

## SISY/ATI Machine et deep learning pour l'image

### 6. Modèles génératifs et application en restauration

Thomas Oberlin

ISAE-SUPAERO, Département d'ingénierie des systèmes complexes (DISC)

[thomas.oberlin@isae-supaero.fr](mailto:thomas.oberlin@isae-supaero.fr)

# Plan de la séance

## 1. Restauration d'images et problèmes inverses

Restauration d'images

Modélisation des problèmes inverses

Problèmes inverses: le point de vue bayésien

## 2. Modèles profonds génératifs

Variational autoencoders (VAE)

Invertible Neural Networks (INNs)

Generative Adversarial Networks (GAN)

## 3. Deep learning pour les problèmes inverses

Approches “model-based”

Approches “deep generative priors”

# Plan de la séance

## 1. Restauration d'images et problèmes inverses

Restauration d'images

Modélisation des problèmes inverses

Problèmes inverses: le point de vue bayésien

## 2. Modèles profonds génératifs

## 3. Deep learning pour les problèmes inverses

# Débruitage

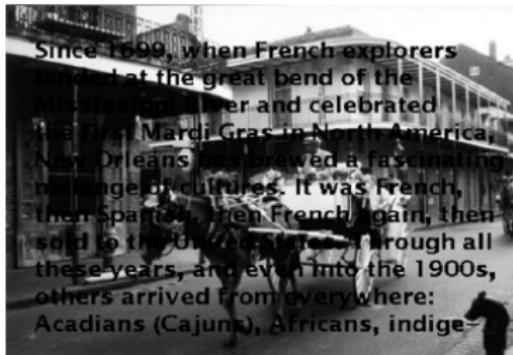
## Sources de bruit

- ▶ Bruit de mesure (Gaussien/Poissonien)
- ▶ Défauts de capteurs
- ▶ Bruit d'égalisation (strip noise)
- ▶ Artefacts JPEG



# Inpainting

Inpainting : reconstruire des pixels manquants (masque structuré ou non)



a

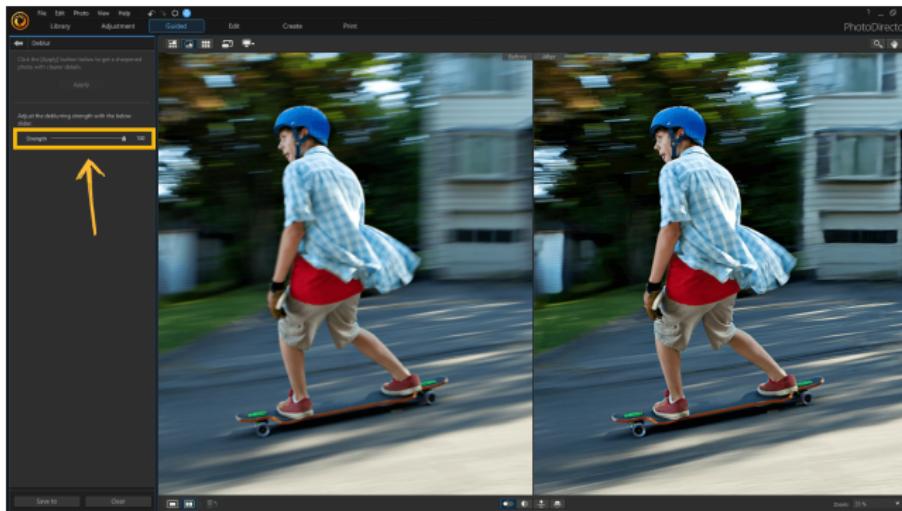


b

# Déconvolution (défloutage)

## Sources de flou

- ▶ Diffraction
- ▶ Mise au point (focus) et défauts optiques du capteur
- ▶ Bougé (de la scène / du capteur)
- ▶ etc



# Super-résolution

Super-résolution : sur-échantillonnage + défloutage



Low-resolution (LR) image

Super-resolution (SR)



High-resolution (HR) image

# Problème direct

## Modélisation d'une dégradation

$$y = Ax + b$$

- ▶  $x$  image réelle recherchée
- ▶  $y$  image observée
- ▶  $A$  opérateur de dégradation, supposé connu
- ▶ "b" bruit additif aléatoire (ex: Gaussien)

## Exemples

- ▶  $A = I$  : débruitage
- ▶  $A$  matrice de masque : inpainting
- ▶  $A = F^*D$  ( $F$ : DFT) : déconvolution ( $Ax = h \star x$  ou encore :  $A$  est Toeplitz)
- ▶  $A = SF^*D$  : super-résolution

# Problème inverse

- ▶ Comment estimer  $x$  étant donné  $y$ ?
- ▶ Plusieurs réponses :
  - ▶ Réponse naïve :  $x = A^{-1}y$ . Si  $A$  n'est pas inversible?
  - ▶ Inverse généralisé (Moore-Penrose, solution aux moindres carrés) :

$$\hat{x} = \arg \min_x \|y - Ax\|_2^2 = (A^* A)^{-1} A^* y \quad (1)$$

- ▶ Mais : instable (et pas forcément défini).  $\longrightarrow$  Régularisation de Tychonov :

$$\hat{x} = \arg \min_x \|y - Ax\|_2^2 + \lambda \|x\|_2^2 = (A^* A + \lambda I)^{-1} A^* y \quad (2)$$

- ▶ Plus généralement : régulariser le problème, si possible en ajoutant de l'information a priori sur  $x$ .

## Quelques régularisations

- ▶  $\|x\|_1 = \sum_k |x_k|$  : parcimonie (LASSO)
- ▶  $\|Dx\|_2^2$  avec  $D$  gradient discret : image lisse
- ▶  $\|Dx\|_1$  image lisse par morceaux (Variation totale)
- ▶  $\|Wx\|_1$  avec  $W$  ondelettes

# Point de vue bayésien

## Estimation bayésienne

- ▶ Ajout d'un **prior**  $x \sim p(x)$
- ▶ **Maximum a posteriori (MAP)** :

$$\begin{aligned}\hat{x} &= \arg \max_x p(x|y) \underbrace{=}_{\text{Bayes}} \arg \max_x p(y|x)p(x) \\ &= \arg \min_x -\log p(y|x) - \log p(x).\end{aligned}$$

- ▶ Bruit Gaussien et prior Gaussien ( $b \sim \mathcal{N}(0, \sigma^2 I)$  et  $p(x) = \mathcal{N}(0, \gamma^2 I)$ ) :

$$\hat{x} = \arg \min_x \frac{1}{2\sigma^2} \|y - Ax\|_2^2 + \frac{1}{2\gamma^2} \|x\|_2^2.$$

## Modèles hiérarchiques

- ▶  $x \sim p_\theta(x)$
- ▶ Estimation jointe de  $\theta$  et de  $x$

# Problèmes inverses en pratique

## Algorithmes

- ▶ Optimisation (convexe ou non)
- ▶ Optimisation alternée, algorithme EM
- ▶ Algorithmes d'échantillonnage : Markov Chain Monte Carlo (MCMC)
- ▶ Inférence variationnelle : approximation mean-field

## Choix du prior

- ▶ Crucial évidemment!
- ▶ Pas de “bonne” solution
- ▶ Dépend de l'expérience, et/ou des possibilités d'inférence (priors conjugués)
- ▶ Pourquoi ne pas l'apprendre?

# Plan de la séance

1. Restauration d'images et problèmes inverses

2. Modèles profonds génératifs

Variational autoencoders (VAE)

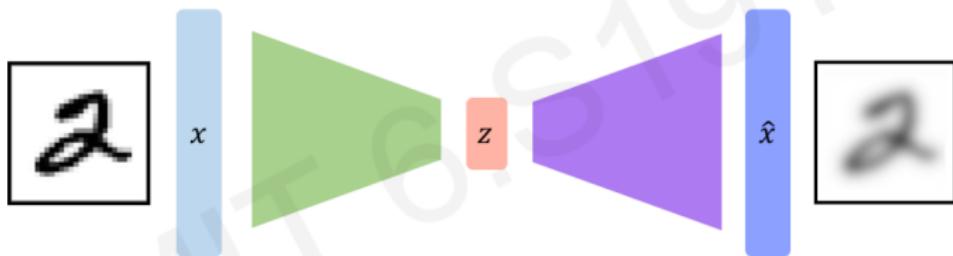
Invertible Neural Networks (INNs)

Generative Adversarial Networks (GAN)

3. Deep learning pour les problèmes inverses

# Motivation

## Traditional autoencoders



[Kingma and Welling 2014]

- ▶ Auto-encodeur  $\approx$  réduction de dimension.
- ▶ But des modèles génératifs : “contrôler” la distribution de la variable latente  $z$ 
  - ▶ Stabilité (deux images proches auront deux représentations latentes proches)
  - ▶ Structure (features dans l'espace latent)
  - ▶ Échantillonnage dans l'espace latent

# Un problème d'inférence

- ▶ Modèle génératif

$$z \sim p(z) \text{ prior}$$

$x \sim p_\theta(x|z) = \mathcal{N}(D_\theta(z), \eta I)$ , avec  $D_\theta$  réseau décodeur ou générateur.

- ▶ On recherche les poids qui maximisent la vraisemblance

$$p_\theta(x) = \int p_\theta(x|z)p(z) dz.$$

- ▶ Intractable, on utilise une approximation appelée inférence variationnelle

$$\begin{aligned} \log p_\theta(x) &= \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x) \\ &\geq \mathbb{E}_{z \sim q_\phi(z|x)} \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \text{ appelée ELBO ou VLB} \end{aligned}$$

# Apprentissage d'un VAE

- ▶ Loss = ELBO :

$$\mathcal{L}_{\theta, \phi}(x) = \underbrace{\mathbb{E}_{z \sim q_{\phi}(z|x)} \log p_{\theta}(x|z)}_{\text{Erreur de reconstruction}} + \underbrace{\mathbb{E}_{z \sim q_{\phi}(z|x)} \log \frac{p(z)}{q_{\phi}(z|x)}}_{\text{Régularisation}}$$

- ▶ Erreur de reconstruction :

$$\log p_{\theta}(x|z) = -\frac{1}{2\eta} \|x - D_{\theta}(z)\|_2^2$$

- ▶ Régularisation :

$$\log \frac{p(z)}{q_{\phi}(z|x)} = -D_{KL}(q_{\phi}(z|x)||p(z))$$

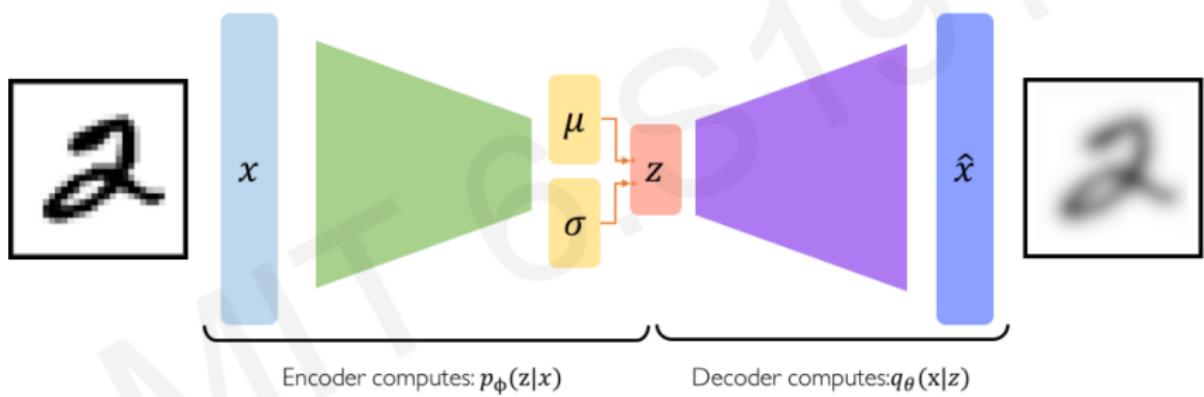
- ▶ Paramétrisation de  $q_{\phi}$ : pour un apprentissage efficace on pose

$$q_{\phi}(z|x) = \mathcal{N}(\mu(x), \sigma(x)I),$$

où  $\mu$  et  $\sigma$  sont calculées par l'encodeur  $E(x)$ .

# Apprentissage d'un VAE

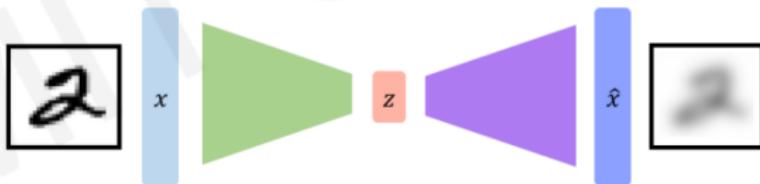
## VAE optimization



$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{regularization term})$$

## VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
5. Generating new examples



## VAE face interpolation

---

We can also gradually add features to faces:

- Find the center of mass of all encoded faces with a certain feature.
- Find the center of mass of all encoded faces without that feature.
- The difference  $\Delta$  of these vectors “point” in the feature’s “direction.”
- Add multiples of  $\Delta$  of increasing length to an encoded face and decode.



# Invertible Neural Networks (INNs)

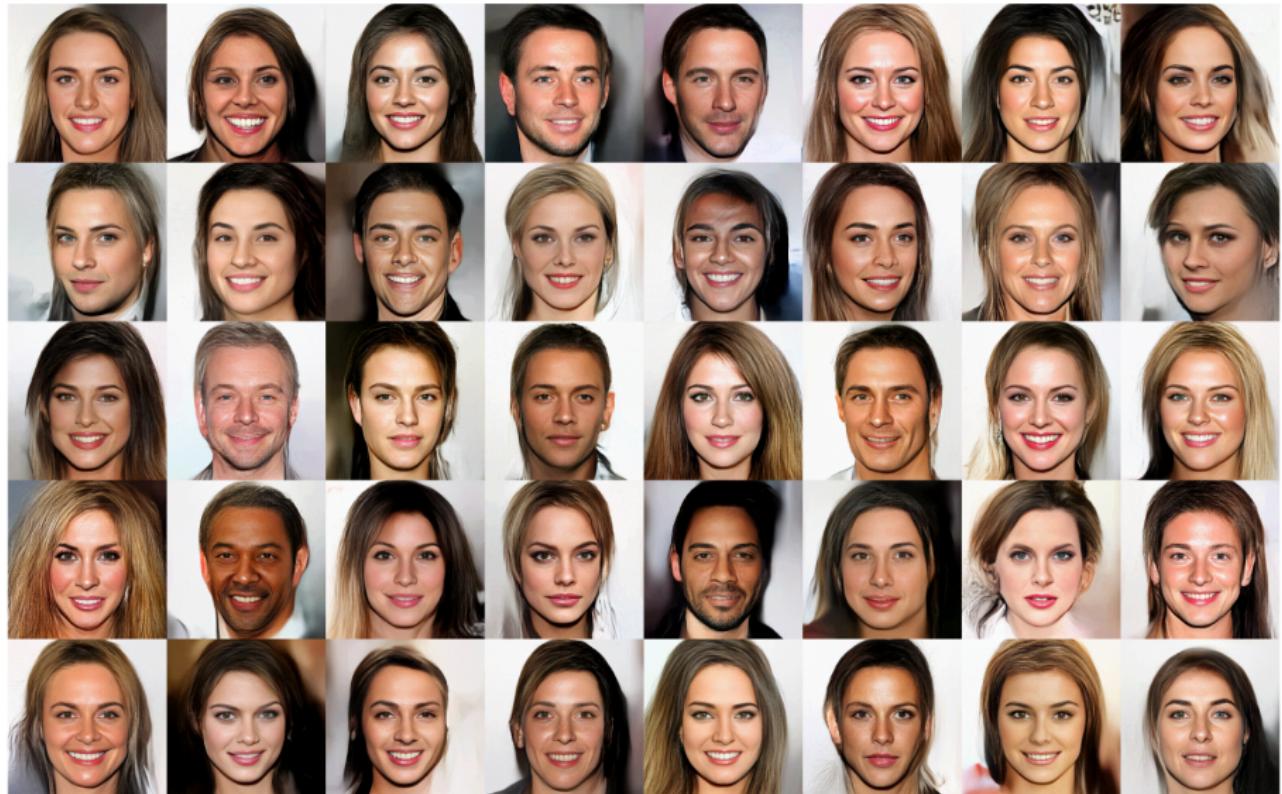
- ▶ Même modèle génératif

$$z \sim p(z) \text{ prior}$$

$x \sim p_\theta(x|z) = \mathcal{N}(D_\theta(z), \eta I)$  réseau décodeur ou générateur.

- ▶ Architecture spécifique de  $D_\theta$  pour permettre l'inférence en optimisant la vraisemblance de manière **exacte**.
- ▶ Idées principales [Dinh et al 2016] :
  - ▶ Couches inversibles —> pas de convolution (sauf  $1 \times 1$ )
  - ▶ Opérations spécifiques avec Jacobien triangulaire (pour le calcul rapide du déterminant)
  - ▶ Réseaux profonds

## Illustration : échantillons



[Kingma et al 2018]

## Illustration : interpolation dans l'espace latent



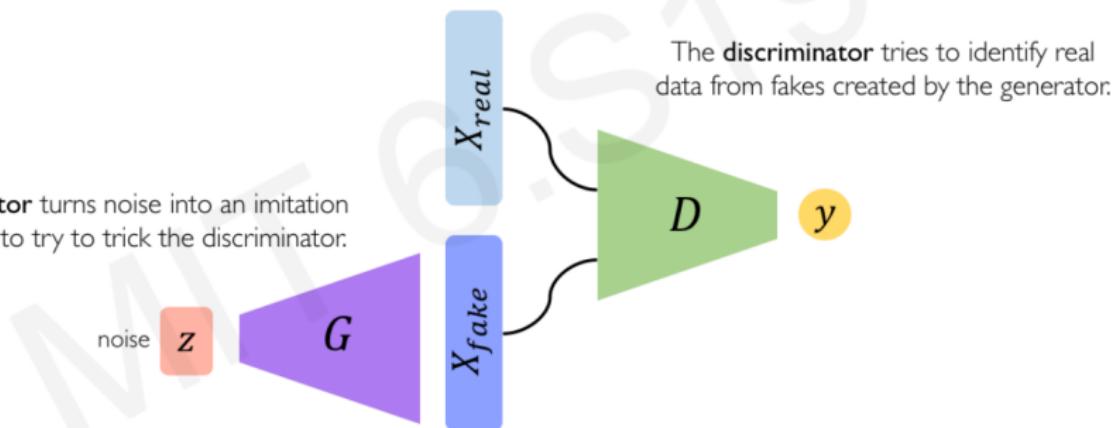
[Kingma et al 2018]

# Generative Adversarial Networks (GAN)

## Generative Adversarial Networks (GANs)

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

The **generator** turns noise into an imitation of the data to try to trick the discriminator.



[Goodfellow et al 2014]

# Entraînement d'un GAN

## Training GANs

**Discriminator** tries to identify real data from fakes created by the generator.

**Generator** tries to create imitations of data to trick the discriminator.

Train GAN jointly via **minimax** game:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left( 1 - D_{\theta_d}(G_{\theta_g}(z)) \right) \right]$$

**Discriminator** wants to maximize objective s.t.  $D(x)$  close to 1,  $D(G(z))$  close to 0.

**Generator** wants to minimize objective s.t.  $D(G(z))$  close to 1.

## Progressive growing of GANs: results



# Plan de la séance

1. Restauration d'images et problèmes inverses

2. Modèles profonds génératifs

3. Deep learning pour les problèmes inverses

Approches “model-based”

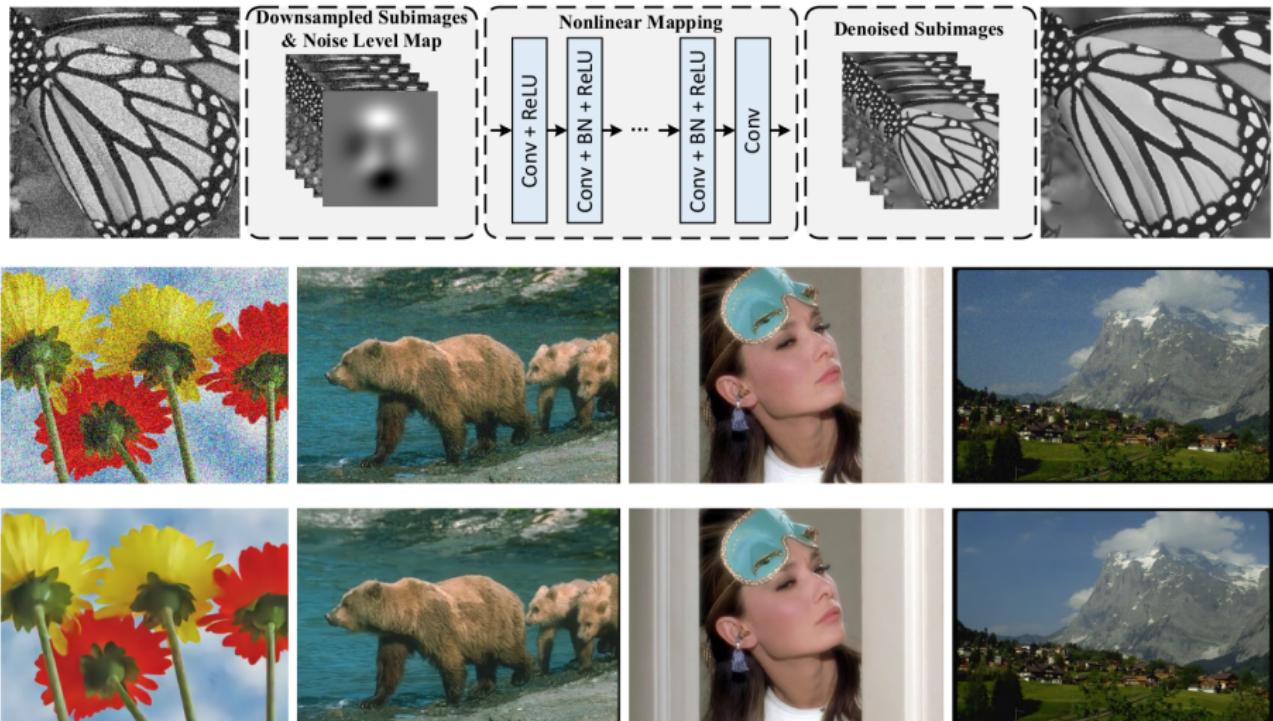
Approches “deep generative priors”

# L'approche “model-based”

## Approche supervisée

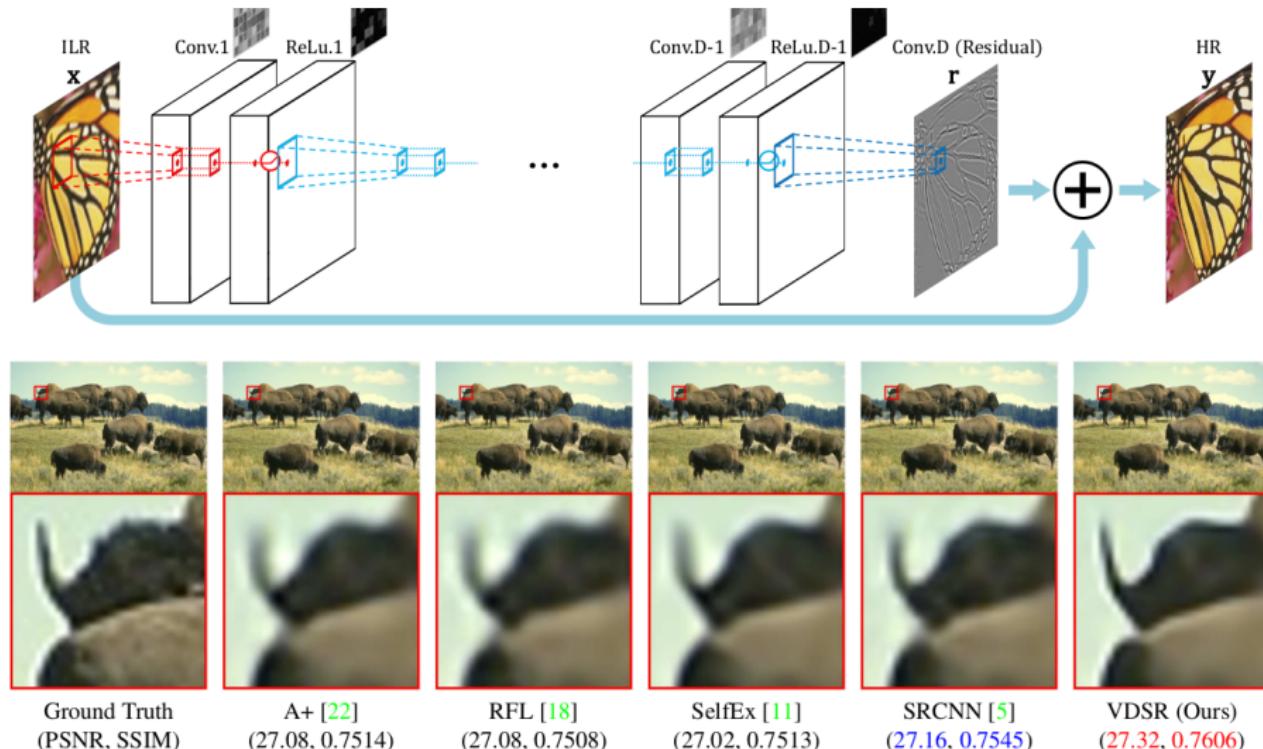
- ▶ On apprend un réseau  $N$  qui résout directement le problème inverse :  
 $x_i \approx N(y_i)$  sur un ensemble d'apprentissage
- ▶ Approche supervisée ( $\approx$  problème de régression)
- ▶ (-) En général on simule les données
- ▶ (-) Le réseau est spécifique au problème : si on change les paramètres (forme du flou, type ou puissance de bruit), on doit refaire l'apprentissage
- ▶ (+) Les performances peuvent être très bonnes
- ▶ (+) L'utilisation est simple (une passe forward dans le réseau)

# FFDNet



[Zhang et al. 2018]

# VDSR



[Kim et al. 2016]

# Approches “deep generative priors”

## Apprentissage d'un modèle génératif

- ▶ GAN, VAE ou INN :  $E(x)$  et/ou  $D(z)$
- ▶ Permet de construire un prior  $p(x)$

## Résolution directe du problème inverse [Bora et al 2017]

- ▶ Une approche MAP donne

$$\hat{z} = \arg \min_z = \|y - AD(z)\| + \lambda \|z\|_2^2$$

- ▶ puis  $\hat{x} = D(\hat{z})$ .

- ▶ (+) Un même réseau pour différents problèmes inverses et paramètres
- ▶ (+) pas besoin de données simulées
- ▶ (+) meilleur contrôle de l'estimation
- ▶ (-) plus complexe car descente de gradient pour estimer  $\hat{x}$
- ▶ (-) Parfois moins performant