frac{1}{2}\|u - v\|^2 + \lambda R(u)$$

where R is a regularization and $\lambda > 0$.

Consider first Tychonov regularization $R(u) = \frac{1}{2}\|\nabla u\|_2^2$**.**

We have $\nabla R(u) = \nabla^T \nabla u$. As F is convex,

$$u \in \operatorname{argmin} F \iff \nabla F(u) = 0 \iff u - v + \lambda \nabla^T \nabla u = 0 \iff u = (I + \lambda \nabla^T \nabla)^{-1} v$$

For $p : \Omega \to \mathbb{R}^2$, $\nabla^T p$ is given by

$$\nabla^T p(x, y) = p_1(x - 1, y) - p_1(x, y) + p_2(x, y - 1) - p_2(x, y).$$

Actually, $\operatorname{div}(p) := -\nabla^T p$ is a discrete divergence and $\Delta u := -\nabla^T \nabla u$ is a discrete Laplacian.

Explicit Solution: Wiener filtering

Theorem

Let $v \in \mathbb{C}^\Omega$ *and* $\lambda > 0$*. The function* $F : \mathbb{C}^\Omega \to \mathbb{R}_+$ *defined by*

$$\forall u \in \mathbb{C}^\Omega, \quad F(u) = \frac{1}{2}\|u - v\|_2^2 + \frac{\lambda}{2}\|\nabla u\|_2^2$$

has a minimum attained at a unique $u_* \in \mathbb{C}^\Omega$*, which is given in Fourier domain by:*

$$\forall(\xi, \zeta) \in \Omega, \quad \hat{u}_*(\xi, \zeta) = \frac{\hat{v}(\xi, \zeta)}{1 + \lambda\, \hat{L}(\xi, \zeta)}$$

where $\hat{L}(\xi, \zeta) = |\hat{d}_1(\xi, \zeta)|^2 + |\hat{d}_2(\xi, \zeta)|^2 = 4\left(\sin^2\left(\frac{\pi\xi}{M}\right) + \sin^2\left(\frac{\pi\zeta}{N}\right)\right)$.

Remarks:

- d_1, d_2 are the kernel derivatives, e.g. $d_1 = \delta_{(-1,0)} - \delta_{(0,0)}$. So $\hat{L}$ is the kernel of $-\Delta$ filter.
- The theorem adapts for deblurring with Tychonov regularization:

$$\forall(\xi, \zeta) \in \Omega, \quad \hat{u}_*(\xi, \zeta) = \frac{\overline{\widehat{k}(\xi, \zeta)}\hat{v}(\xi, \zeta)}{|\widehat{k}(\xi, \zeta)|^2 + \lambda\, \hat{L}(\xi, \zeta)}$$

Link with an evolution model

The gradient descent on

$$F(u) = \frac{1}{2}\|u - v\|_2^2 + \frac{\lambda}{2}\|\nabla u\|_2^2$$

writes as

$$u_{n+1} - u_n = -\tau(u_n - v) + \lambda\tau\Delta u_n \ .$$

The sequence (u_n) converges to u_* as soon as $\tau < \frac{2}{L}$ with $L = \|I + \lambda\nabla^T\nabla\| = 1 + 8\lambda$.

Link with an evolution model

The gradient descent on

$$F(u) = \frac{1}{2}\|u - v\|_2^2 + \frac{\lambda}{2}\|\nabla u\|_2^2$$

writes as

$$u_{n+1} - u_n = -\tau(u_n - v) + \lambda\tau\Delta u_n \ .$$

The sequence (u_n) converges to u_* as soon as $\tau < \frac{2}{L}$ with $L = \|I + \lambda\nabla^T\nabla\| = 1 + 8\lambda$.

If we drop the data-fidelity... then gradient descent on $u \mapsto \frac{1}{2}\|\nabla u\|_2^2$ gives

$$u_{n+1} - u_n = \tau\Delta u_n$$

This is a discretization of the heat equation $\partial_t u = c\Delta u$ with initial condition u_0.

Smoothed Total Variation

What if we want to minimize

$$F(u) = \frac{1}{2}\|u - v\|_2^2 + \lambda \mathsf{TV}(u).$$

Problem: The total variation is not differentiable.

A simple solution: consider a smoothed variant: For $\varepsilon > 0$, let

$$\mathsf{TV}_\varepsilon(u) = \sum_{(x,y)\in\Omega} \sqrt{\varepsilon^2 + \partial_1 u(x,y)^2 + \partial_2 u(x,y)^2}\ .$$

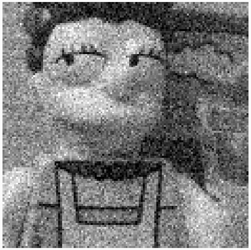
One can see that

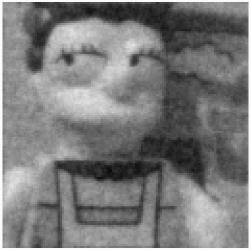
$$\nabla \mathsf{TV}_\varepsilon(u) = \nabla^T \left(\frac{\nabla u}{\sqrt{\varepsilon^2 + \|\nabla u\|_2^2}} \right).$$

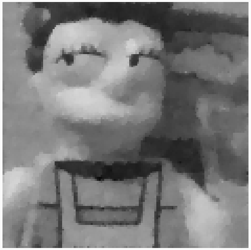
And one can show that $\nabla \mathsf{TV}_\varepsilon$ is $\frac{8}{\varepsilon}$-Lipschitz.

We can thus minimize F by gradient descent with $\tau < \frac{2}{1+\frac{8\lambda}{\varepsilon}}$.

Denoising Examples


Noisy
PSNR = 19.93


Tychonov denoising
PSNR = 25.89


TV$_\varepsilon$ denoising
PSNR = 27.21

Projected Gradient Descent

Imagine that we want to constrain the solution into a convex closed set $C \subset \mathbb{R}^d$:

$$\operatorname{argmin}_{u \in C} F(u)$$

For that, we can use the orthogonal projection $p_C : \mathbb{R}^d \to C$.

Theorem
Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex differentiable such that ∇f is L-Lipschitz.

Let $C \subset \mathbb{R}^d$ be a closed convex set. Assume that $\operatorname{argmin}_C f$ is non-empty.

For $\tau \in (0, \frac{2}{L})$, $x_0 \in \mathbb{R}^d$, let (x_n) be defined by

$$x_{n+1} = p_C(x_n - \tau \nabla f(x_n)) \; .$$

Then (x_n) converges to an element of $\operatorname{argmin}_C f$.

Example : For inpainting, we can deal with the noiseless problem $v = Au$.

In this case, we can perform constrained minimization of only the regularization term:

$$\min_{v=Au} R(u).$$

Metrics for Inverse Problems

Euclidean metrics

- Given two images u and v of size $M \times N$ with graylevels between 0 and 255.
- Denote $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$ the pixel domain
- Mean Square Error $\downarrow$:

$$\text{MSE}(u,v) = \frac{1}{MN} \sum_{\mathbf{x} \in \Omega} (u(\mathbf{x}) - v(\mathbf{x}))^2$$

- Root Mean Square Error $\downarrow$:

$$\text{RMSE}(u,v) = \left(\frac{1}{MN} \sum_{\mathbf{x} \in \Omega} (u(\mathbf{x}) - v(\mathbf{x}))^2 \right)^{\frac{1}{2}}$$

- Peak Signal to Noise Ratio $\uparrow$:

$$\text{PSNR}(u,v) = 20 \log_{10} \left(\frac{\text{MAX}}{\text{RMSE}(u,v)} \right) \quad (\text{where MAX} = 255)$$

- Useful for inverse problems such as denoising.
- Not ideal when one hopes to generate new content.

Structural similarity index measure (SSIM $\uparrow$) (Wang et al., 2004)

Between patches:

- Given two patches x, y (typically of size 8×8 or 11×11 with a Gaussian windowing)

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \in [-1, 1]$$

  with:
  - μ_x the pixel sample mean of x
  - μ_y the pixel sample mean of y
  - σ_x^2 the variance of x
  - σ_y^2 the variance of y
  - σ_{xy} the covariance of x and y
  - $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator, with the range $L = 255$ or 1 and $k_1 = 0.01$ and $k_2 = 0.03$ by default.
- $\mathrm{SSIM}(x,y)$ is the product of three terms:

Luminance	Contrast	Structure
$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$	$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$	$s(x,y) = \frac{\sigma_{xy} + c_2/2}{\sigma_x\sigma_y + c_2/2}$

Structural similarity index measure (SSIM $\uparrow$) (Wang et al., 2004)

Between images:

- Given two images u and v of size $M \times N$ with graylevels between 0 and 255, define the Mean-SSIM by averaging over all patches:

$$\text{(M)SSIM}(u,v) = \text{mean}(\{\text{SSIM}(P_{\mathbf{x}}(u), P_{\mathbf{x}}(v)),\ \mathbf{x}+\omega \subset \Omega\})$$

  where $P_{\mathbf{x}}(u)$ is the restriction of u on the patch $\mathbf{x}+\omega$.
- There are also multiscale variants.
- SSIM is not a distance, its range is $[-1, 1]$.
- SSIM is closer to a perceptual distance, especially regarding local textures.

Structural similarity index measure (SSIM)

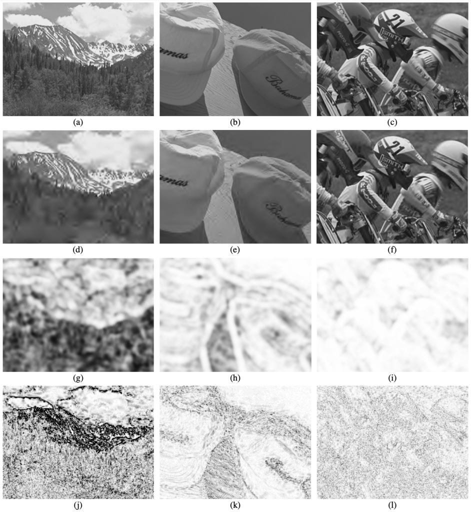


Fig. 7. Sample JPEG2000 images compressed to different quality levels (original size: 768×512; cropped to 256×192 for visibility). (a), (b) and (c) are the original "Stream", "Caps" and "Bikes" images, respectively. (d) Compressed to 0.1896 bits/pixel, PSNR = 23.46dB, MSSIM = 0.7339; (e) Compressed to 0.1982 bits/pixel, PSNR = 34.56dB, MSSIM = 0.9409; (f) Compressed to 1.1454 bits/pixel, PSNR = 33.47dB, MSSIM = 0.9747. (g), (h) and (i) show SSIM maps of the compressed images, where brightness indicates the magnitude of the local SSIM index (squared for visibility). (j), (k) and (l) show absolute error maps of the compressed images (contrast-inverted for easier comparison to the SSIM maps).

(source: From (Wang et al., 2004))

LPIPS $\downarrow$ (Zhang et al., 2018)

LPIPS: Learned Perceptual Image Patch Similarity

- Previous works on texture synthesis (Gatys et al., 2015) and style transfer (Gatys et al., 2016) (Johnson et al., 2016) have shown the importance of the VGG (Simonyan and Zisserman, 2015) features for perceptual similarity between images.
- This means that intermediate features of classification CNN are useful in their own: "**a good feature is a good feature**. Features that are good at semantic tasks are also good at self-supervised and unsupervised tasks, and also provide good models of both human perceptual behavior and macaque neural activity." (Zhang et al., 2018)

LPIPS $\downarrow$ (Zhang et al., 2018)

LPIPS: Learned Perceptual Image Patch Similarity

LPIPS model: Define a perceptual distance between 64×64 patches by computing a Euclidean norm between features:

$$\text{LPIPS}(u, u_0)^2 = \sum_{\text{layers } \ell} \frac{1}{H_\ell W_\ell} \sum_{i,j} \| w_\ell \odot (F^\ell(u)_{i,j} - F^\ell(u_0)_{i,j}) \|_2^2$$

where for each layer ℓ, the neural response $F^\ell(u)$ is weighted by channel weights $w_\ell \in \mathbb{R}^{C_\ell}$ that are learned to reproduce human evaluation of distortion between patches.

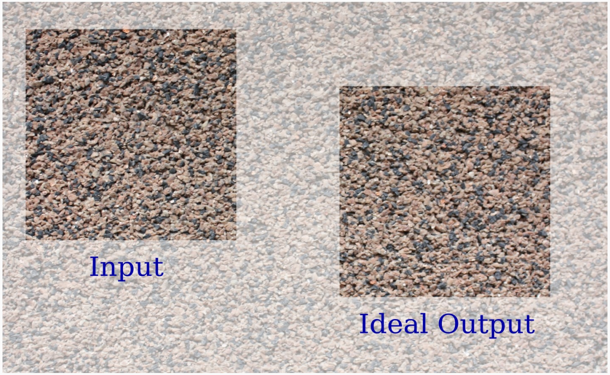
Textures synthesis using CNN statistics

Gatys et al algorithm

References: L. Gatys, A. S. Ecker, and M. Bethge, *Texture synthesis using convolutional neural networks*, in Advances in Neural Information Processing Systems, 2015

Texture synthesis

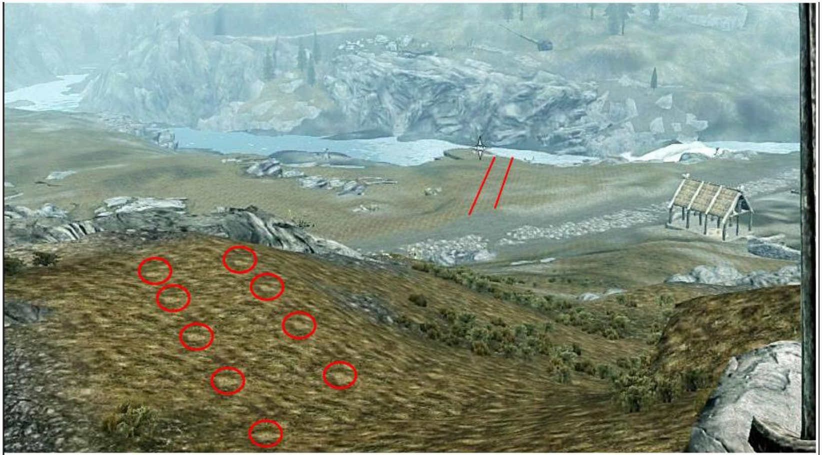
Texture Synthesis: Given an input texture image, produce an output texture image being both visually similar to and pixel-wise different from the input texture.



The output image should ideally be perceived as another part of the same large piece of homogeneous material the input texture is taken from.

Texture synthesis: Motivation

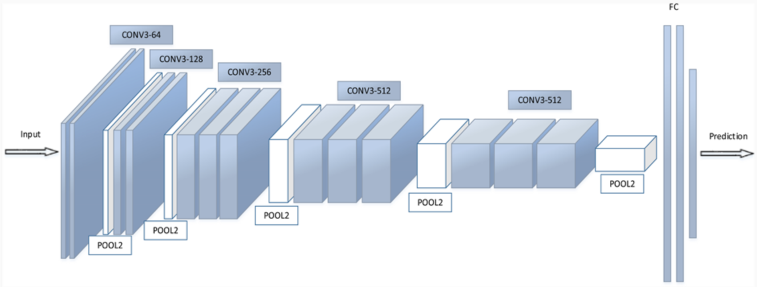
- Important problem in the industry of virtual reality (video games, movies, special effects,. . .).
- Periodic repetition is not satisfying !



2011: Skyrim (Bethesda) *screenshot from Three Parts Theory*

Convolutional Neural Networks (CNN)

- Main idea: Use the feature layers of a trained deep CNN, namely VGG 19 (Simonyan and Zisserman, 2015), as statistics.
- VGG 19 was trained for image classification.
- It only uses 3×3 convolution kernels followed by RELU (= positive part) and max-pooling.



- We do not use the last “fully connected” layers that perform classification.
- VGG 19 is understood as a multiscale nonlinear transform adapted to natural images.

Gatys et al algorithm

Given an example image u and a random initialization x_0, one optimizes the loss function

$$E(x) = \sum_{\text{for selected layers } L} w_L \left\| G^L(x) - G^L(u) \right\|_F^2$$

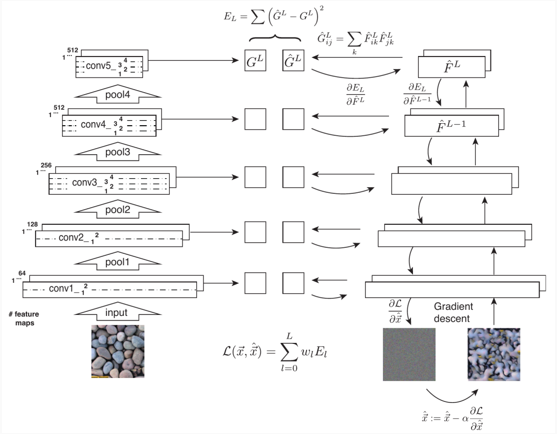
where

- w_L is a weight parameter for each layer
- $\| \cdot \|_F$ is the Frobenius norm
- for an image y and a layer index L, $G^L(y)$ denotes the **Gram matrix** of the VGG-19 features at layer L: if $V^L(y)$ is the feature response of y at layer L that has spatial size $w \times h$ and n channels,

$$G^L(y) = \frac{1}{wh} \sum_{k \in \{0,\dots,w-1\} \times \{0,\dots,h-1\}} V^L(y)_k V^L(y)_k^T \in \mathbb{R}^{n \times n}.$$

The Gram matrix is a spatial statistics of order 2 that contains mean and covariance information.

Gatys et al algorithm



$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
conv5_ 3 4 2 1
512
pool4
conv4_ 3 4 2 1
512
pool3
conv3_ 3 4 2 1
256
pool2
conv2_ 1 2
128
pool1
conv1_ 1 2
64
input
# feature maps
G^L
$\hat{G}^L$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$
Gradient descent
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$
$\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$

- The gradient of the energy is computed using back-propagation routines.
- The authors use a quasi-Newton algorithm: L-BFGS that stands for Limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (low-rank approximation of the Hessian matrix for computing the descent direction).

Gatys et al algorithm

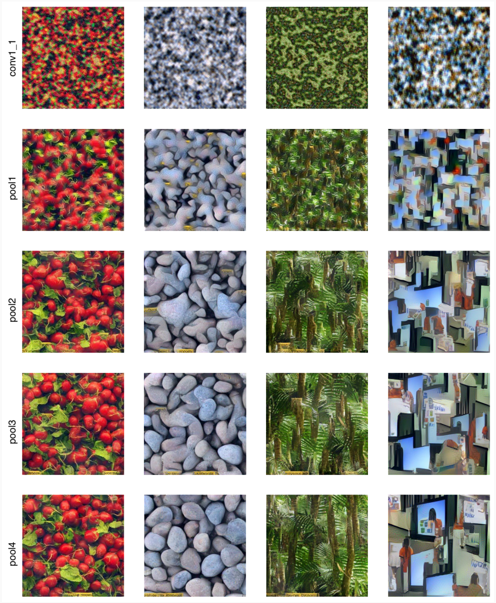
Given an example image u and a random initialization x_0, one optimizes the loss function

$$E(x) = \sum_{\text{for selected layers } L} w_L \left\| G^L(x) - G^L(u) \right\|_F^2$$

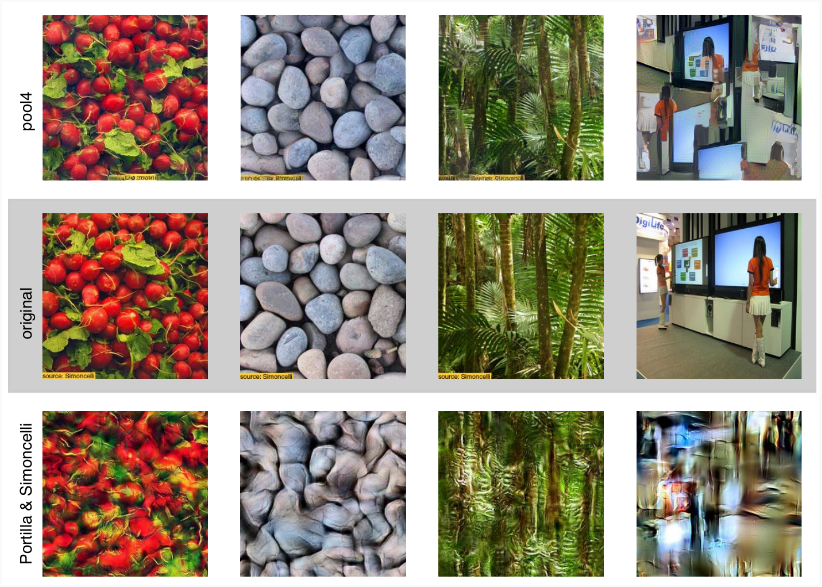
Pseudo-code:

```
Require: Input image $u$, set of selected VGG-19 layers $\mathcal{L}$ and associated
    layer weights parameter $\{w_L,\ L \in \mathcal{L}\}$
Ensure: Synthesized texture $x$
    Apply VGG-19 to $u$ and extract the layers $\{V_L(u)),\ L \in \mathcal{L}\}$
    Compute the target Gram matrices $\{G_L(u) = \mathrm{Gram}(V_L(u)),\ L \in \mathcal{L}\}$.
    Initialize $x$ with some Gaussian white noise.
    for $N_{\text{it}}$ iterations do
        Apply VGG-19 to $x$ and extract the layers $\{V_L(x)),\ L \in \mathcal{L}\}$.
        Compute the current Gram matrices of $x$: $\{G_L(x) = \mathrm{Gram}(V_L(x)),\ L \in \mathcal{L}\}$
        Compute the loss $E(x)$ and its gradient $\nabla_x E(x)$ using backpropagation.
        $x \leftarrow \text{L-BFGS-step}(x, E(x), \nabla_x E(x))$.
    end for
```

Gatys et al algorithm: Depth influence



Gatys et al algorithm: Results

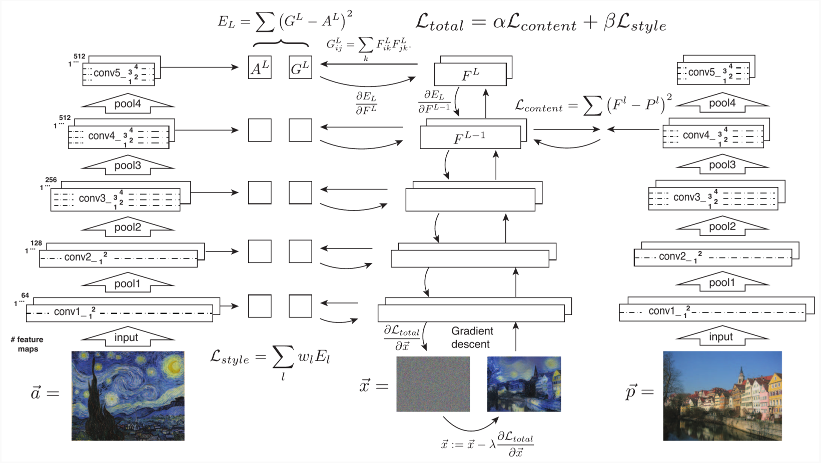


Gatys et al algorithm

- This algorithm is the current state of the art.
- The computational cost is really high (even with high-end GPUs it takes minutes).
- A lot of improvements have been proposed, eg by adding term to the energy or by adding correlation between layers.
- Extension for style transfer with equally impressive results, and maybe more impact.

Gatys et al for style transfer

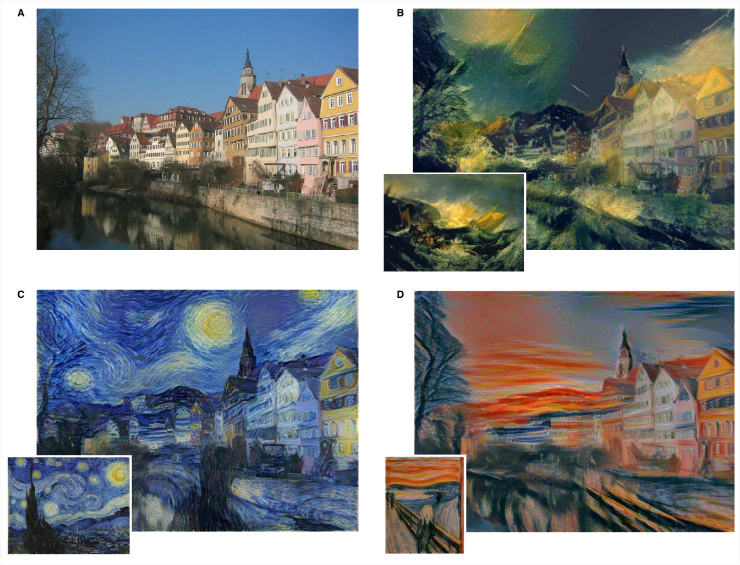
Reference: (Gatys et al., 2016)



$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$G^L_{ij} = \sum_k F^L_{ik}F^L_{jk}$
A^L
G^L
F^L
F^{L-1}
$\frac{\partial E_L}{\partial F^L}$
$\frac{\partial E_L}{\partial F^{L-1}}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
conv5_1 2 3 4
pool4
conv4_1 2 3 4
pool3
conv3_1 2 3 4
pool2
conv2_1 2
pool1
conv1_1 2
input
512
512
256
128
64
# feature maps
$\vec{a} =$
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
Gradient descent
$\vec{x} =$
$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
$\vec{p} =$

Gatys et al for style transfer

Reference: (Gatys et al., 2016)

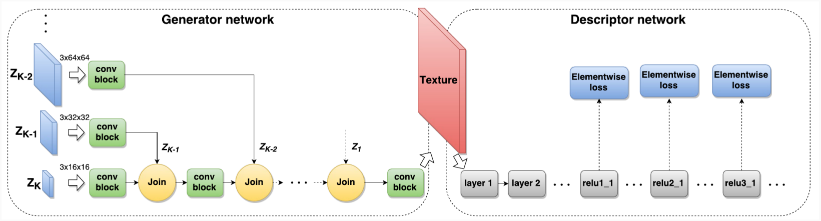


Gatys et al for texture synthesis and style transfer

- Very nice and clean PyTorch implementation:
  `https://github.com/leongatys/PytorchNeuralStyleTransfer`
- Today: Practice session based in this code.
- **Very slow** on CPU and computationally demanding with high-end GPU (and memory consuming, e.g. 8 GB of memory for a 1024×1024 image). See practice session.
- Regarding texture modeling, the **number of parameters is huge**: Textures are described by the Gram matrices and the number of elements in the Gram matrices totals 850k. That is 1000 times more than Portilla-Simoncelli !

Generative networks for texture synthesis

- A workaround for speeding up synthesis is to train generative forward networks to mimic Gatys algorithm, as proposed by (Ulyanov et al., 2016) (coined Texture Networks).
- This is **an example of generative network**.
- The generator is trained to produce images with low Gatys loss (self-supervised training) from multi-scale white noise.
- Synthesis is fast thanks to the feedforward architecture.



Generative networks for texture synthesis

Texture Networks

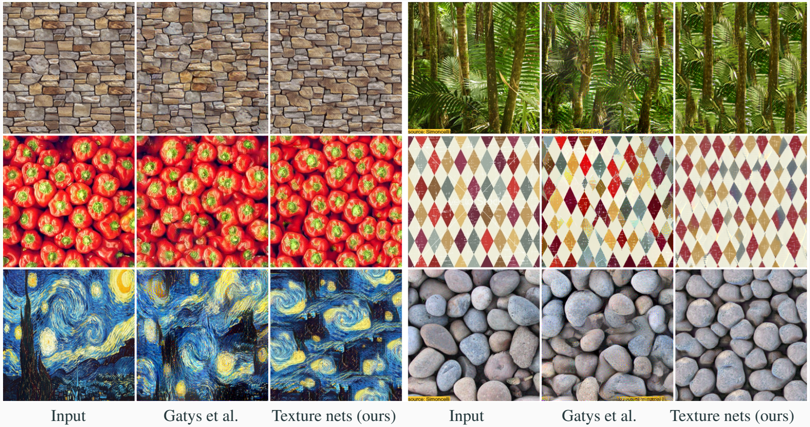
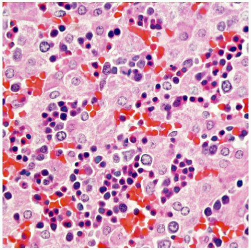


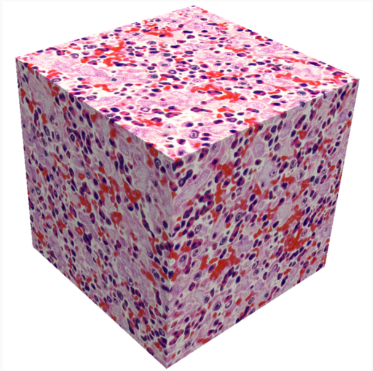
Figure 1. Texture networks proposed in this work are feed-forward architectures capable of learning to synthesize complex textures based on a single training example. The perceptual quality of the feed-forwardly generated textures is similar to the results of the closely related method suggested in (Gatys et al., 2015a), which use slow optimization process.

- However texture quality is not as good, and a network has to be trained for each new image.

On demand solid texture synthesis using deep 3D networks

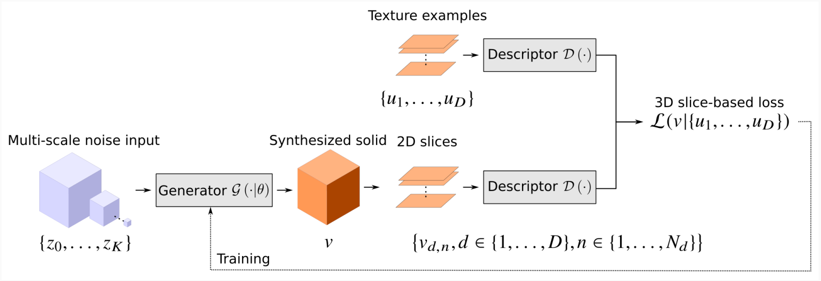
- Texture networks can be extended to 3D (Gutierrez et al., 2020) while the Gatys approach is infeasible.


Input


Output

On demand solid texture synthesis using deep 3D networks

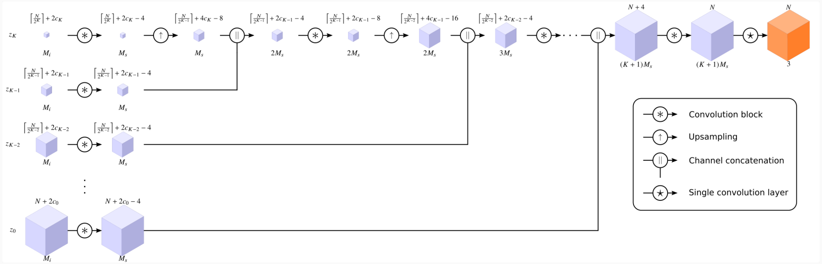
Training framework for the proposed CNN Generator network:


Texture examples
Descriptor $\mathcal{D}(\cdot)$
$\{u_1,...,u_D\}$
3D slice-based loss
$\mathcal{L}(v|\{u_1,...,u_D\})$
Multi-scale noise input
Synthesized solid
2D slices
Generator $\mathcal{G}(\cdot|\theta)$
Descriptor $\mathcal{D}(\cdot)$
$\{z_0,...,z_K\}$
v
$\{v_{d,n}, d \in \{1,...,D\}, n \in \{1,...,N_d\}\}$
Training

- The generator $\mathcal{G}(\cdot|\theta)$ with parameters θ processes a multi-scale noise input Z to produce a solid texture v
- The loss $\mathcal{L}$ compares, for each direction d, the feature statistics induced by the example u_d in the layers of the pre-trained Descriptor network $\mathcal{D}(\cdot)$ (loss of Gatys et al).

On demand solid texture synthesis using deep 3D networks

Schematic of the Generator's architecture:

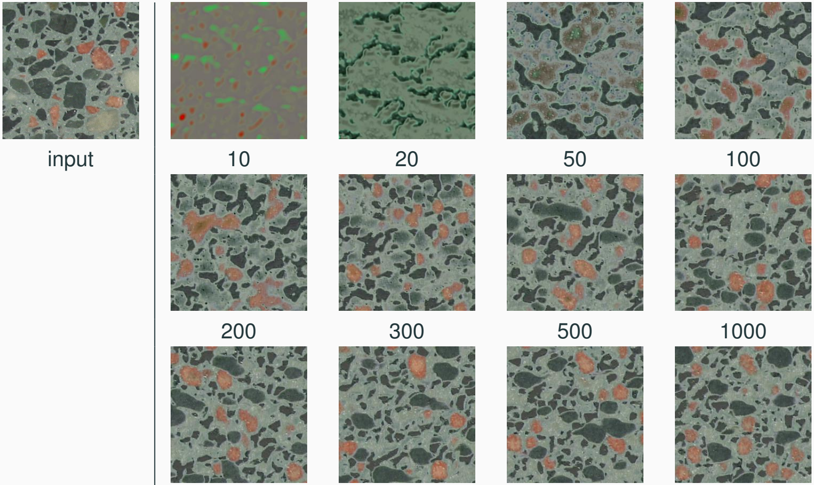


- Processes a set of noise inputs $Z = \{z_0, \dots, z_K\}$ at $K+1$ different scales using convolution operations and non-linear activations
- The information at different scales is combined using upsampling and channel concatenation (similar to the right part of a U-net).

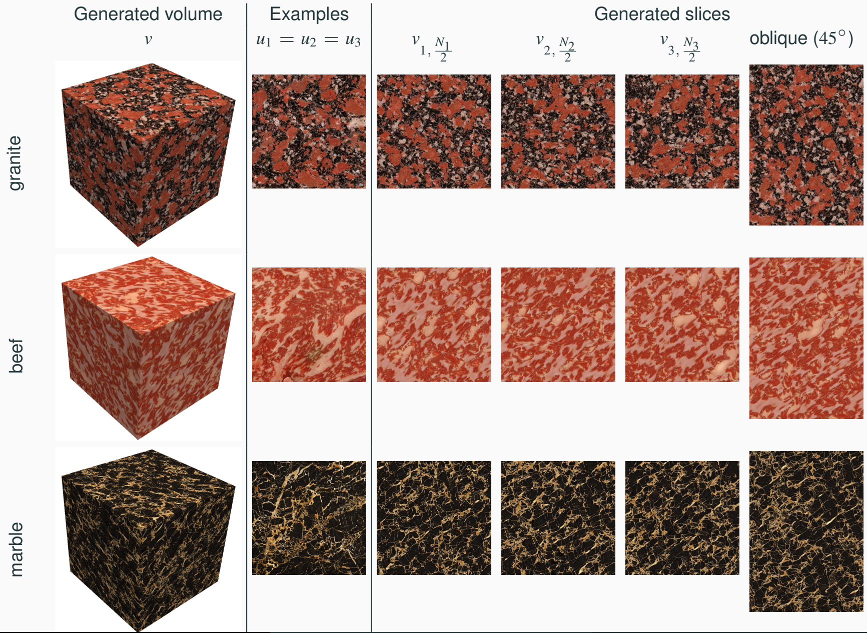
On demand solid texture synthesis using deep 3D networks

Training: Find the parameters θ for a given texture

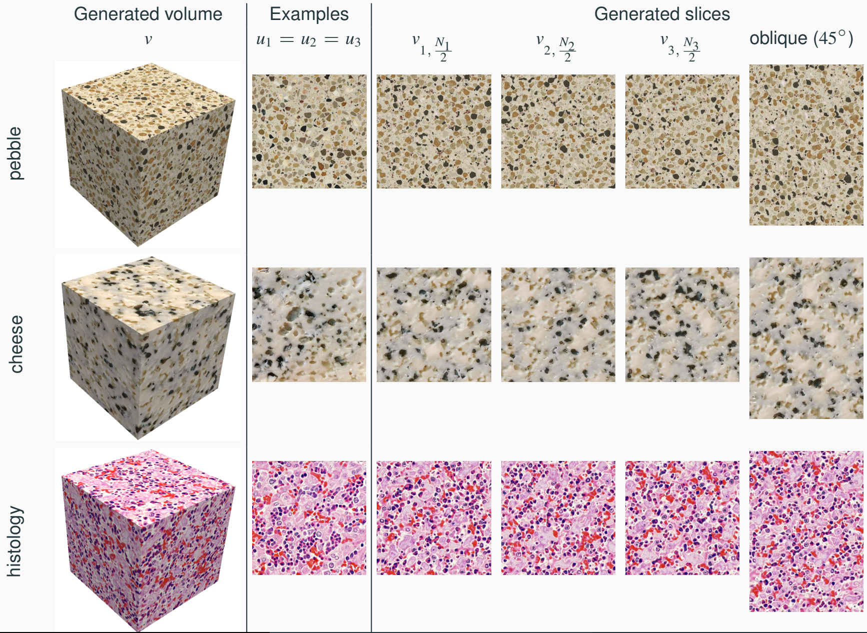
- Exploit invariance by translation to generate batches of width one voxel only ("single-slice training scheme")
- Minimize Gatys' loss for each slice, using 3000 iterations of Adam algorithm



On demand solid texture synthesis using deep 3D networks

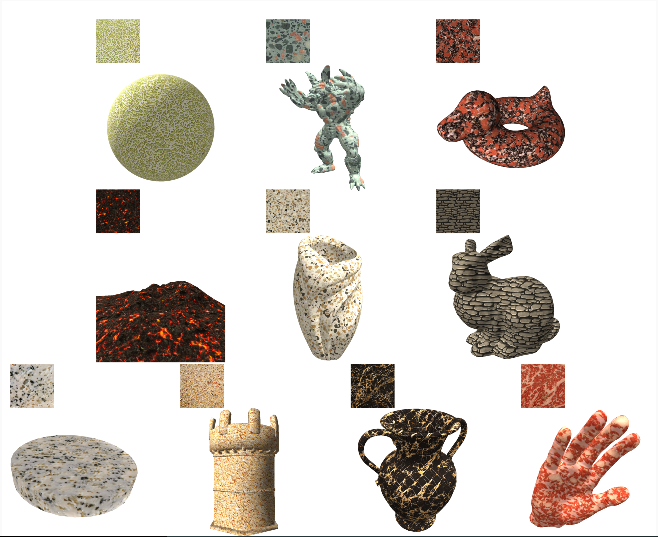


On demand solid texture synthesis using deep 3D networks



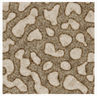
On demand solid texture synthesis using deep 3D networks

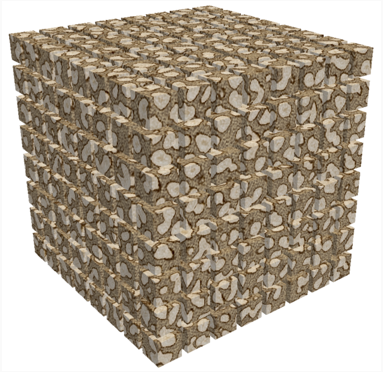
- Solid textures can be used to apply textures on surfaces without parametrization.



On demand solid texture synthesis using deep 3D networks

- Fast synthesis thanks to the feed forward network (1 sec. for 256^3)
- On demand synthesis using a pseudo random number generator seed with spatial coordinates





- Training and synthesis with high resolution images without memory issues thanks to the single slice strategy.

Generative networks for texture synthesis

- Other contributions propose generative networks for **universal** style transfer/texture synthesis (e.g. (Li et al., 2017)).
- **Universal** means that a single network is adapted to all images.
- This still relies in approximating the Gatys procedure with some approximation for a faster style transfer: auto-encoders to invert VGG19 and imposing Gram matrices.

- Using Wasserstein distance between Gaussians in VGG19 feature space is efficient for texture mixing (Vacher et al., 2020) (see practice session).

References

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*.

Galerne, B., Gousseau, Y., and Morel, J.-M. (2011). Random phase textures: Theory and synthesis. *IEEE Trans. Image Process.*, 20(1):257 – 267.

Gatys, L., Ecker, A. S., and Bethge, M. (2015). Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, pages 262 – 270.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.

References ii

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gutierrez, J., Rabin, J., Galerne, B., and Hurtut, T. (2020). On demand solid texture synthesis using deep 3d networks. *Computer Graphics Forum*, 39(1):511–530.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

References iii

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham. Springer International Publishing.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., and Yang, M.-H. (2017). Universal style transfer via feature transforms. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ongie, G., Jalal, A., Metzler, C. A., Baraniuk, R. G., Dimakis, A. G., and Willett, R. (2020). Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56.

References iv

Phongthawee, P., Chinchuthakun, W., Sinsunthithet, N., Jampani, V., Raj, A., Khungurn, P., and Suwajanakorn, S. (2024). Diffusionlight: Light probes for free by painting a chrome ball. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 98–108.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023). Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding.

References v

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France. PMLR.

Song, J., Vahdat, A., Mardani, M., and Kautz, J. (2023). Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*.

Ulyanov, D., Lebedev, V., Vedaldi, A., and Lempitsky, V. (2016). Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357.

References vi

Vacher, J., Davila, A., Kohn, A., and Coen-Cagli, R. (2020). Texture interpolation for probing visual perception. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22146–22157. Curran Associates, Inc.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Zhang, L., Rao, A., and Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.