Modèles génératifs profonds

Guillaume Bourmaud

PLAN

I. Introduction

II. Notion de divergence

III. Réseau inversible (NF)

IV. Auto-encodeur variationnel (VAE)

V. Réseau débruiteur (DDPM)

VI. Réseau antagoniste (GAN)

I) Introduction

I)

Principe d’un modèle génératif

Base de données : $\{\mathbf{X}_i\}_{i=1\dots N}$

$\in \mathbb{R}^n$

I)

Principe d'un modèle génératif

Base de données : $\{\mathbf{X}_i\}_{i=1\dots N}$

$\in \mathbb{R}^n$

Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** qui **ressemblent** aux $\{\mathbf{X}_i\}_{i=1\dots N}$

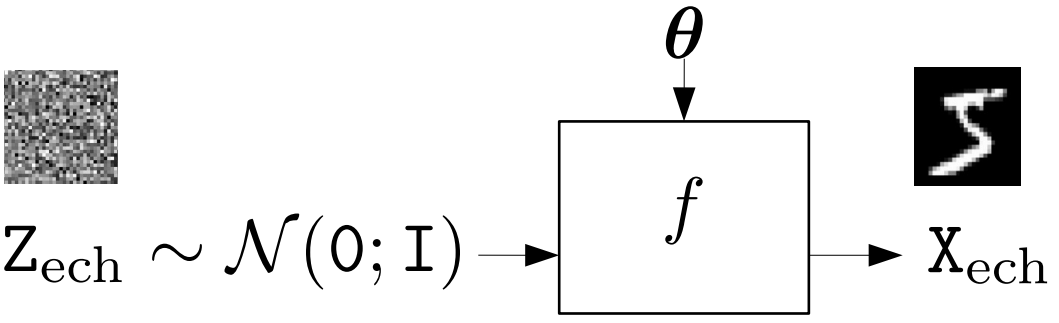
I)

Principe d'un modèle génératif

Base de données : $\{\mathtt{X}_i\}_{i=1\dots N}$

$\in \mathbb{R}^n$

Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** qui **ressemblent** aux $\{\mathtt{X}_i\}_{i=1\dots N}$



I)

Principe d'un modèle génératif

Base de données : $\{\mathbf{X}_i\}_{i=1\dots N}$

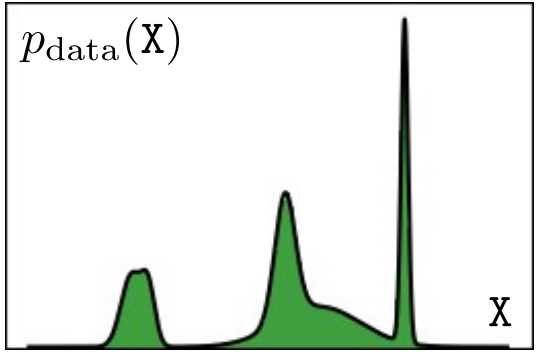
$\in \mathbb{R}^n$

Point de vue probabiliste

"est un échantillon de"

$\mathbf{X}_i \sim p_{\text{data}}(\mathbf{X})$

Densité de probabilité **inconnue**



Exemple 1D

I)

Principe d'un modèle génératif

Base de données : $\{\mathbf{X}_i\}_{i=1\dots N}$

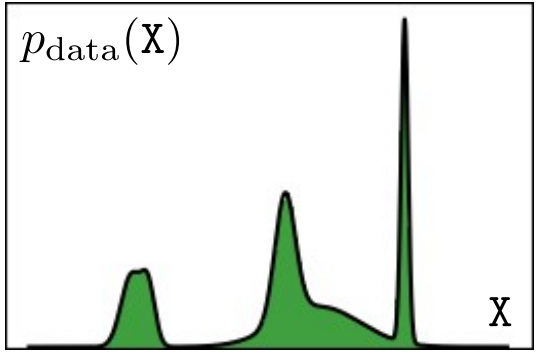
$\in \mathbb{R}^n$

Point de vue probabiliste

"est un échantillon de"

$\mathbf{X}_i \sim p_{\text{data}}(\mathbf{X})$

Densité de probabilité **inconnue**



Exemple 1D

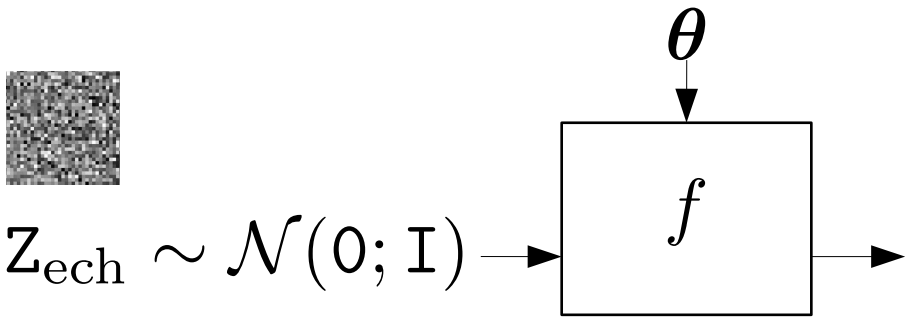
On supposera les $\mathbf{X}_i$ i.i.d.

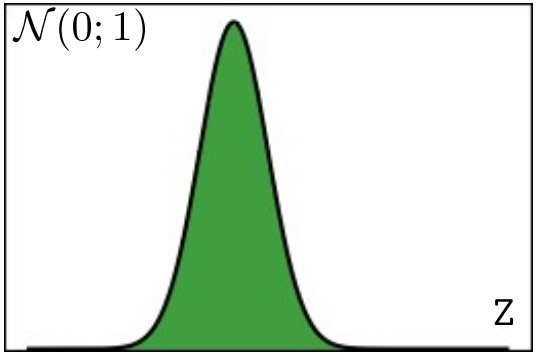
Principe d’un modèle génératif (suite)

Objectif : Optimiser les paramètres d’un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathrm{X})$

Principe d'un modèle génératif (suite)

Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathtt{X})$

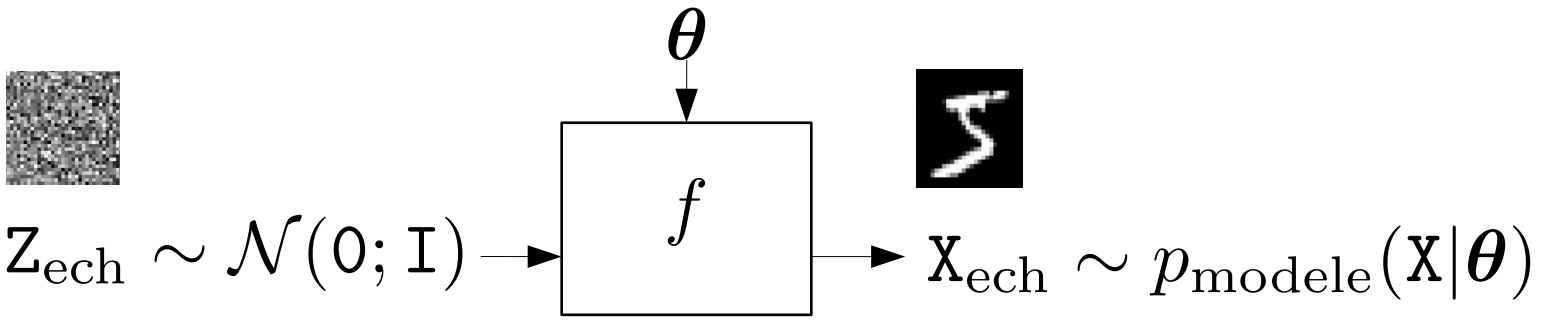


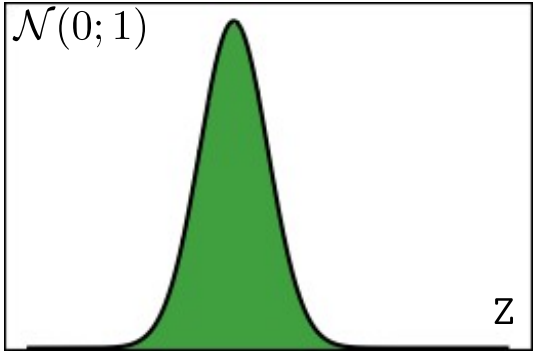


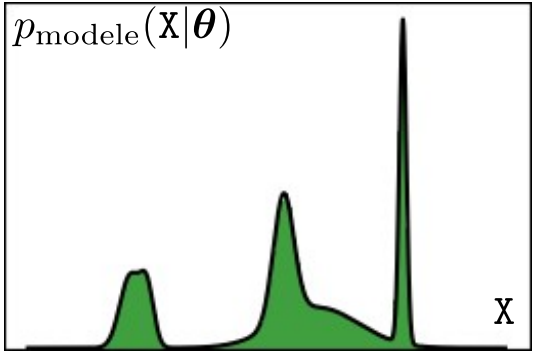
I)

Principe d'un modèle génératif (suite)

Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathtt{X})$



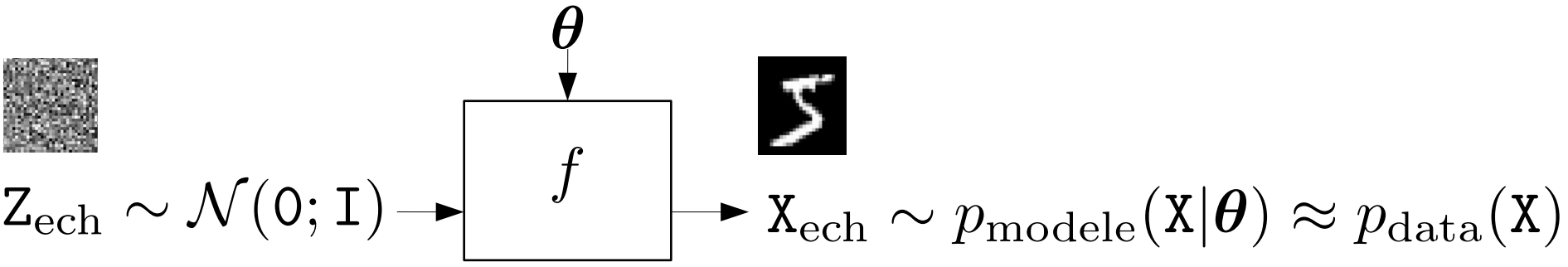


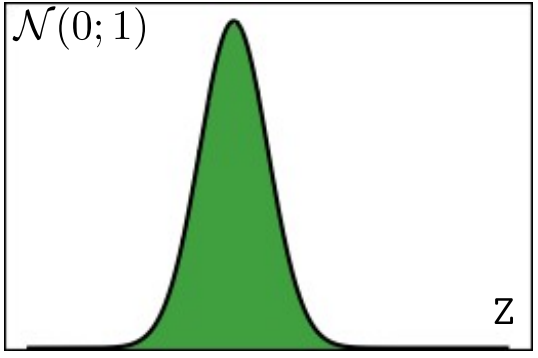


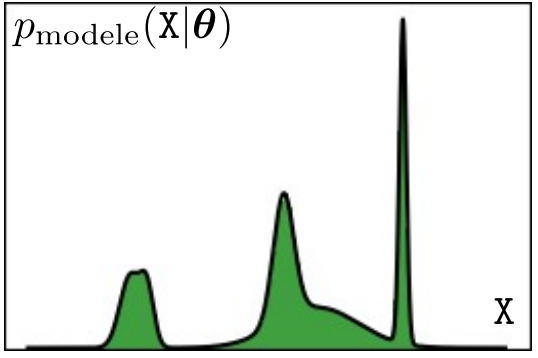
I)

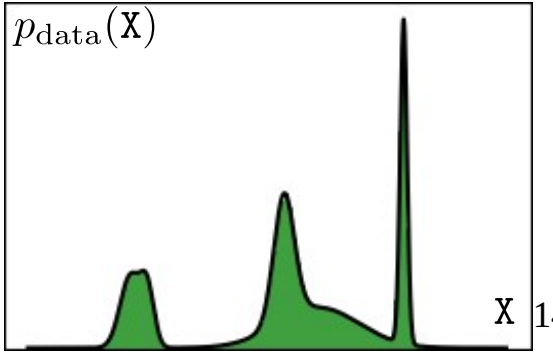
Principe d'un modèle génératif (suite)

Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathtt{X})$









I)

Difficulté de la tâche d'apprentissage

Base de données **non-étiquetées** : $\{\mathbf{X}_i\}_{i=1\ldots N}$

I)

Difficulté de la tâche d'apprentissage

Base de données **non-étiquetées** : $\{\mathrm{X}_i\}_{i=1\ldots N}$

On ne dispose pas de couples $\{\mathrm{Z}_i, \mathrm{X}_i\}_{i=1\ldots N}$

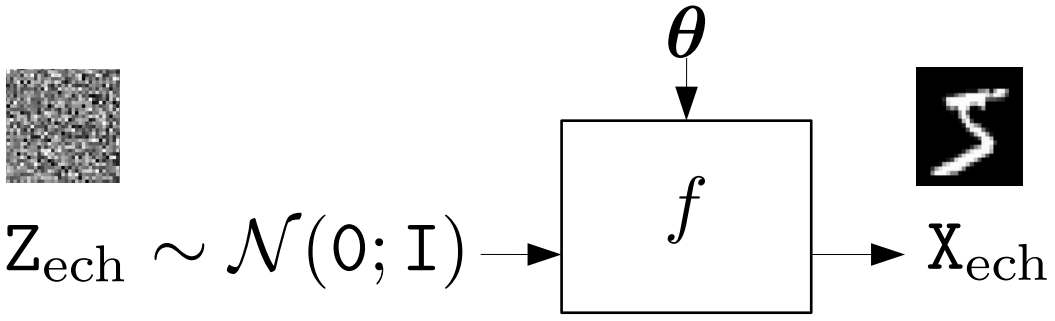
→ problème d'apprentissage non-supervisé

Difficulté de la tâche d'apprentissage

Base de données **non-étiquetées** : $\{\mathrm{X}_i\}_{i=1\dots N}$

On ne dispose pas de couples $\{\mathrm{Z}_i, \mathrm{X}_i\}_{i=1\dots N}$

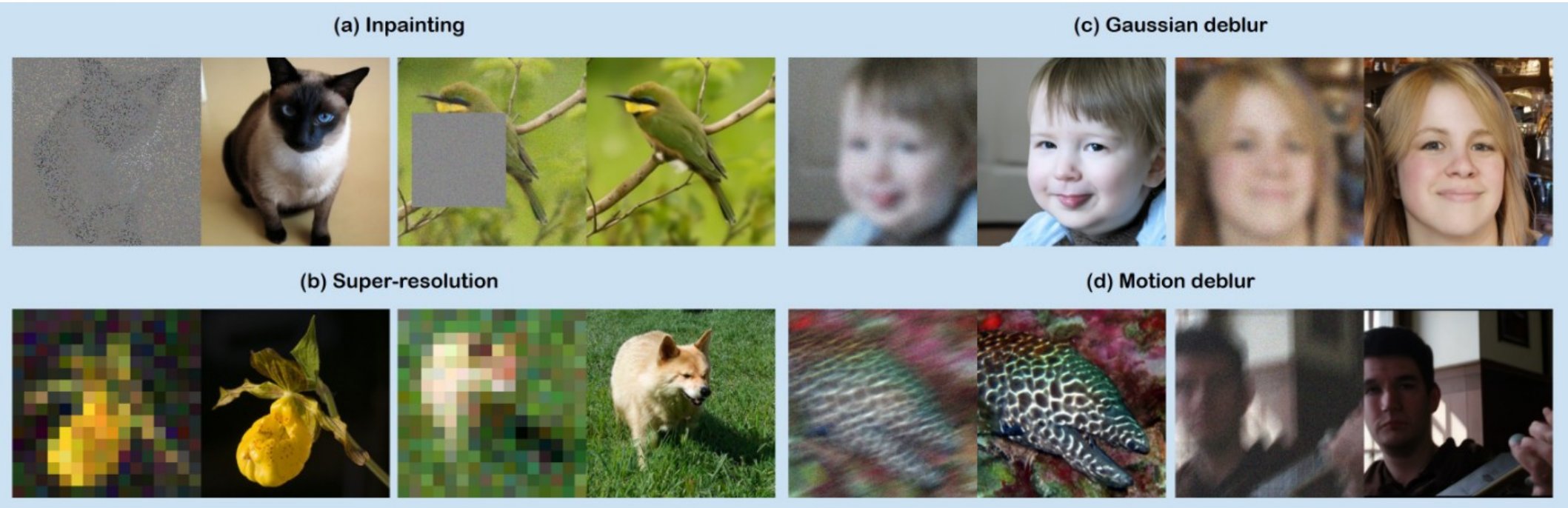
→ problème d'apprentissage non-supervisé



Terminologie : Z_i = variable "latente"

I)

Exemple d’utilisation : Apprentissage d’apriori pour des problèmes inverses



Chung et al., DPS, ICLR 2023 → TP inpainting 4h

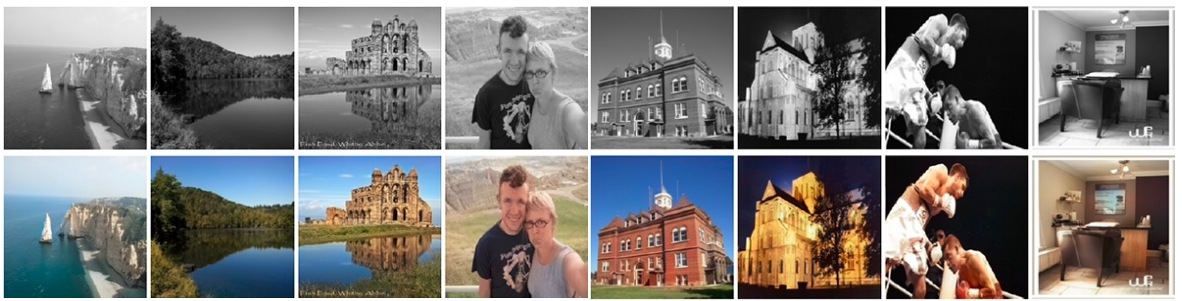
I)

Principe d’un modèle génératif conditionnel

Base de données **étiquetées** : $\{\mathrm{Y}_i, \mathrm{X}_i\}_{i=1\ldots N}$

Y_i

X_i



Exemple : Colorisation d’image

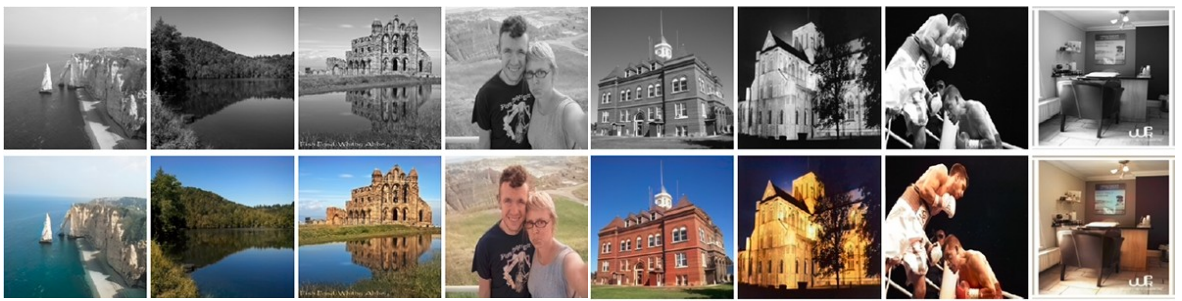
I)

Principe d'un modèle génératif conditionnel

Base de données **étiquetées** : $\{\mathrm{Y}_i, \mathrm{X}_i\}_{i=1\ldots N}$

Y_i

X_i



Exemple : Colorisation d'image

Point de vue probabiliste : $\mathrm{X}_i \sim p_{\mathrm{data}}(\mathrm{X}|\mathrm{Y}_i)$ **Inconnue**

Échantillon de la distribution a posteriori

I)

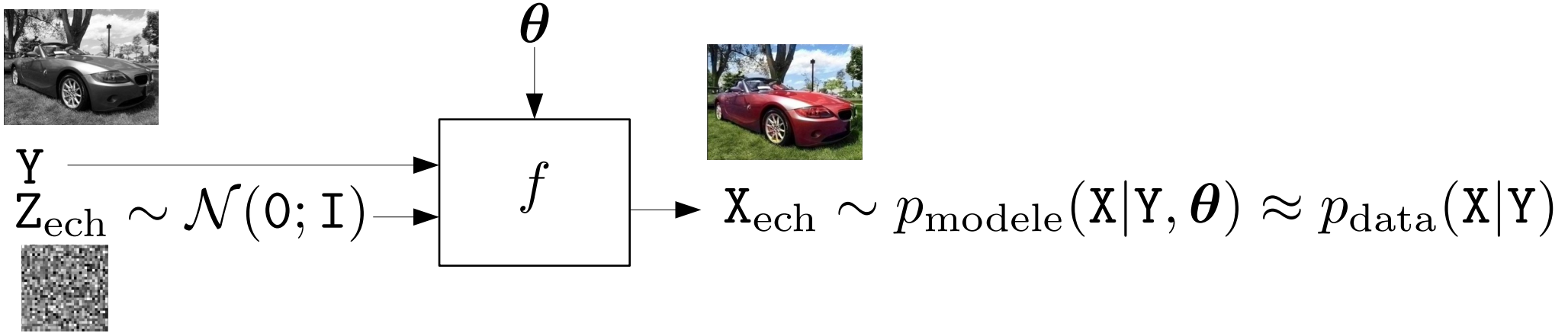
Principe d’un modèle génératif conditionnel (suite)

Objectif : Optimiser les paramètres d’un réseau de neurones afin de générer, à partir d’un $\mathtt{Y}$, de **nouveaux échantillons** de $p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$

I)

Principe d'un modèle génératif conditionnel (suite)

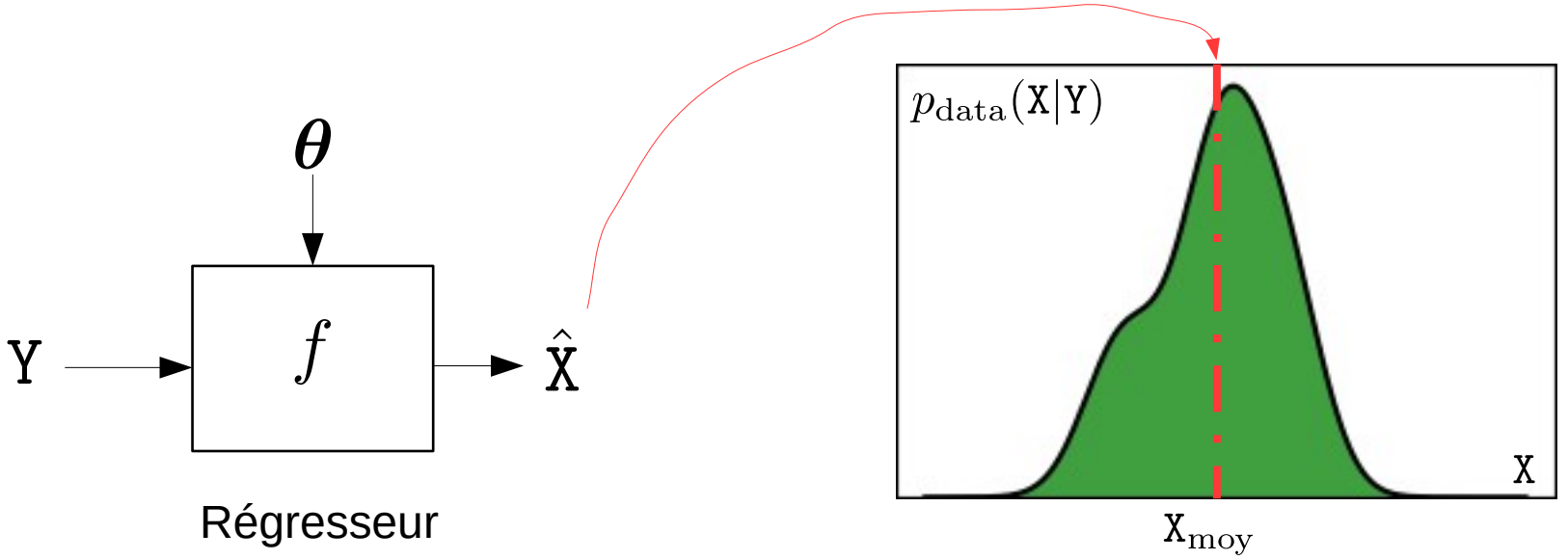
Objectif : Optimiser les paramètres d'un réseau de neurones afin de générer, à partir d'un $\mathtt{Y}$, de **nouveaux échantillons** de $p_{\text{data}}(\mathtt{X}|\mathtt{Y})$



I)

Différence vis-à-vis de l'apprentissage supervisé

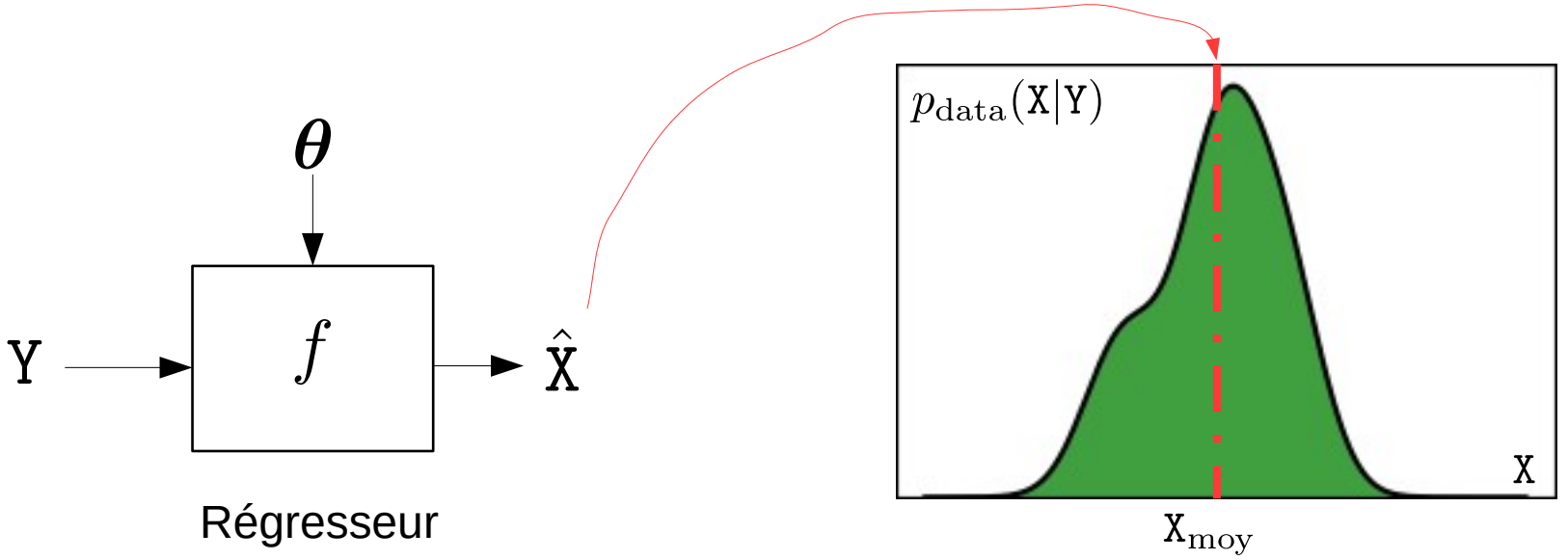
Rappel apprentissage supervisé pour de la régression



I)

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour de la régression

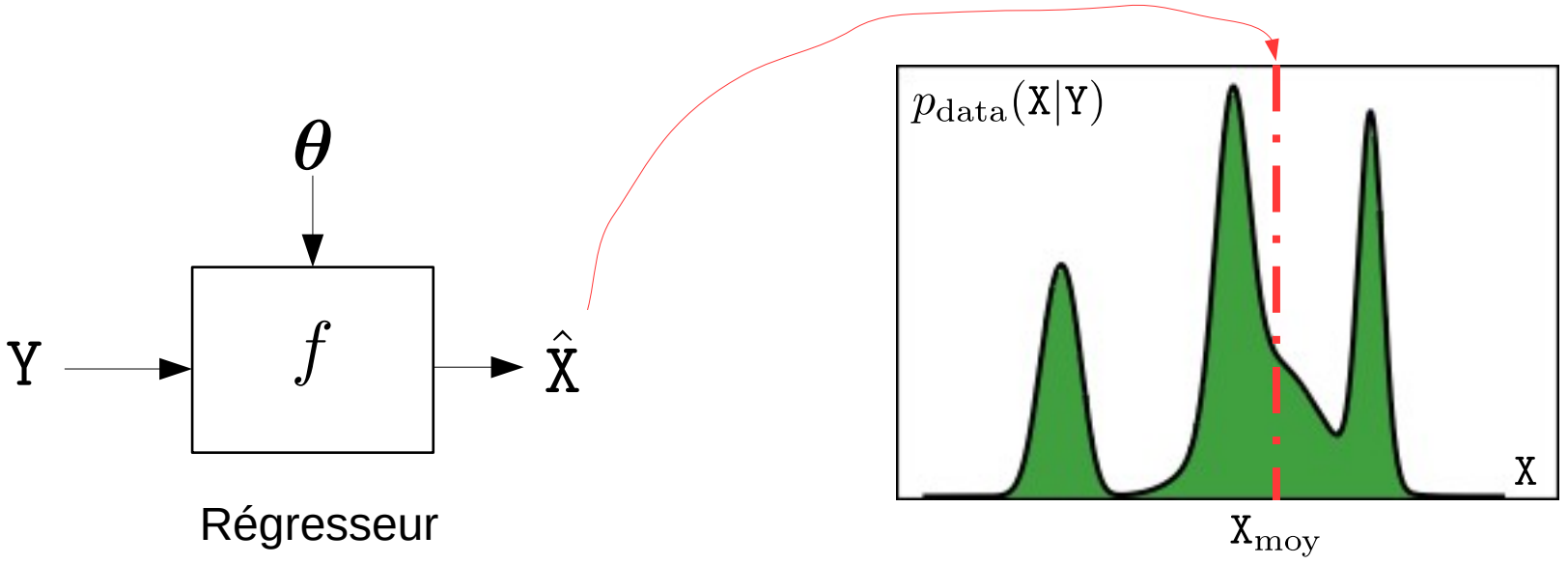


Ici, une prédiction telle que $\hat{\mathrm{X}} \approx \mathrm{X}_{\text{moy}}$ est satisfaisante car $p_{\text{data}}(\mathrm{X}|\mathrm{Y})$ est **unimodale**.

I)

Différence vis-à-vis de l'apprentissage supervisé

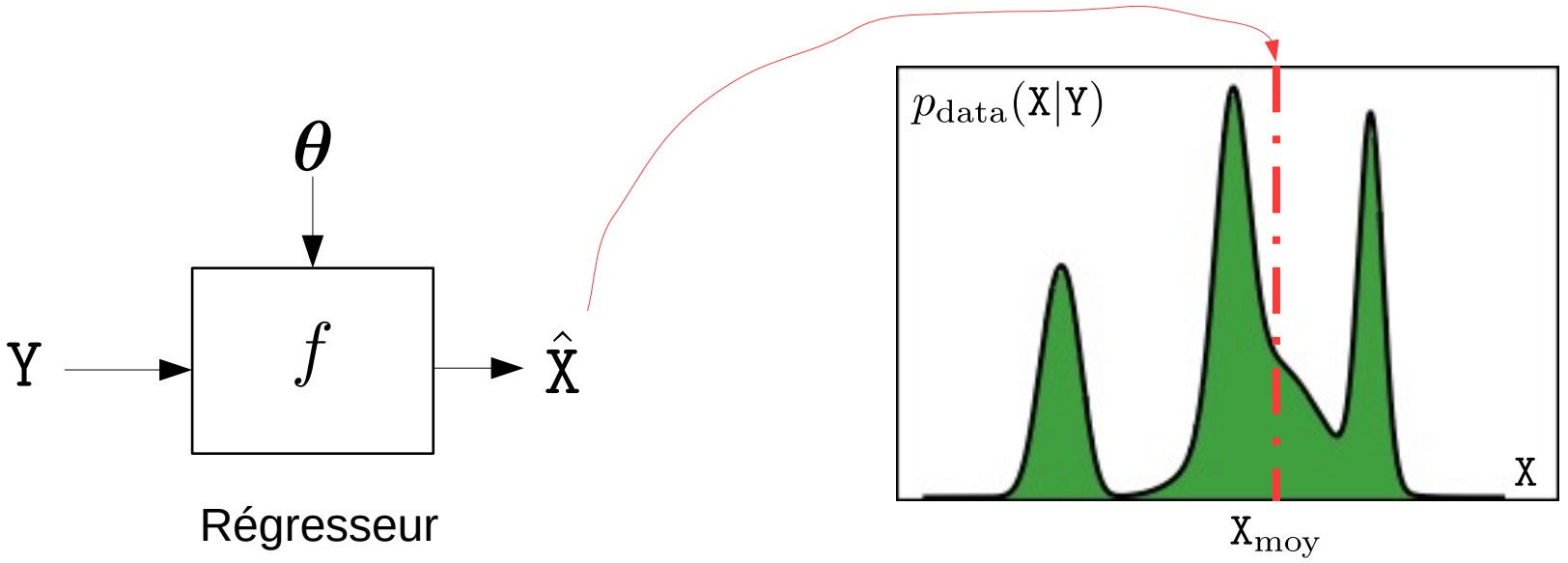
Rappel apprentissage supervisé pour de la régression



I)

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour de la régression

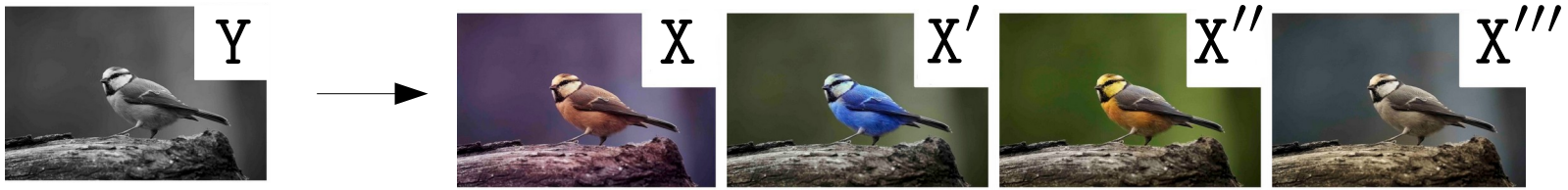


Ici, une prédiction telle que $\hat{\mathrm{X}} \approx \mathrm{X}_{\text{moy}}$ n'est pas satisfaisante car $p_{\text{data}}(\mathrm{X}|\mathrm{Y})$ est **multi-modale**.

I)

Différence vis-à-vis de l’apprentissage supervisé

Question : Que se passe-t-il si on entraîne un régresseur pour colorer l’image suivante?



I)

Différence vis-à-vis de l'apprentissage supervisé

Question : Que se passe-t-il si on entraîne un régresseur pour colorer l'image suivante?

Nécessité d'apprendre à **échantillonner** $p_{\mathrm{data}}(\mathrm{X}|\mathrm{Y})$

I)

Différence vis-à-vis de l’apprentissage supervisé

Question : Que se passe-t-il si on entraîne un régresseur pour prédire un “5” ?

“5”

I)

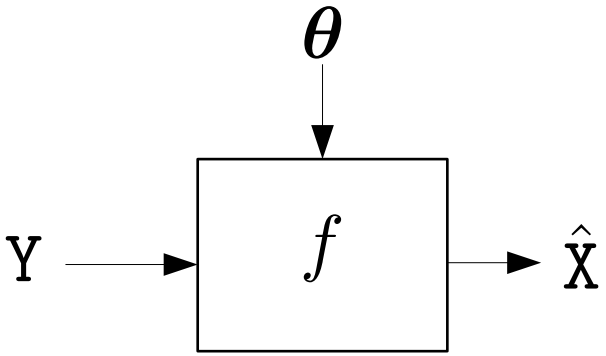
Différence vis-à-vis de l’apprentissage supervisé

Question : Que se passe-t-il si on entraîne un régresseur pour prédire un “5” ?

“5”

Nécessité d’apprendre à **échantillonner**

Différence vis-à-vis de l'apprentissage supervisé

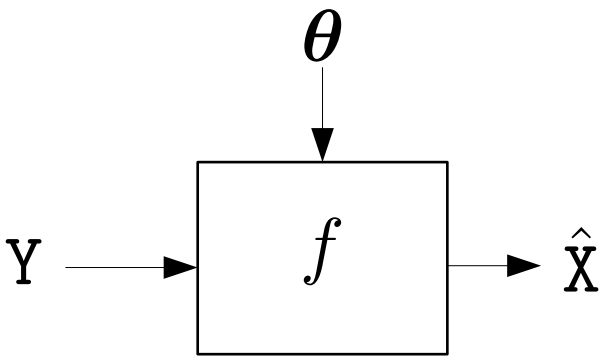


Supervisé → Régresseur

$p_{\mathrm{data}}(\mathrm{X}|\mathrm{Y})$ de forme "simple"
(ex : unimodale)

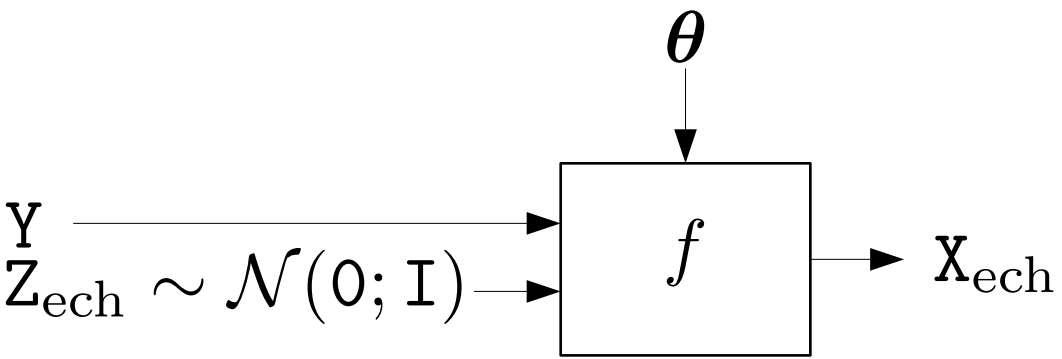
I)

Différence vis-à-vis de l'apprentissage supervisé



Supervisé → Régresseur

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$ de forme "simple" (ex : unimodale)



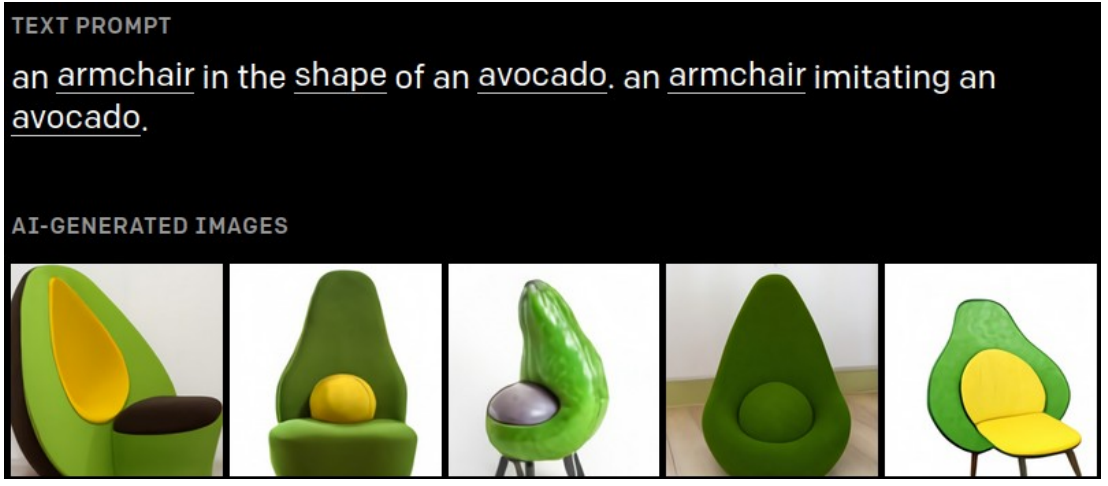
Modèle génératif → Échantillonneur

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$ de forme "complexe"

I)

Exemple d’utilisation

Édition/Synthèse de données (ex : retouche d’image)

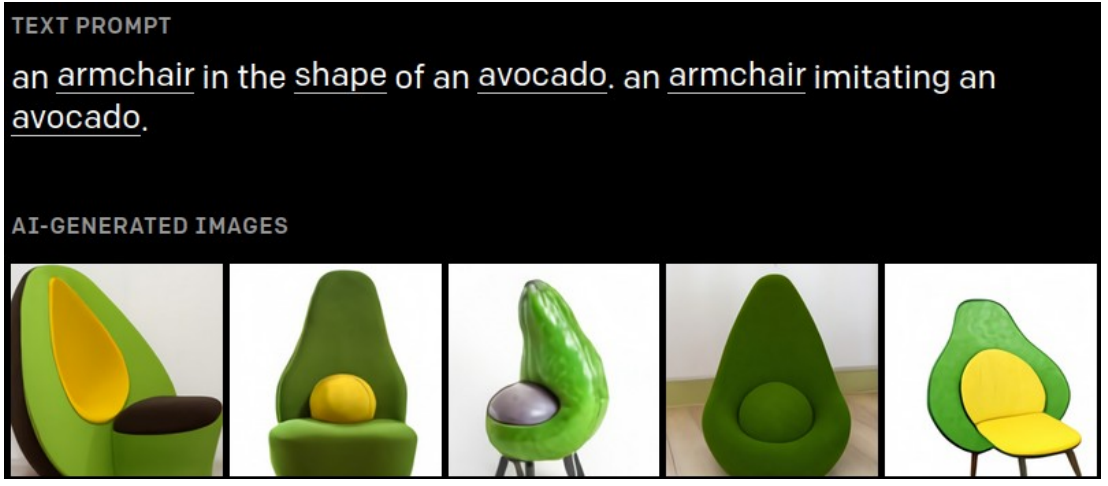


https://openai.com/blog/dall-e/

I)

Exemple d’utilisation

Édition/Synthèse de données (ex : retouche d’image)



https://openai.com/blog/dall-e/

Génération de faux contenus :

- photo
- vidéo
- signal sonore

Comment entraîner un modèle génératif profond ?

Plusieurs solutions possibles, nous allons en étudier quatre :

Comment entraîner un modèle génératif profond ?

Plusieurs solutions possibles, nous allons en étudier quatre :

- le réseau inversible (NF)

Comment entraîner un modèle génératif profond ?

Plusieurs solutions possibles, nous allons en étudier quatre :

- le réseau inversible (NF)
- l’auto-encodeur variationnel (VAE)

Comment entraîner un modèle génératif profond ?

Plusieurs solutions possibles, nous allons en étudier quatre :

- le réseau inversible (NF)
- l’auto-encodeur variationnel (VAE)
- le réseau débruiteur – méthode de diffusion (DDPM)

Comment entraîner un modèle génératif profond ?

Plusieurs solutions possibles, nous allons en étudier quatre :

- le réseau inversible (NF)
- l’auto-encodeur variationnel (VAE)
- le réseau débruiteur – méthode de diffusion (DDPM)
- le réseau antagoniste (GAN)

Comment entraîner un modèle génératif profond ?

Plusieurs solutions possibles, nous allons en étudier quatre :

- le réseau inversible (NF)
- l’auto-encodeur variationnel (VAE)
- le réseau débruiteur – méthode de diffusion (DDPM)
- le réseau antagoniste (GAN)

Nous n’aurons pas le temps d’étudier les réseaux auto-régressifs.

II) Notion de divergence

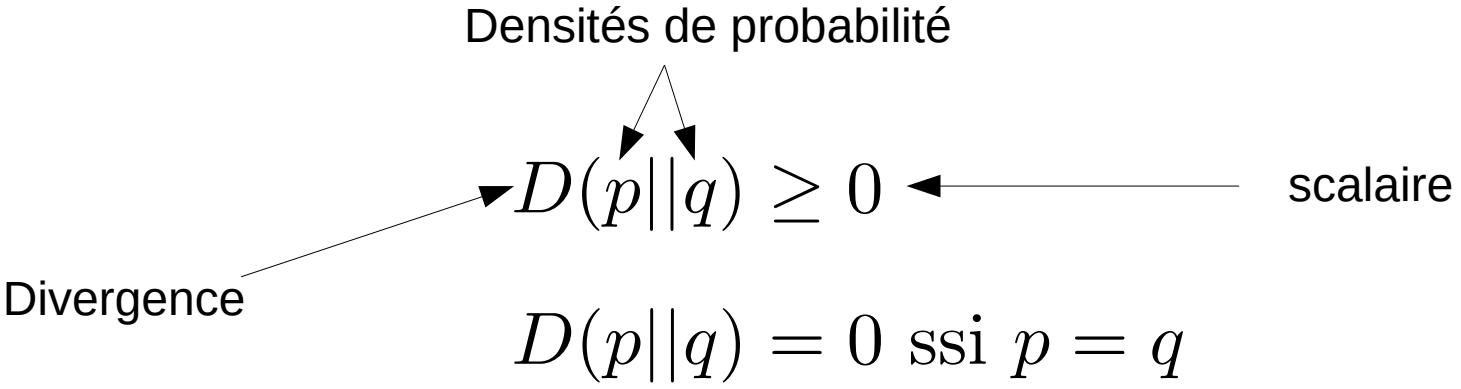
II)

Notion de divergence

Nécessité de pouvoir comparer deux densités de probabilité, par exemple pour savoir si (pour une valeur de $\boldsymbol{\theta}$) $p_{\mathrm{modele}}(\mathbf{X}|\boldsymbol{\theta})$ “ressemble” à $p_{\mathrm{data}}(\mathbf{X})$

Notion de divergence

Nécessité de pouvoir comparer deux densités de probabilité, par exemple pour savoir si (pour une valeur de $\boldsymbol{\theta}$) $p_{\mathrm{modele}}(\mathtt{X}|\boldsymbol{\theta})$ “ressemble” à $p_{\mathrm{data}}(\mathtt{X})$



Notion de divergence (suite)

La divergence de Kullback-Leibler est souvent utilisée car elle est “pratique”.

$$KL(p(x)||q(x)) = \int p(x) \ln \frac{p(x)}{q(x)} dx$$

II)

Notion de divergence (suite)

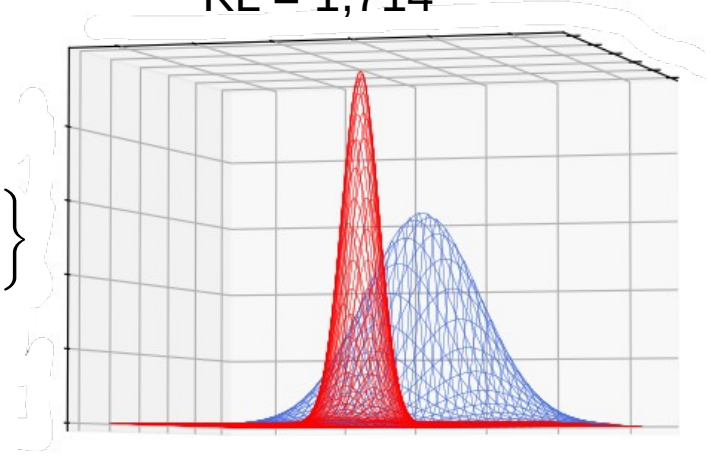
La divergence de Kullback-Leibler est souvent utilisée car elle est “pratique”.

$$KL(p(x)||q(x)) = \int p(x) \ln \frac{p(x)}{q(x)} dx$$

Exemple : Expression analytique pour deux gaussiennes $\mathcal{N}_0(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ et $\mathcal{N}_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$

$$KL(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2} \left\{ \operatorname{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}} \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - k + \ln \frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|} \right\}$$

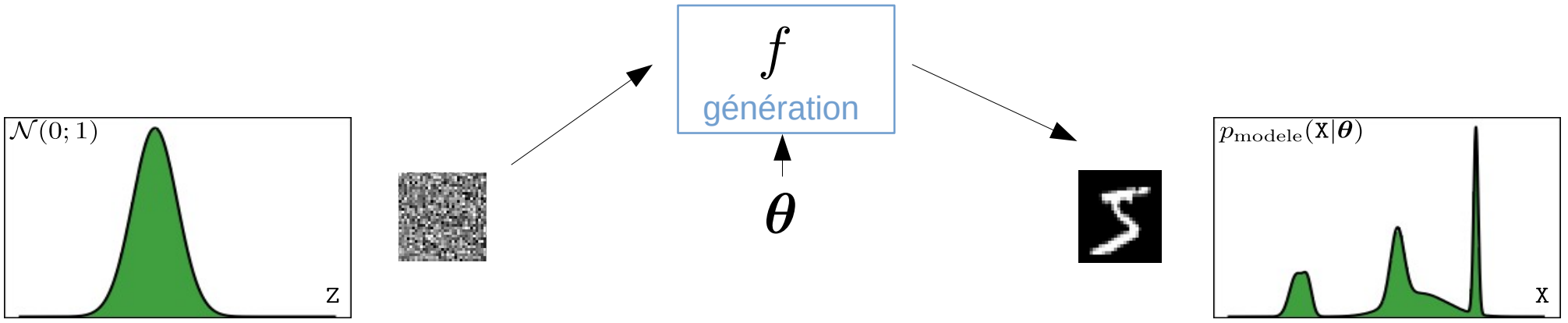
KL = 1,714



III) Réseau inversible (NF)

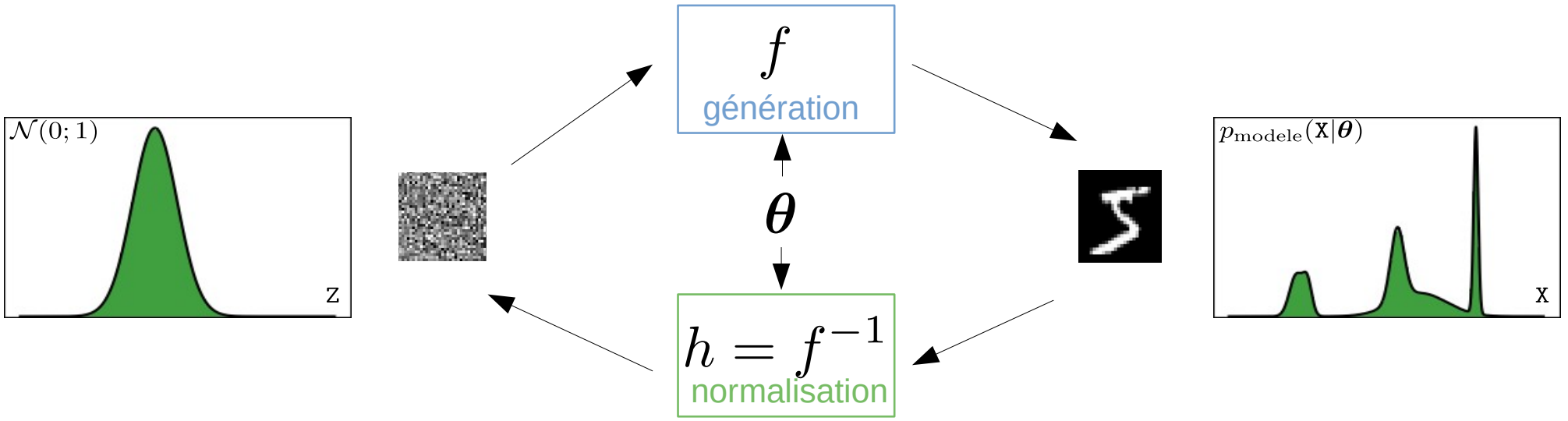
III)

Réseau inversible : idée générale



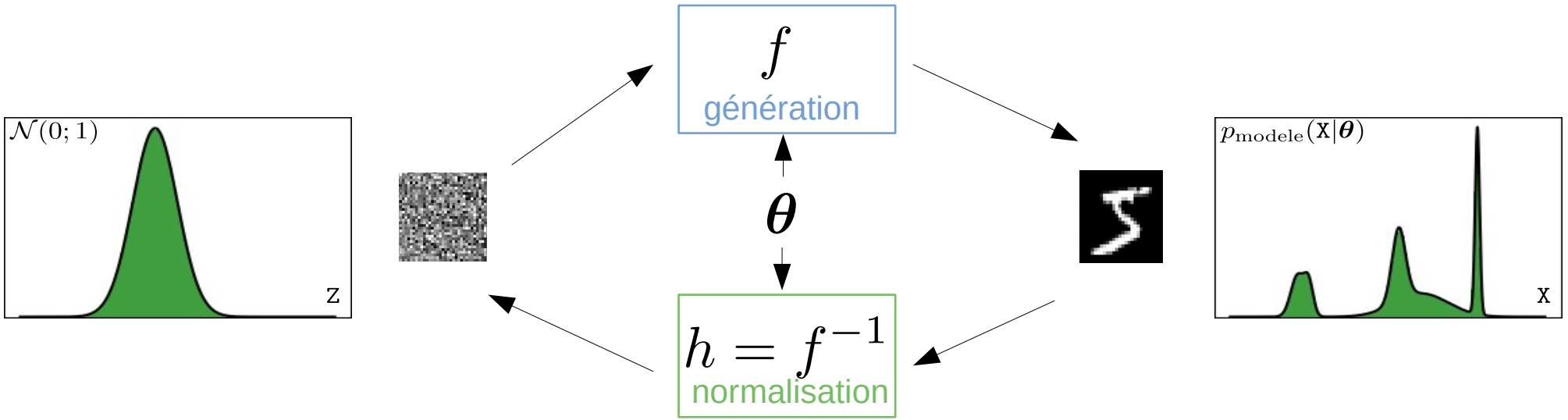
III)

Réseau inversible : idée générale



III)

Réseau inversible : idée générale

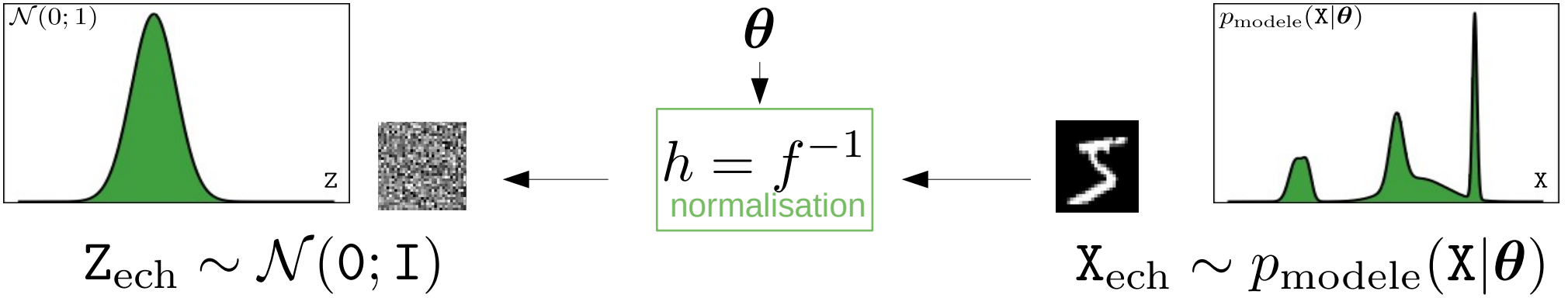


Ne pas disposer de couples $\{\mathrm{Z}_i, \mathrm{X}_i\}_{i=1\ldots N}$ n'est plus un problème.

Terminologie : “Invertible neural network”, “Normalizing Flow” (NF)

III)

Réseau inversible : formalisme

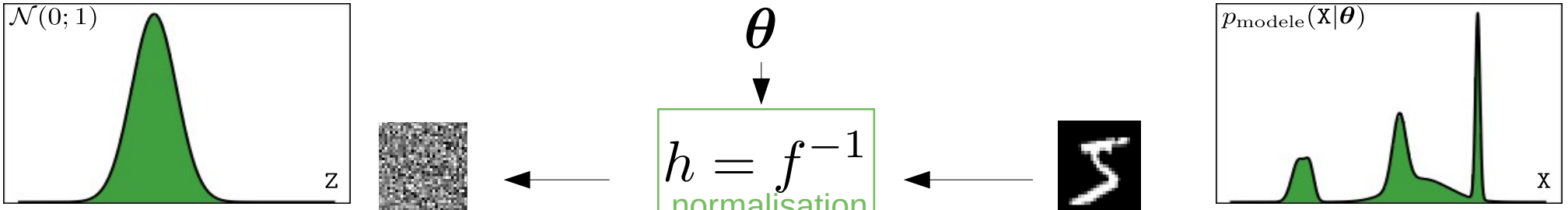


$\mathtt{Z}_{\text{ech}} \sim \mathcal{N}(0; \mathtt{I})$

$\mathtt{X}_{\text{ech}} \sim p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})$

III)

Réseau inversible : formalisme

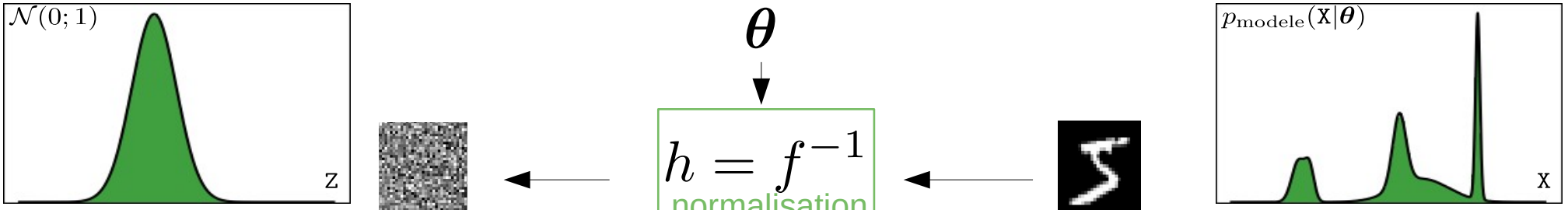


$\mathtt{Z}_{\text{ech}} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$ $\qquad$ $\mathtt{X}_{\text{ech}} \sim p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})$

Changement de variable : $\mathtt{Z} = h(\mathtt{X};\boldsymbol{\theta}) \longrightarrow d\mathtt{Z} = |\det J_h(\mathtt{X})| d\mathtt{X}$

III)

Réseau inversible : formalisme



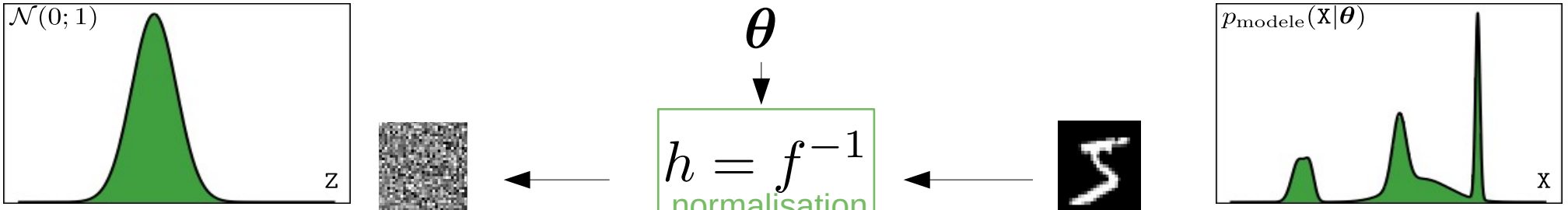
$$\mathtt{Z}_{\text{ech}} \sim \mathcal{N}(\mathtt{0};\mathtt{I}) \qquad \mathtt{X}_{\text{ech}} \sim p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})$$

Changement de variable : $\mathtt{Z} = h(\mathtt{X};\boldsymbol{\theta}) \longrightarrow d\mathtt{Z} = |\det J_h(\mathtt{X})| d\mathtt{X}$

$$1 = \int \mathcal{N}(\mathtt{Z};\mathtt{0},\mathtt{I}) d\mathtt{Z} = \int \underbrace{\mathcal{N}(h(\mathtt{X};\boldsymbol{\theta});\mathtt{0},\mathtt{I})|\det J_h(\mathtt{X})|}_{p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})} d\mathtt{X}$$

III)

Réseau inversible : formalisme (suite)



$$p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) = \mathcal{N}(h(\mathtt{X};\boldsymbol{\theta});\mathtt{0},\mathtt{I})\left|\det J_h(\mathtt{X})\right|$$

$$-\ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) = ||h(\mathtt{X};\boldsymbol{\theta})||^2 - \ln\left|\det J_h(\mathtt{X};\boldsymbol{\theta})\right| + \text{cst}_{\boldsymbol{\theta}}$$

III)

Réseau inversible : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Réseau inversible : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un NF on choisit généralement la divergence de Kullback-Leibler suivante :

$$\begin{aligned} KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) &= \int p_{\text{data}}(\mathtt{X}) \ln \frac{p_{\text{data}}(\mathtt{X})}{p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})} d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}} \\ &= -\int p_{\text{data}}(\mathtt{X}) \ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}} \\ &= \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(-\ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) + \text{cst}_{\boldsymbol{\theta}} \end{aligned}$$

Réseau inversible : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un NF on choisit généralement la divergence de Kullback-Leibler suivante :

$$KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) = \int p_{\text{data}}(\mathtt{X}) \ln \frac{p_{\text{data}}(\mathtt{X})}{p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})} d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}}$$

$$= -\int p_{\text{data}}(\mathtt{X}) \ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}}$$

$$= \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(-\ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) + \text{cst}_{\boldsymbol{\theta}}$$

Voir “Flow-GAN” pour un exemple d’utilisation d’un apprentissage antagoniste pour un NF

Réseau inversible : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un NF on choisit généralement la divergence de Kullback-Leibler suivante :

$$KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) = \int p_{\text{data}}(\mathtt{X}) \ln \frac{p_{\text{data}}(\mathtt{X})}{p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})} d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}}$$

$$= -\int p_{\text{data}}(\mathtt{X}) \ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}}$$

$$= \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(-\ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) + \text{cst}_{\boldsymbol{\theta}}$$

Voir “Flow-GAN” pour un exemple d’utilisation d’un apprentissage antagoniste pour un NF

Approximation de $\mathbb{E}_{p_{\text{data}}(\mathtt{X})}$

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} -\ln p_{\text{modele}}(\mathtt{X}_i|\boldsymbol{\theta}))$$ = maximisation de la vraisemblance

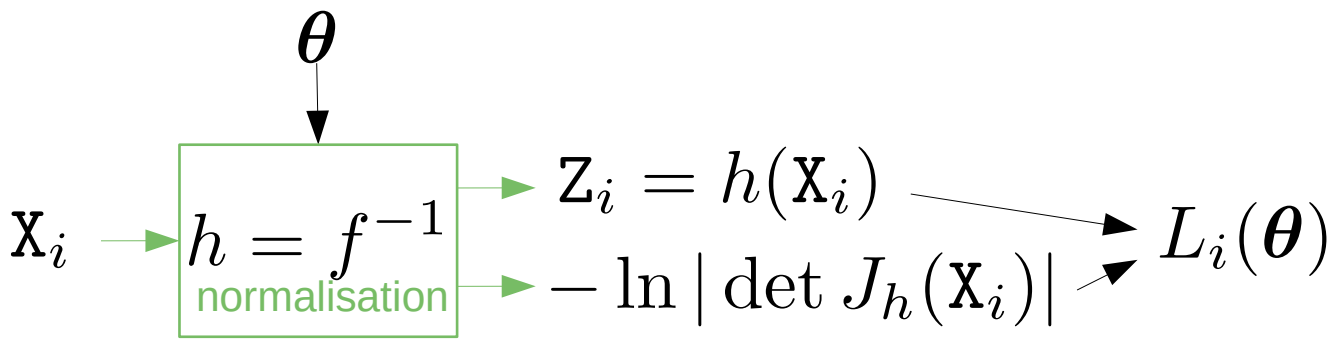
Réseau inversible : apprentissage (suite)

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} ||h(\mathtt{X}_i;\boldsymbol{\theta})||^2 - \ln|\det J_h(\mathtt{X}_i;\boldsymbol{\theta})|$$

Essaye de transformer $\mathtt{X}_i$ proche du tenseur “zéro”

Empêche que tout le monde se transforme en un tenseur “zéro”

Descente de gradient



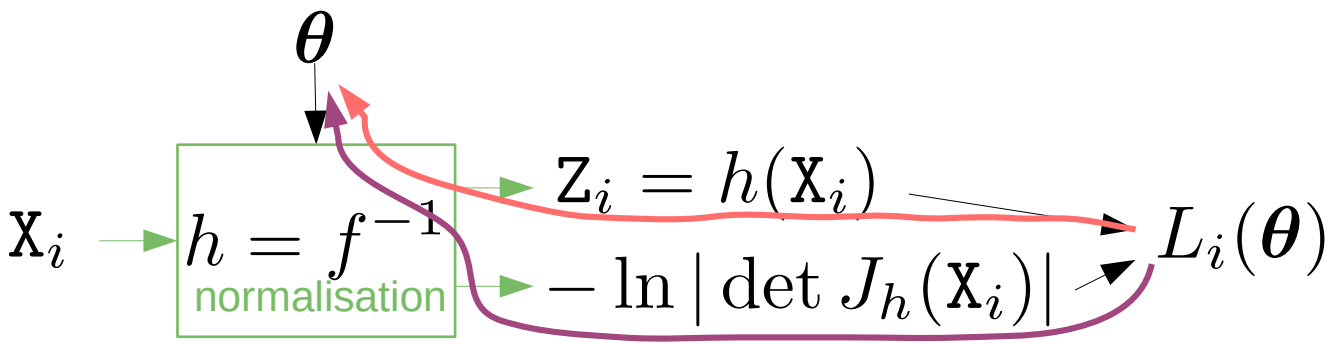
Réseau inversible : apprentissage (suite)

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} ||h(\mathtt{X}_i;\boldsymbol{\theta})||^2 - \ln|\det J_h(\mathtt{X}_i;\boldsymbol{\theta})|$$

Essaye de transformer $\mathtt{X}_i$ proche du tenseur “zéro”

Empêche que tout le monde se transforme en un tenseur “zéro”

Descente de gradient



En pratique : arrêt prématuré (“early stopping”) sur un ensemble de validation.

Réseau inversible : architecture

Besoin d'une architecture très spécifique, chaque couche doit :

1. avoir la même taille en entrée et en sortie

2. être inversible, et pour pouvoir échantillonner efficacement cette fonction inverse doit se calculer efficacement

3. avoir comme propriété que le "logdet" de sa jacobienne se calcule efficacement et soit différentiable.

Exemple de couche : "Affine coupling layer"

$$\mathtt{X}_1 = \mathtt{Z}_1$$

$$\mathtt{X}_2 = \exp(s_{\boldsymbol{\theta}}(\mathtt{Z}_1)) \odot \mathtt{Z}_2 + m_{\boldsymbol{\theta}}(\mathtt{Z}_1)$$

ResNet

jacobienne triangulaire !

Réseau inversible : avantages et inconvénients

+ capable d’échantillonner efficacement
+ capable de calculer la (log-)probabilité d’une donnée
+ apprentissage possible par maximum de vraisemblance

Réseau inversible : avantages et inconvénients

+ capable d’échantillonner efficacement
+ capable de calculer la (log-)probabilité d’une donnée
+ apprentissage possible par maximum de vraisemblance

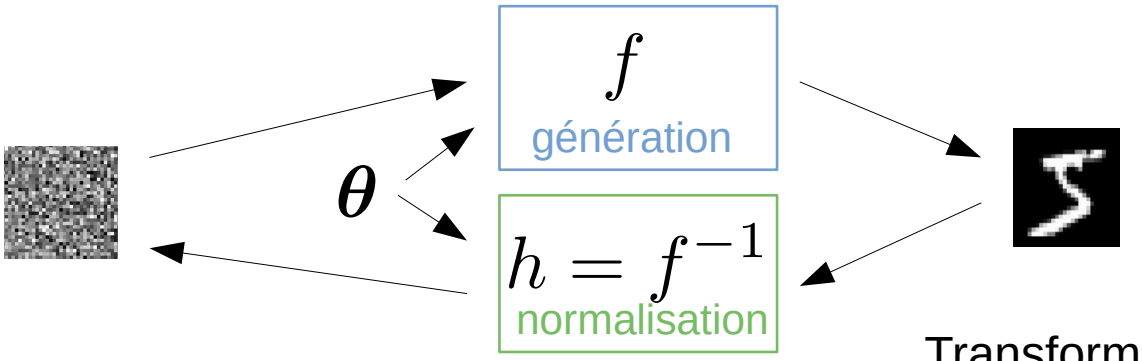
- contrainte d’architecture inversible
- contrainte du même nombre de dimensions entrée/sortie
- la forme de la distribution qu’on transforme est fortement contrainte
- **contrainte sur le calcul du log-déterminant**

IV) Auto-encodeur variationnel (VAE)

IV)

Auto-encodeur variationnel : idée générale

Réseau inversible

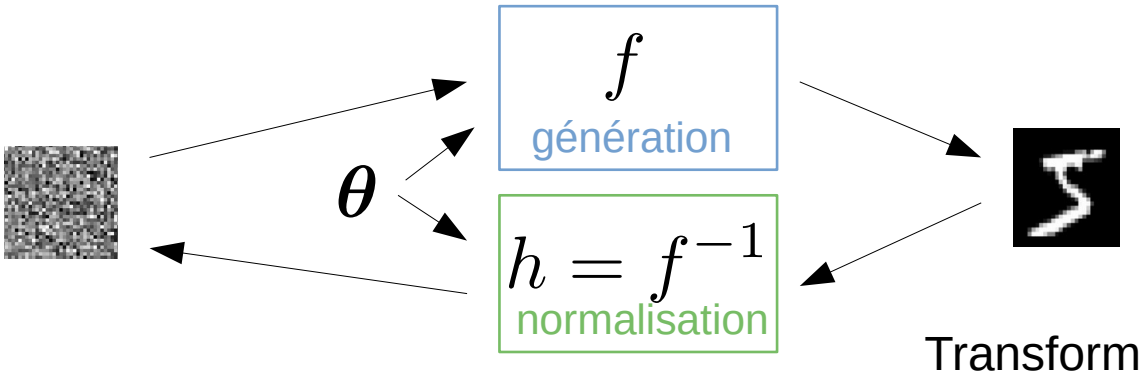


Transformations déterministes

IV)

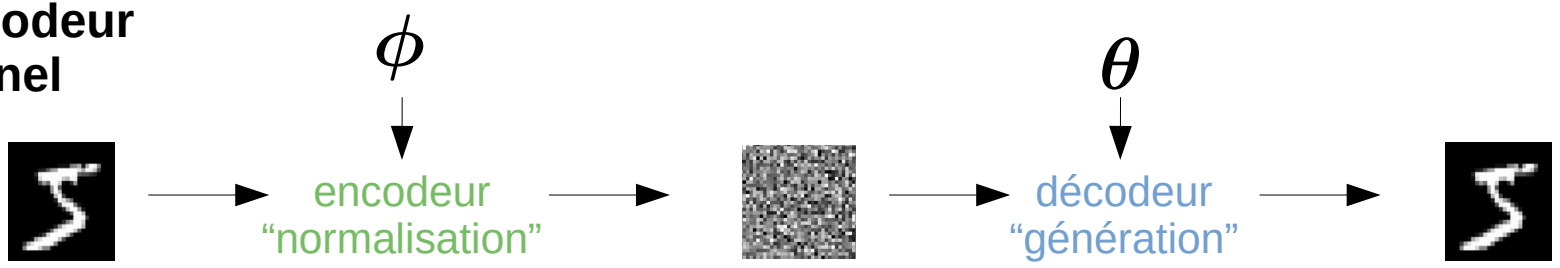
Auto-encodeur variationnel : idée générale

Réseau inversible



Transformations déterministes

Auto-encodeur variationnel

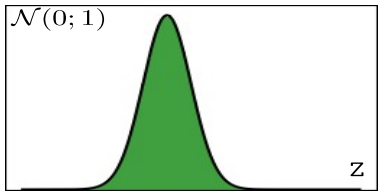


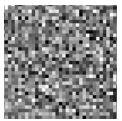
Terminologie : “Variational Auto-Encoder” (VAE)

Transformations stochastiques

IV)

Auto-encodeur variationnel : formalisme

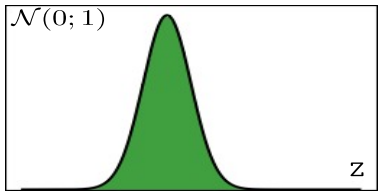


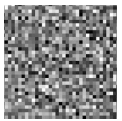


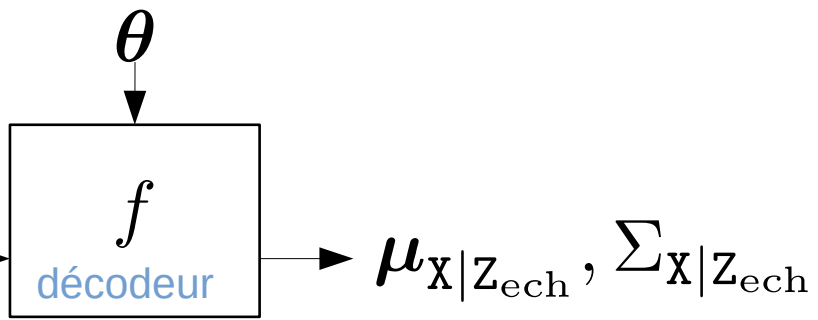
$$\mathtt{z}_{\mathrm{ech}} \sim p(\mathtt{z}) = \mathcal{N}(\mathtt{0};\mathtt{I}) \rightarrow \boxed{\begin{array}{c}\boldsymbol{\theta} \\ f \\ \text{décodeur}\end{array}} \rightarrow \boldsymbol{\mu}_{\mathtt{x}|\mathtt{z}_{\mathrm{ech}}}, \Sigma_{\mathtt{x}|\mathtt{z}_{\mathrm{ech}}}$$

IV)

Auto-encodeur variationnel : formalisme

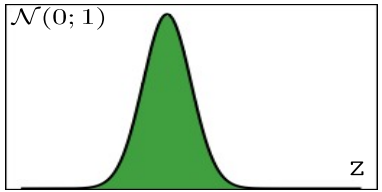


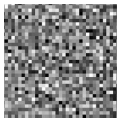




$$p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}})$$

Auto-encodeur variationnel : formalisme



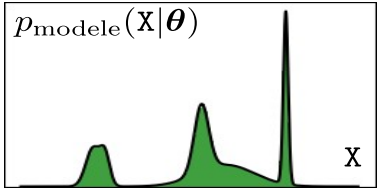


$$\mathtt{Z}_{\mathrm{ech}} \sim p(\mathtt{Z}) = \mathcal{N}(\mathtt{0};\mathtt{I}) \rightarrow \boxed{f \\ \text{décodeur}} \rightarrow \boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}$$

$\boldsymbol{\theta}$

$$p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}})$$

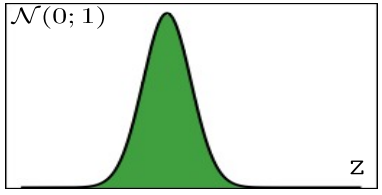
échantillon

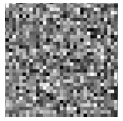


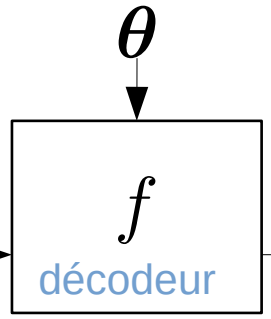
Décodeur stochastique

IV)

Auto-encodeur variationnel : formalisme



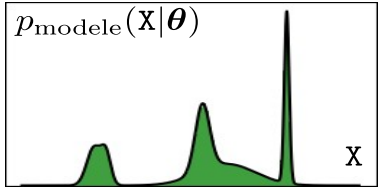




$$\mathtt{Z}_{\mathrm{ech}} \sim p(\mathtt{Z}) = \mathcal{N}(\mathtt{0};\mathtt{I}) \rightarrow f \rightarrow \boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}$$

$$p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}})$$

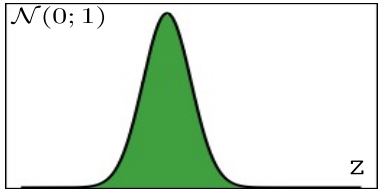
échantillon

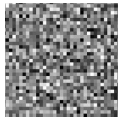


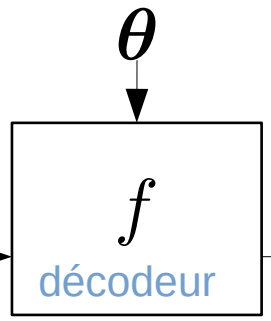
$$p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta}) = \int p_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) d\mathtt{Z} = \int p(\mathtt{Z}) p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta}) d\mathtt{Z}$$

IV)

Auto-encodeur variationnel : formalisme



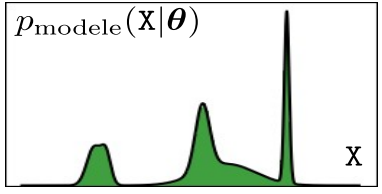




$$\mathtt{Z}_{\mathrm{ech}} \sim p(\mathtt{Z}) = \mathcal{N}(\mathtt{0};\mathtt{I}) \rightarrow f \rightarrow \boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}$$

$$p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}})$$

échantillon

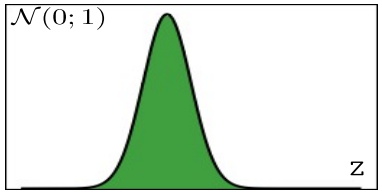


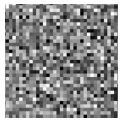
$$p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta}) = \int p_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) d\mathtt{Z} = \int p(\mathtt{Z}) p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta}) d\mathtt{Z}$$

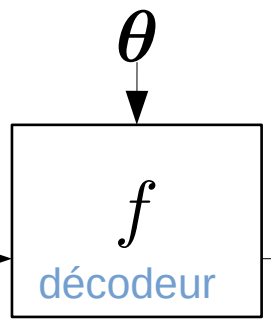
Intégrale en grande dimension, trop coûteux à calculer !

IV)

Auto-encodeur variationnel : formalisme



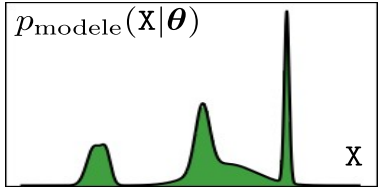




$$\mathtt{Z}_{\text{ech}} \sim p(\mathtt{Z}) = \mathcal{N}(\mathtt{0};\mathtt{I}) \rightarrow f \rightarrow \boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\text{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\text{ech}}}$$

$$p_{\text{m}}(\mathtt{X}|\mathtt{Z}_{\text{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\text{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\text{ech}}})$$

échantillon



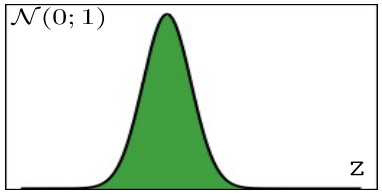
$$p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta}) = \int p_{\text{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) d\mathtt{Z} = \int p(\mathtt{Z}) p_{\text{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta}) d\mathtt{Z}$$

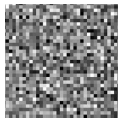
Intégrale en grande dimension, trop coûteux à calculer !

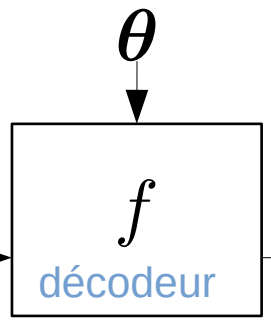
→ On ne peut pas calculer $p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})$ (on sait juste l'échantillonner)

IV)

Auto-encodeur variationnel : formalisme

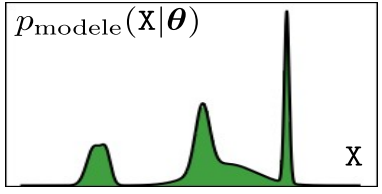






$$\mathtt{Z}_{\mathrm{ech}} \sim p(\mathtt{Z}) = \mathcal{N}(\mathtt{0};\mathtt{I}) \longrightarrow f \longrightarrow \boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}$$

$$p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}) \xrightarrow{\text{échantillon}}$$



$$p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta}) = \int p_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) d\mathtt{Z} = \int p(\mathtt{Z}) p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta}) d\mathtt{Z}$$

Intégrale en grande dimension, trop coûteux à calculer !

→ On ne peut pas calculer $p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})$ (on sait juste l'échantillonner)

→ Mais on sait calculer $p_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta})$

Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

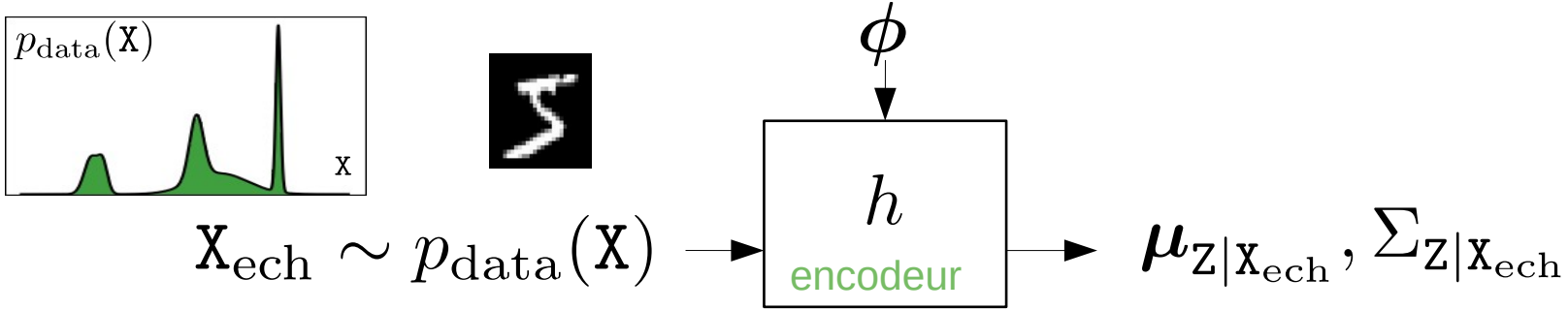
Problème : on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \rightarrow KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$

Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Problème : on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \rightarrow KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$

Pour contourner ce problème, on introduit un “encodeur”



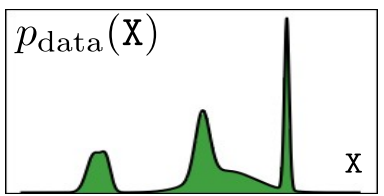
IV)

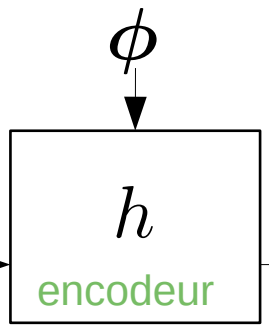
Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Problème : on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \rightarrow KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$

Pour contourner ce problème, on introduit un “encodeur”





$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X}) \rightarrow$ encodeur $\rightarrow \boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}$

$$q_{\text{m}}(\mathtt{Z}|\mathtt{X}_{\text{ech}}, \boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\text{ech}}})$$

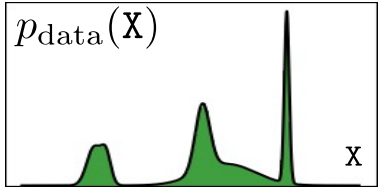
IV)

Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Problème : on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \rightarrow KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$

Pour contourner ce problème, on introduit un “encodeur”



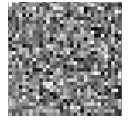
$\boldsymbol{\phi}$

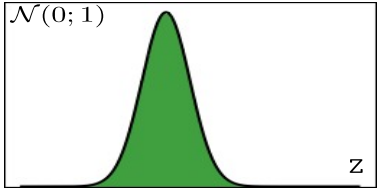
h

encodeur

$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X})$ → $\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}$

$q_{\text{m}}(\mathtt{Z}|\mathtt{X}_{\text{ech}}, \boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\text{ech}}})$ → échantillon





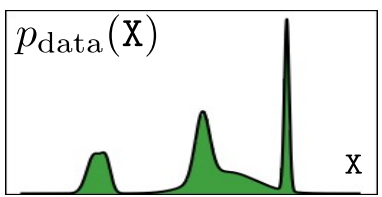
IV)

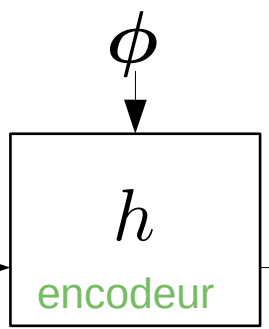
Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Problème : on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \rightarrow KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$

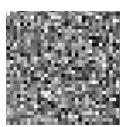
Pour contourner ce problème, on introduit un “encodeur”

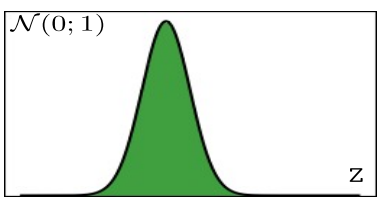




$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X})$ → $\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}$

$q_{\text{m}}(\mathtt{Z}|\mathtt{X}_{\text{ech}}, \boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\text{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\text{ech}}})$ → échantillon





Nouvel objectif (dégradé par rapport au 1er) : optimiser $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$ pour que

$$p_{\text{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\text{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta}) \approx q_{\text{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\phi}) = p_{\text{data}}(\mathtt{X})q_{\text{m}}(\mathtt{Z}|\mathtt{X}, \boldsymbol{\phi})$$

Auto-encodeur variationnel : apprentissage (suite)

Nouvel objectif : optimiser $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$ pour que

$$p_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta}) \approx q_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\phi}) = p_{\mathrm{data}}(\mathtt{X})q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X}, \boldsymbol{\phi})$$

Auto-encodeur variationnel : apprentissage (suite)

Nouvel objectif : optimiser $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$ pour que

$$p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z},\boldsymbol{\theta}) \approx q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi}) = p_{\mathrm{data}}(\mathtt{X})q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X},\boldsymbol{\phi})$$

Pour un VAE on choisit la divergence de Kullback-Leibler suivante :

$$KL(q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi})||p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}))$$

IV)

Auto-encodeur variationnel : apprentissage (suite)

Nouvel objectif : optimiser $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$ pour que

$$p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z},\boldsymbol{\theta}) \approx q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi}) = p_{\mathrm{data}}(\mathtt{X})q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X},\boldsymbol{\phi})$$

Pour un VAE on choisit la divergence de Kullback-Leibler suivante :

$$KL(q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi})||p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}))$$

$$= \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}},\Sigma_{\mathtt{Z}|\mathtt{X}})}\left(-\ln\mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}},\Sigma_{\mathtt{X}|\mathtt{Z}})\right)$$

"Erreur de reconstruction" : Incite la sortie du décodeur à ressembler aux X

$$+\mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}},\Sigma_{\mathtt{Z}|\mathtt{X}})||\mathcal{N}(\mathtt{0},\mathtt{I})))$$

"Régularisation" : Incite la sortie de l'encodeur a être centrée réduite → logique car lorsqu'on générera des z, on les tirera selon une gaussienne centrée réduite

$$+\mathrm{cst}_{\boldsymbol{\phi},\boldsymbol{\theta}}$$

IV)

Auto-encodeur variationnel : apprentissage (suite)

Nouvel objectif : optimiser $\boldsymbol{\theta}$ et $\boldsymbol{\phi}$ pour que

$$p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z},\boldsymbol{\theta}) \approx q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi}) = p_{\mathrm{data}}(\mathtt{X})q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X},\boldsymbol{\phi})$$

Pour un VAE on choisit la divergence de Kullback-Leibler suivante :

$$KL(q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi})||p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}))$$

$$= \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}},\Sigma_{\mathtt{Z}|\mathtt{X}})}\left(-\ln\mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}},\Sigma_{\mathtt{X}|\mathtt{Z}})\right)$$

"Erreur de reconstruction" : Incite la sortie du décodeur à ressembler aux X

$$+\mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}},\Sigma_{\mathtt{Z}|\mathtt{X}})||\mathcal{N}(\mathtt{0},\mathtt{I})))$$

"Régularisation" : Incite la sortie de l'encodeur a être centrée réduite → logique car lorsqu'on générera des z, on les tirera selon une gaussienne centrée réduite

$$+\mathrm{cst}_{\boldsymbol{\phi},\boldsymbol{\theta}}$$

Terminologie : ELBO ("Evidence Lower Bound")

Auto-encodeur variationnel : apprentissage (suite)

Approximation de $\mathbb{E}_{p_{\text{data}}(\mathtt{x})}$

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i})} \left(-\ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{x}|\mathtt{z}}, \Sigma_{\mathtt{x}|\mathtt{z}}) \right) + KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i}) || \mathcal{N}(\mathtt{0}, \mathtt{I}))$$

Auto-encodeur variationnel : apprentissage (suite)

Approximation de $\mathbb{E}_{p_{\text{data}}(\mathtt{x})}$

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i})} \left(-\ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{x}|\mathtt{z}}, \Sigma_{\mathtt{x}|\mathtt{z}}) \right) + KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i}) || \mathcal{N}(\mathtt{0}, \mathtt{I}))$$

Problème : $\boldsymbol{\phi}$ intervient dans cette espérance (dans le calcul de ces moyenne et covariance)

Auto-encodeur variationnel : apprentissage (suite)

Approximation de $\mathbb{E}_{p_{\text{data}}(\mathtt{x})}$

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i})} \left(-\ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{x}|\mathtt{z}}, \Sigma_{\mathtt{x}|\mathtt{z}}) \right) + KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i}) || \mathcal{N}(\mathtt{0}, \mathtt{I}))$$

Astuce de reparamétrisation (“Reparameterization trick”)

$$= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathcal{N}(\mathtt{0},\mathtt{I})} \left(-\ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{x}|(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i} + \Sigma^{1/2}_{\mathtt{z}|\mathtt{x}_i} \boldsymbol{\epsilon})}, \Sigma_{\mathtt{x}|(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i} + \Sigma^{1/2}_{\mathtt{z}|\mathtt{x}_i} \boldsymbol{\epsilon})}) \right) + KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{z}|\mathtt{x}_i}, \Sigma_{\mathtt{z}|\mathtt{x}_i}) || \mathcal{N}(\mathtt{0}, \mathtt{I}))$$

Auto-encodeur variationnel : apprentissage (suite)

En considérant un seul échantillon

$$\boldsymbol{\epsilon}_{\text{ech}} \sim \mathcal{N}(0; \mathtt{I})$$

$$L_i(\boldsymbol{\theta}, \boldsymbol{\phi}) = KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}, \Sigma_{\mathtt{Z}|\mathtt{X}_i}) || \mathcal{N}(0, \mathtt{I})) - \ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \boldsymbol{\epsilon}_{\text{ech}})}, \Sigma_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \boldsymbol{\epsilon}_{\text{ech}})})$$

Auto-encodeur variationnel : apprentissage (suite)

En considérant un seul échantillon

$$\boldsymbol{\epsilon}_{\text{ech}} \sim \mathcal{N}(0; \mathtt{I})$$

$$L_i(\boldsymbol{\theta}, \boldsymbol{\phi}) = KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}, \Sigma_{\mathtt{Z}|\mathtt{X}_i}) || \mathcal{N}(0, \mathtt{I})) - \ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \boldsymbol{\epsilon}_{\text{ech}})}, \Sigma_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \boldsymbol{\epsilon}_{\text{ech}})})$$

$\boldsymbol{\phi}$

$\mathtt{X}_i$ → h (encodeur) → $\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}, \Sigma_{\mathtt{Z}|\mathtt{X}_i}$

$\boldsymbol{\epsilon}_{\text{ech}} \sim \mathcal{N}(0; \mathtt{I})$

$\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \boldsymbol{\epsilon}_{\text{ech}}$ → f (décodeur) → $\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\text{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\text{ech}}}$ → $L_i(\boldsymbol{\theta})$

$\boldsymbol{\theta}$

Auto-encodeur variationnel : apprentissage (suite)

En considérant un seul échantillon

$$\epsilon_{\text{ech}} \sim \mathcal{N}(0; \mathtt{I})$$

$$L_i(\boldsymbol{\theta}, \boldsymbol{\phi}) = KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}, \Sigma_{\mathtt{Z}|\mathtt{X}_i}) || \mathcal{N}(0, \mathtt{I})) - \ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \epsilon_{\text{ech}})}, \Sigma_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i} \epsilon_{\text{ech}})})$$

Descente de gradient

Auto-encodeur variationnel : avantages et inconvénients

+ capable d’échantillonner efficacement
+ pas de contrainte particulière sur l’architecture
+ taille de Z non contrainte par celle de X

Auto-encodeur variationnel : avantages et inconvénients

+ capable d'échantillonner efficacement
+ pas de contrainte particulière sur l'architecture
+ taille de Z non contrainte par celle de X

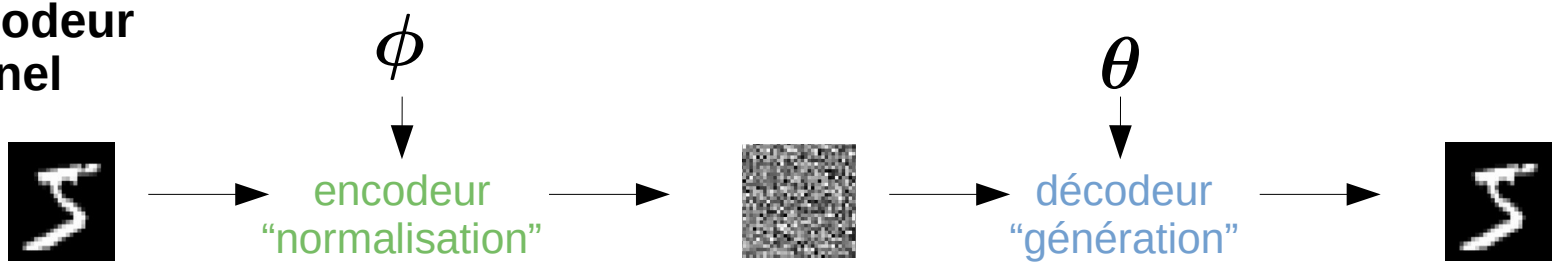
- pas d'expression exacte de la (log-)probabilité d'une donnée
- pas d'apprentissage par maximum de vraisemblance (mais borne inférieure quand même)
- contrainte sur la forme des distributions conditionnelles (exemple : covariance diagonale pour avoir un calcul rapide)

V) Réseau débruiteur – méthode de diffusion (DDPM)

V)

Réseau débruiteur : idée générale

Auto-encodeur variationnel

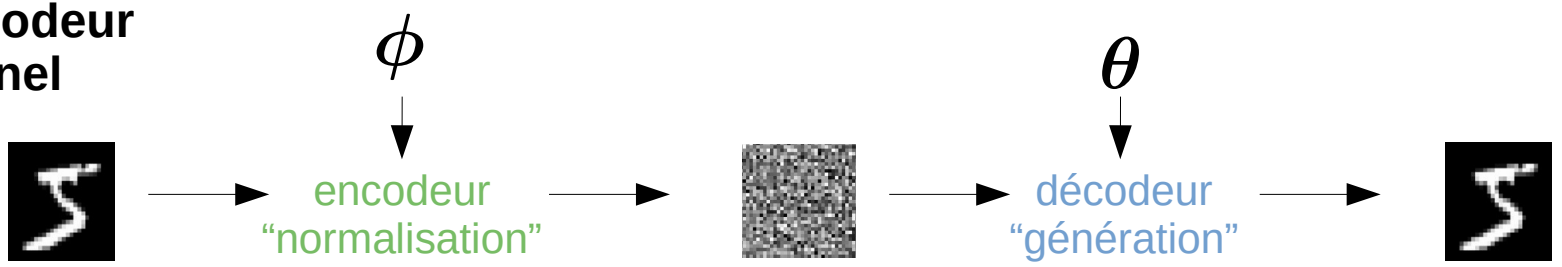


Transformations stochastiques

V)

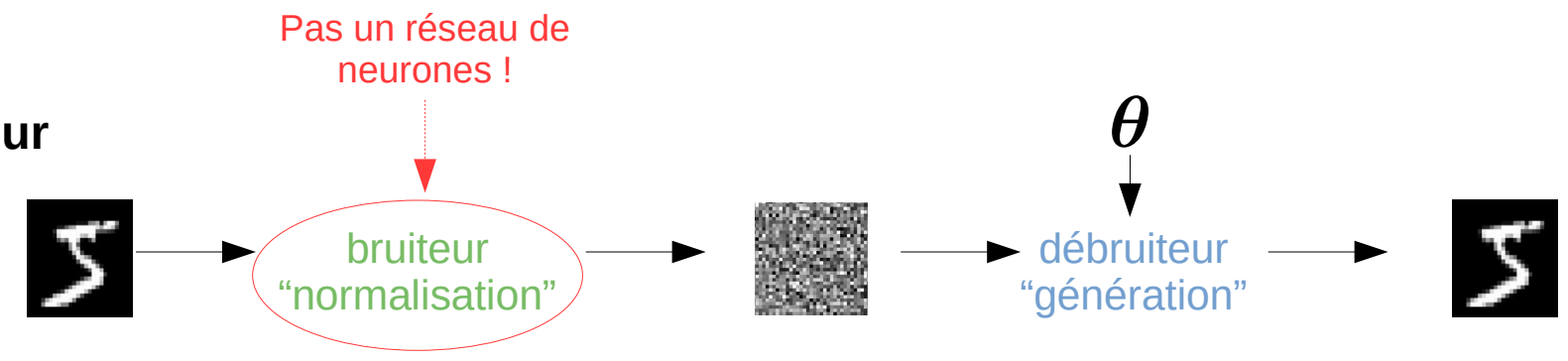
Réseau débruiteur : idée générale

Auto-encodeur variationnel



Transformations stochastiques

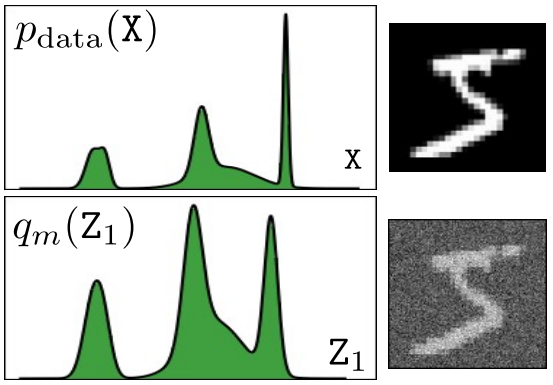
Réseau débruiteur



Terminologie : Modèle de diffusion (“Diffusion Models”)
“Denoising Diffusion Probabilistic models”

V)

Bruiteur (Diffusion)

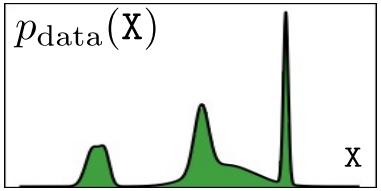


$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X})$$

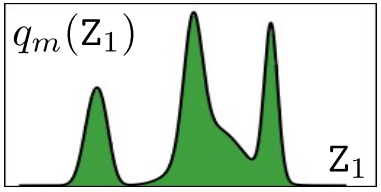
$$\mathtt{Z}_1 = \sqrt{1-\beta_1}\mathtt{X}_{\text{ech}} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \leftarrow \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

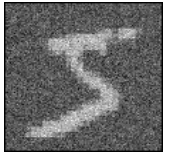
v)

Bruiteur (Diffusion)

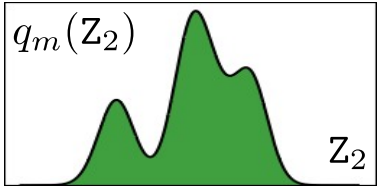


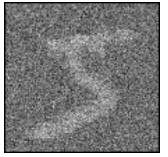
$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X})$$





$$\mathtt{Z}_1 = \sqrt{1-\beta_1}\mathtt{X}_{\text{ech}} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \leftarrow \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

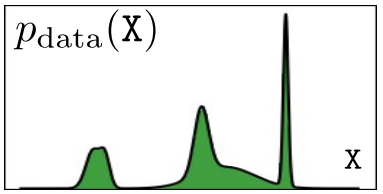




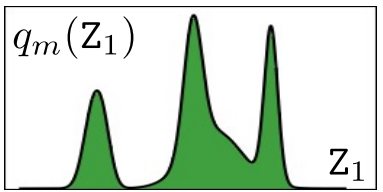
$$\mathtt{Z}_2 = \sqrt{1-\beta_2}\mathtt{Z}_1 + \sqrt{\beta_2}\boldsymbol{\epsilon}_2 \leftarrow \boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

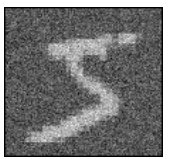
V)

Bruiteur (Diffusion)

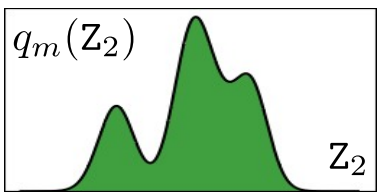


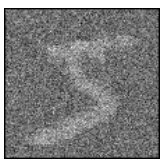
$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X})$$



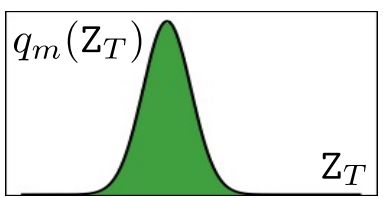


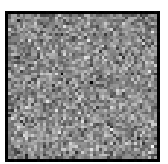
$$\mathtt{Z}_1 = \sqrt{1-\beta_1}\mathtt{X}_{\text{ech}} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \leftarrow \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$





$$\mathtt{Z}_2 = \sqrt{1-\beta_2}\mathtt{Z}_1 + \sqrt{\beta_2}\boldsymbol{\epsilon}_2 \leftarrow \boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

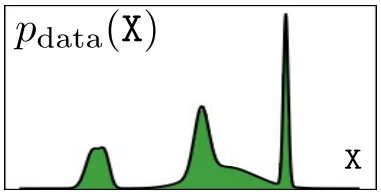




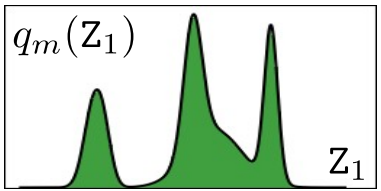
$$\mathtt{Z}_T = \sqrt{1-\beta_T}\mathtt{Z}_T + \sqrt{\beta_T}\boldsymbol{\epsilon}_T \leftarrow \boldsymbol{\epsilon}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

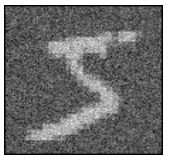
v)

Bruiteur (Diffusion)

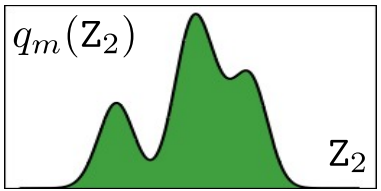


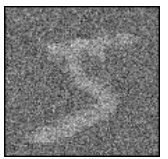
$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X})$$





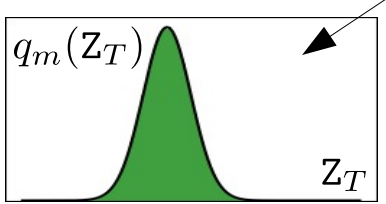
$$\mathtt{Z}_1 = \sqrt{1-\beta_1}\mathtt{X}_{\text{ech}} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1 \leftarrow \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

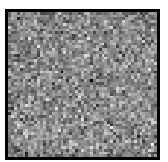




$$\mathtt{Z}_2 = \sqrt{1-\beta_2}\mathtt{Z}_1 + \sqrt{\beta_2}\boldsymbol{\epsilon}_2 \leftarrow \boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

$$\vdots \qquad \approx \mathcal{N}(\mathtt{0};\mathtt{I}) \qquad \vdots$$



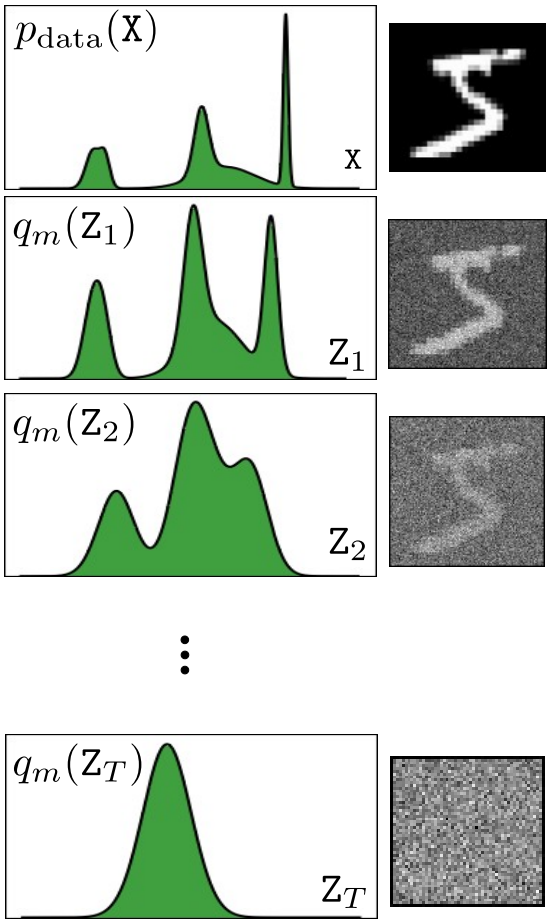


$$\mathtt{Z}_T = \sqrt{1-\beta_T}\mathtt{Z}_T + \sqrt{\beta_T}\boldsymbol{\epsilon}_T \leftarrow \boldsymbol{\epsilon}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

V)

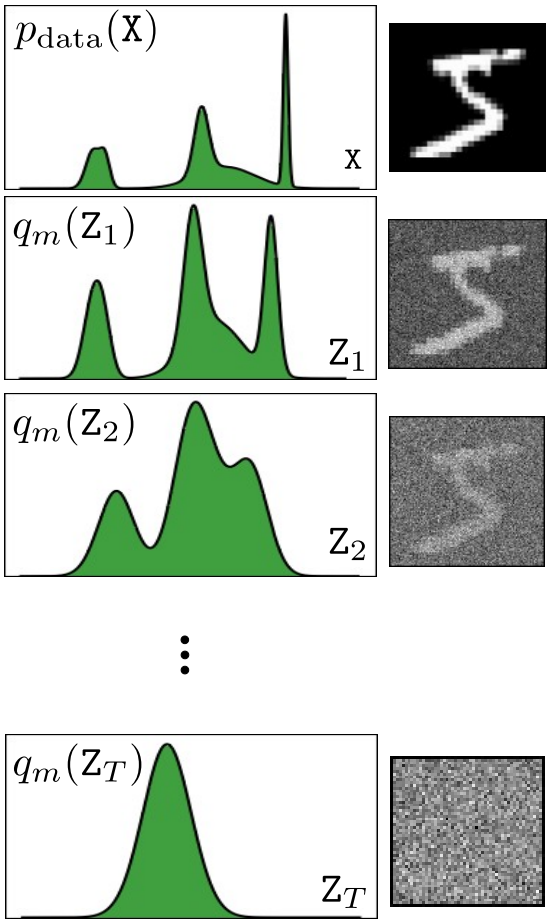
Bruiteur (Diffusion)

$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X}) \qquad \alpha_t = \prod_{k=1}^{t}(1-\beta_k)$$



V)

Bruiteur (Diffusion)

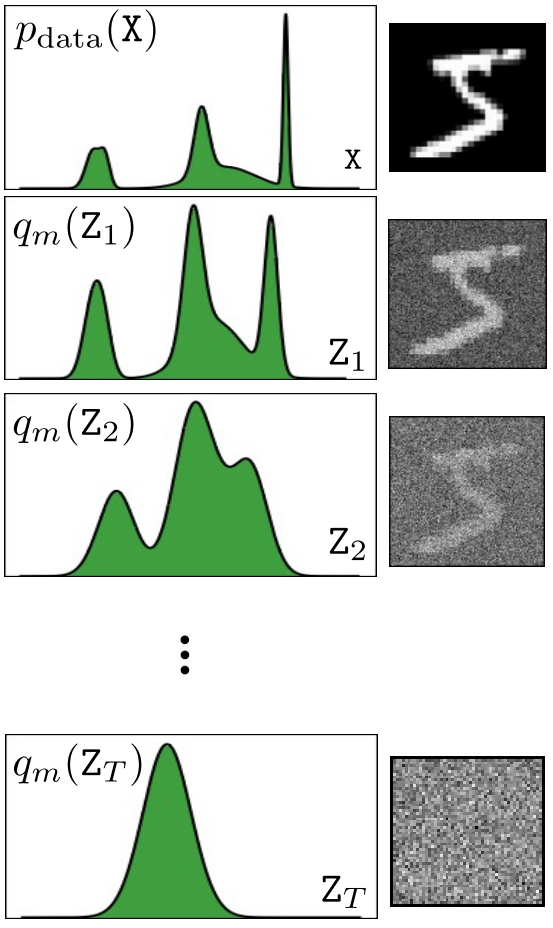


$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X}) \qquad \alpha_t = \prod_{k=1}^{t}(1-\beta_k)$$

$$\mathtt{Z}_t = \sqrt{\alpha_t}\mathtt{X}_{\text{ech}} + \sqrt{1-\alpha_t}\epsilon_t$$

V)

Bruiteur (Diffusion)



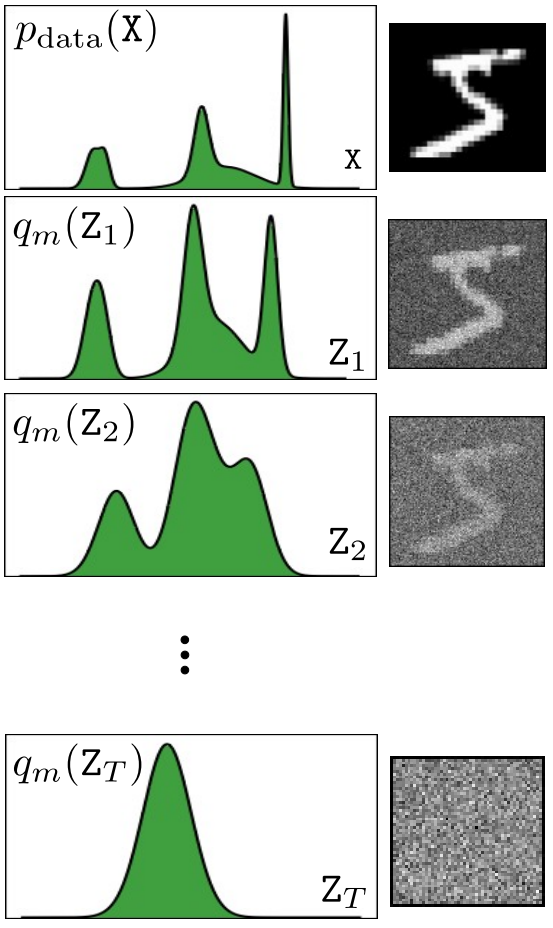
$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X}) \qquad \alpha_t = \prod_{k=1}^{t}(1-\beta_k)$$

$$\mathtt{Z}_t = \sqrt{\alpha_t}\mathtt{X}_{\text{ech}} + \sqrt{1-\alpha_t}\epsilon_t$$

$$q_m(\mathtt{Z}_t|\mathtt{X}) = \mathcal{N}\left(\sqrt{\alpha_t}\mathtt{X}, (1-\alpha_t)\,\mathtt{I}\right)$$

V)

Bruiteur (Diffusion)



$$\mathtt{X}_{\text{ech}} \sim p_{\text{data}}(\mathtt{X}) \qquad \alpha_t = \prod_{k=1}^{t}(1-\beta_k)$$

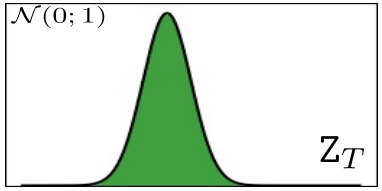
$$\mathtt{Z}_t = \sqrt{\alpha_t}\mathtt{X}_{\text{ech}} + \sqrt{1-\alpha_t}\epsilon_t$$

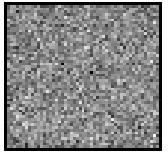
$$q_m(\mathtt{Z}_t|\mathtt{X}) = \mathcal{N}\left(\sqrt{\alpha_t}\mathtt{X}, (1-\alpha_t)\,\mathtt{I}\right)$$

Il faut choisir les $\{\beta_t\}_{t=1\ldots T}$ de telle sorte que $\prod_{k=1}^{T}(1-\beta_k) \to 0$

v)

Débruiteur

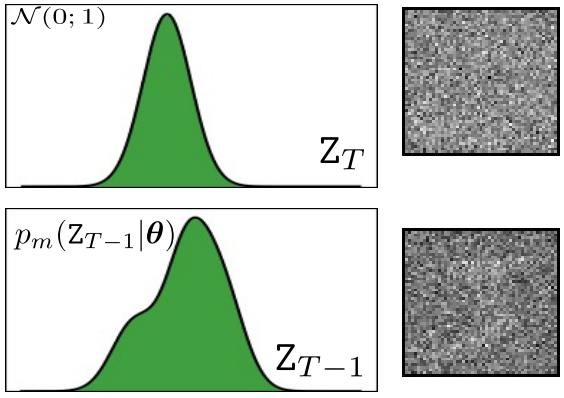




$$\mathbf{z}_T = \mathbf{n}_T \longleftarrow \mathbf{n}_T \sim \mathcal{N}(\mathbf{0}; \mathbf{I})$$

v)

Débruiteur



$$\mathtt{Z}_T = \mathbf{n}_T \qquad \mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

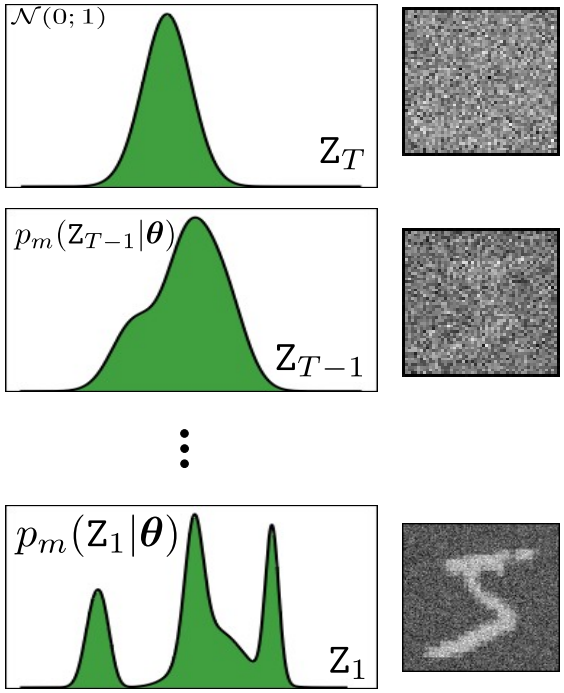
$$\mathtt{Z}_{T-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}_{T-1}}(\mathtt{Z}_T) + \sigma_{T-1}\mathbf{n}_{T-1} \qquad \mathbf{n}_{T-1} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

Fonction paramétrique
(réseau de neurones)

Paramètre
généralement fixé
manuellement

v)

Débruiteur



$$\mathtt{Z}_T = \mathbf{n}_T \longleftarrow \mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

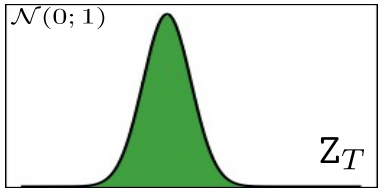
$$\mathtt{Z}_{T-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}_{T-1}}(\mathtt{Z}_T) + \sigma_{T-1}\mathbf{n}_{T-1}$$

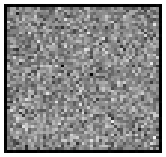
$$\mathbf{n}_{T-1} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

$$\mathtt{Z}_1 = \boldsymbol{\mu}_{\boldsymbol{\theta}_1}(\mathtt{Z}_2) + \sigma_1\mathbf{n}_1 \longleftarrow \mathbf{n}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

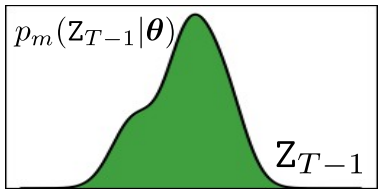
v)

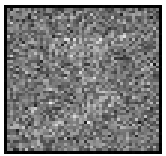
Débruiteur





$$\mathtt{Z}_T = \mathbf{n}_T \longleftarrow \mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

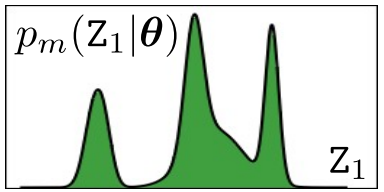




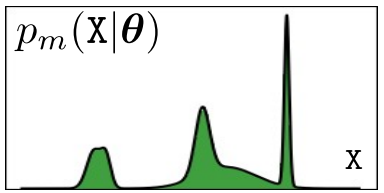
$$\mathtt{Z}_{T-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}_{T-1}}(\mathtt{Z}_T) + \sigma_{T-1}\mathbf{n}_{T-1}$$

$$\mathbf{n}_{T-1} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

⋮ ⋮



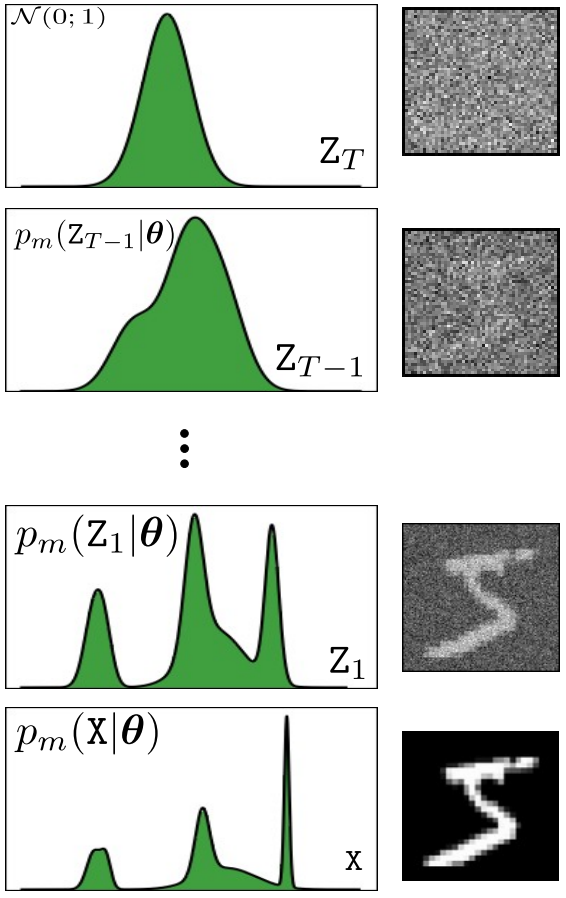
$$\mathtt{Z}_1 = \boldsymbol{\mu}_{\boldsymbol{\theta}_1}(\mathtt{Z}_2) + \sigma_1\mathbf{n}_1 \longleftarrow \mathbf{n}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$



$$\mathtt{X} = \boldsymbol{\mu}_{\boldsymbol{\theta}_0}(\mathtt{Z}_1) + \sigma_0\mathbf{n}_0 \longleftarrow \mathbf{n}_0 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

v)

Débruiteur



$$p_m(\mathtt{Z}_{t-1}|\mathtt{Z}_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}_{t-1}}(\mathtt{Z}_t), \sigma_{t-1}^2 \mathtt{I})$$

V)

Réseau débruiteur : apprentissage

Comme pour le VAE, on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})$

$$KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$$

V)

Réseau débruiteur : apprentissage

Comme pour le VAE, on ne peut pas calculer $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})$

$$KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}))$$

Comme pour le VAE, on minimise la ELBO, ce qui ici correspond à minimiser (par rapport à $\boldsymbol{\theta}$)

$$KL(q_{\text{m}}(\mathtt{X}, \mathtt{Z}_1, ..., \mathtt{Z}_T)||p_{\text{m}}(\mathtt{X}, \mathtt{Z}_1, ...\mathtt{Z}_T|\boldsymbol{\theta}))$$

V)

Réseau débruiteur : apprentissage (suite)

Paramétrisation de $\boldsymbol{\mu}_{\boldsymbol{\theta}_t}$: $\boldsymbol{\mu}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathrm{Z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t)\right)$

V)

Réseau débruiteur : apprentissage (suite)

Paramétrisation de $\boldsymbol{\mu}_{\boldsymbol{\theta}_t}$: $\boldsymbol{\mu}_{\boldsymbol{\theta}_{t-1}}(\mathtt{Z}_t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathtt{Z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\mathtt{Z}_t)\right)$

Fonction de coût (obtenue pour $\sigma_{t-1} = \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\alpha_t)}}$) :

$$L_i(\boldsymbol{\theta}) = \sum_{t=1}^{T} \| \boldsymbol{\epsilon}_{\text{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\sqrt{\alpha_t}\mathtt{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\text{ech},i,t}) \|_2^2$$

$\boldsymbol{\epsilon}_{\text{ech},i,t} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$

V)

Réseau débruiteur : apprentissage (suite)

Paramétrisation de $\boldsymbol{\mu}_{\boldsymbol{\theta}_t}$: $\boldsymbol{\mu}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathrm{Z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t)\right)$

Fonction de coût (obtenue pour $\sigma_{t-1} = \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\alpha_t)}}$) :

$$L_i(\boldsymbol{\theta}) = \sum_{t=1}^{T} \| \boldsymbol{\epsilon}_{\mathrm{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\sqrt{\alpha_t}\mathrm{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\mathrm{ech},i,t}) \|_2^2$$

$\boldsymbol{\epsilon}_{\mathrm{ech},i,t} \sim \mathcal{N}(0;\mathrm{I})$

Réseau de neurones qui apprend à prédire le bruit $\boldsymbol{\epsilon}_{\mathrm{ech},i,t}$

V)

Réseau débruiteur : apprentissage (suite)

Paramétrisation de $\boldsymbol{\mu}_{\boldsymbol{\theta}_t}$: $\boldsymbol{\mu}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathrm{Z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t)\right)$

Fonction de coût (obtenue pour $\sigma_{t-1} = \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\alpha_t)}}$) :

$$L_i(\boldsymbol{\theta}) = \sum_{t=1}^{T} \| \boldsymbol{\epsilon}_{\mathrm{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\sqrt{\alpha_t}\mathrm{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\mathrm{ech},i,t}) \|_2^2$$

$\boldsymbol{\epsilon}_{\mathrm{ech},i,t} \sim \mathcal{N}(0; \mathrm{I})$

Un réseau de neurones par pas de temps !

→ T réseaux de neurones à apprendre …

V)

Réseau débruiteur : apprentissage (suite)

Paramétrisation de $\boldsymbol{\mu}_{\boldsymbol{\theta}_t}$: $\boldsymbol{\mu}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathrm{Z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\mathrm{Z}_t)\right)$

Fonction de coût (obtenue pour $\sigma_{t-1} = \frac{\beta_t}{\sqrt{(1-\beta_t)(1-\alpha_t)}}$) :

$$L_i(\boldsymbol{\theta}) = \sum_{t=1}^{T} \| \boldsymbol{\epsilon}_{\mathrm{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}_{t-1}}(\sqrt{\alpha_t}\mathrm{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\mathrm{ech},i,t}) \|_2^2$$

$\boldsymbol{\epsilon}_{\mathrm{ech},i,t} \sim \mathcal{N}(0;\mathrm{I})$

En pratique, un seul réseau de la forme $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathrm{Z}_t, t-1)$

V)

Réseau débruiteur : apprentissage (suite)

$$L_i(\boldsymbol{\theta}) = \sum_{t=1}^{T} \| \boldsymbol{\epsilon}_{\text{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\sqrt{\alpha_t}\mathtt{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\text{ech},i,t}, t-1) \|_2^2$$

$$\boldsymbol{\epsilon}_{\text{ech},i,t} \sim \mathcal{N}(\mathtt{0}; \mathtt{I})$$

V)

Réseau débruiteur : apprentissage (suite)

$$L_i(\boldsymbol{\theta}) = \sum_{t=1}^{T} \| \boldsymbol{\epsilon}_{\text{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\sqrt{\alpha_t}\mathtt{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\text{ech},i,t}, t-1) \|_2^2$$

$$\boldsymbol{\epsilon}_{\text{ech},i,t} \sim \mathcal{N}(\mathtt{0}; \mathtt{I})$$

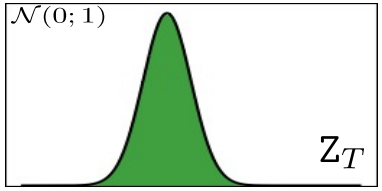
Apprentissage par descente de gradient stochastique

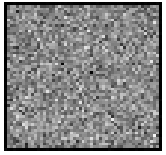
$$\sum_{i \in \Omega_{i,\text{data}}} \sum_{t \in \Omega_{i,\text{time}}} \| \boldsymbol{\epsilon}_{\text{ech},i,t} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\sqrt{\alpha_t}\mathtt{X}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\text{ech},i,t}, t-1) \|_2^2$$

Minibatch sur les données et les pas de temps → méthode “simulation free” !

V)

Résumé de l’étape de génération

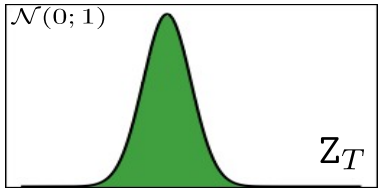


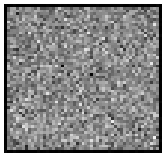


$\mathtt{Z}_T = \mathbf{n}_T$ ⟵ $\mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$

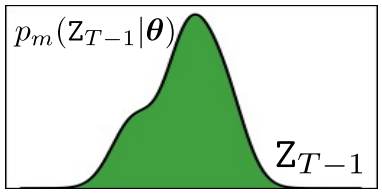
V)

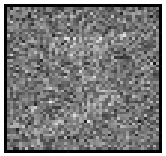
Résumé de l'étape de génération





$$\mathtt{Z}_T = \mathbf{n}_T \qquad \mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$



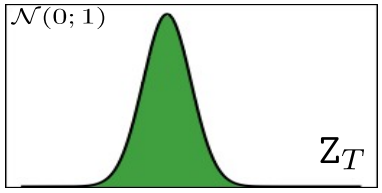


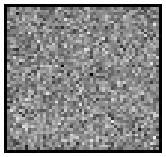
$$\mathtt{Z}_{T-1} = \frac{1}{\sqrt{1-\beta_T}}\left(\mathtt{Z}_T - \frac{\beta_T}{\sqrt{1-\alpha_T}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathtt{Z}_T, T-1)\right) + \frac{\beta_T}{\sqrt{(1-\beta_T)(1-\alpha_T)}}\mathbf{n}_{T-1}$$

$$\mathbf{n}_{T-1} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

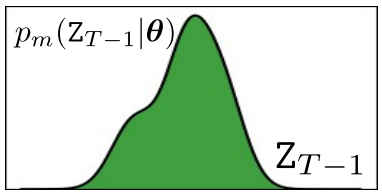
V)

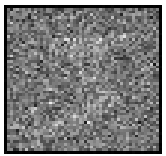
Résumé de l'étape de génération





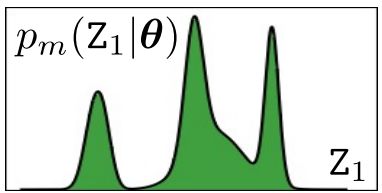
$$\mathtt{Z}_T = \mathbf{n}_T \qquad \mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

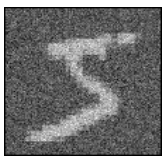




$$\mathbf{n}_{T-1} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

$$\mathtt{Z}_{T-1} = \frac{1}{\sqrt{1-\beta_T}}\left(\mathtt{Z}_T - \frac{\beta_T}{\sqrt{1-\alpha_T}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathtt{Z}_T, T-1)\right) + \frac{\beta_T}{\sqrt{(1-\beta_T)(1-\alpha_T)}}\mathbf{n}_{T-1}$$



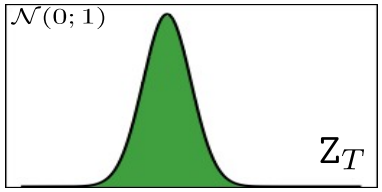


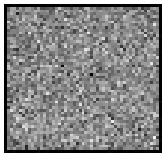
$$\mathbf{n}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

$$\mathtt{Z}_1 = \frac{1}{\sqrt{1-\beta_2}}\left(\mathtt{Z}_2 - \frac{\beta_2}{\sqrt{1-\alpha_2}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathtt{Z}_2, 1)\right) + \frac{\beta_2}{\sqrt{(1-\beta_2)(1-\alpha_2)}}\mathbf{n}_1$$

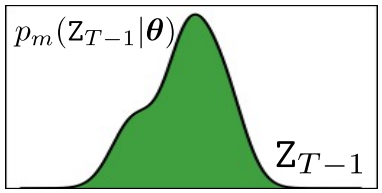
V)

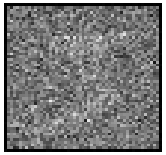
Résumé de l'étape de génération





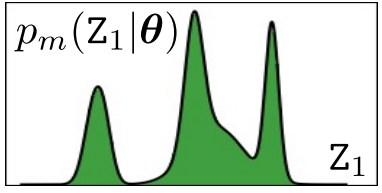
$$\mathtt{Z}_T = \mathbf{n}_T \longleftarrow \mathbf{n}_T \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

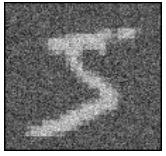




$$\mathtt{Z}_{T-1} = \frac{1}{\sqrt{1-\beta_T}}\left(\mathtt{Z}_T - \frac{\beta_T}{\sqrt{1-\alpha_T}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathtt{Z}_T, T-1)\right) + \frac{\beta_T}{\sqrt{(1-\beta_T)(1-\alpha_T)}}\mathbf{n}_{T-1}$$

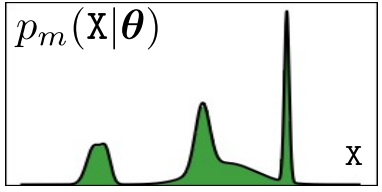
$$\mathbf{n}_{T-1} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$





$$\mathtt{Z}_1 = \frac{1}{\sqrt{1-\beta_2}}\left(\mathtt{Z}_2 - \frac{\beta_2}{\sqrt{1-\alpha_2}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathtt{Z}_2, 1)\right) + \frac{\beta_2}{\sqrt{(1-\beta_2)(1-\alpha_2)}}\mathbf{n}_1$$

$$\mathbf{n}_1 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$



$$\mathtt{X} = \frac{1}{\sqrt{1-\beta_1}}\left(\mathtt{Z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathtt{Z}_1, 0)\right) + \frac{\beta_1}{\sqrt{(1-\beta_1)(1-\alpha_1)}}\mathbf{n}_0$$

$$\mathbf{n}_0 \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

Réseau débruiteur : avantages et inconvénients

+ pas d’encodeur à apprendre
+ pas de contrainte particulière sur l’architecture
+ méthode “simulation free” (lors de l’apprentissage on ne fait pas toute l’inférence, juste des petits morceaux)

Réseau débruiteur : avantages et inconvénients

+ pas d'encodeur à apprendre
+ pas de contrainte particulière sur l'architecture
+ méthode "simulation free" (lors de l'apprentissage on ne fait pas toute l'inférence, juste des petits morceaux)

- pas d'expression exacte de la (log-)probabilité d'une donnée
- pas d'apprentissage par maximum de vraisemblance (mais borne inférieure quand même)
- échantillonnage lent (T généralement grand, Z_t de la taille de X)

Réseau débruiteur : réduction du coût calculatoire/mémoire

“Stable diffusion” - High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022

1) entraîner un VAE pour encoder les images dans un espace latent de plus faible dimension

2) entraîner un réseau débruiteur sur cet espace latent

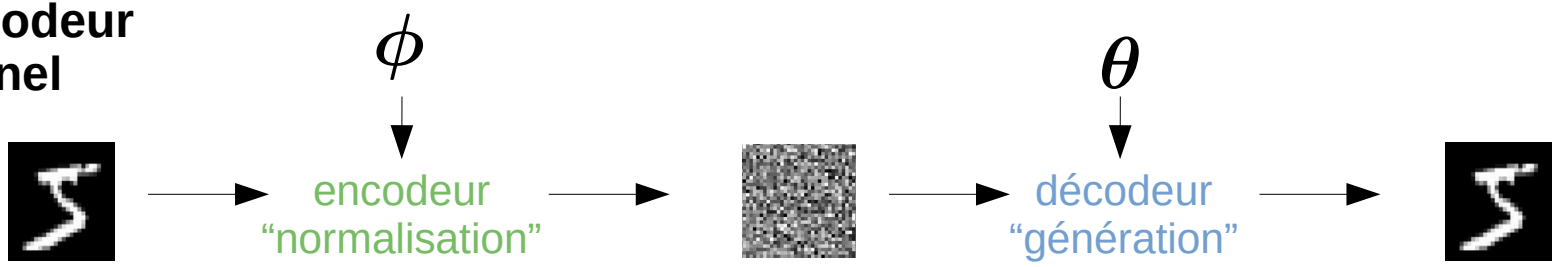
Exemple de super-résolution

VI) Réseau antagoniste (GAN)

VI)

Réseau antagoniste : idée générale

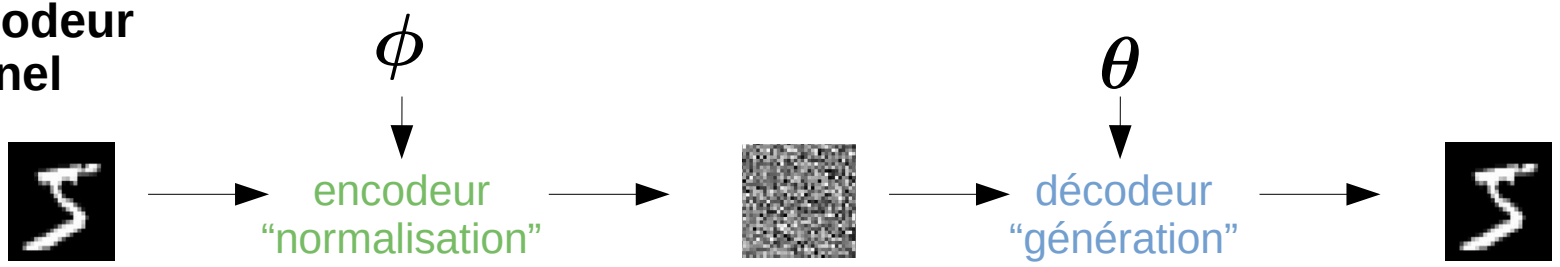
Auto-encodeur variationnel



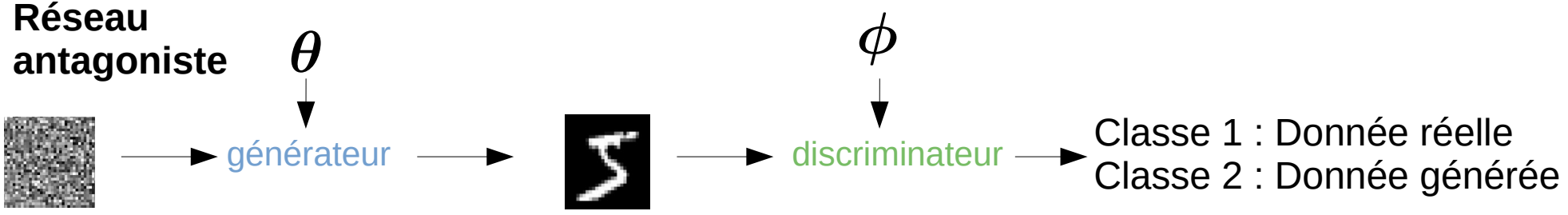
VI)

Réseau antagoniste : idée générale

Auto-encodeur variationnel

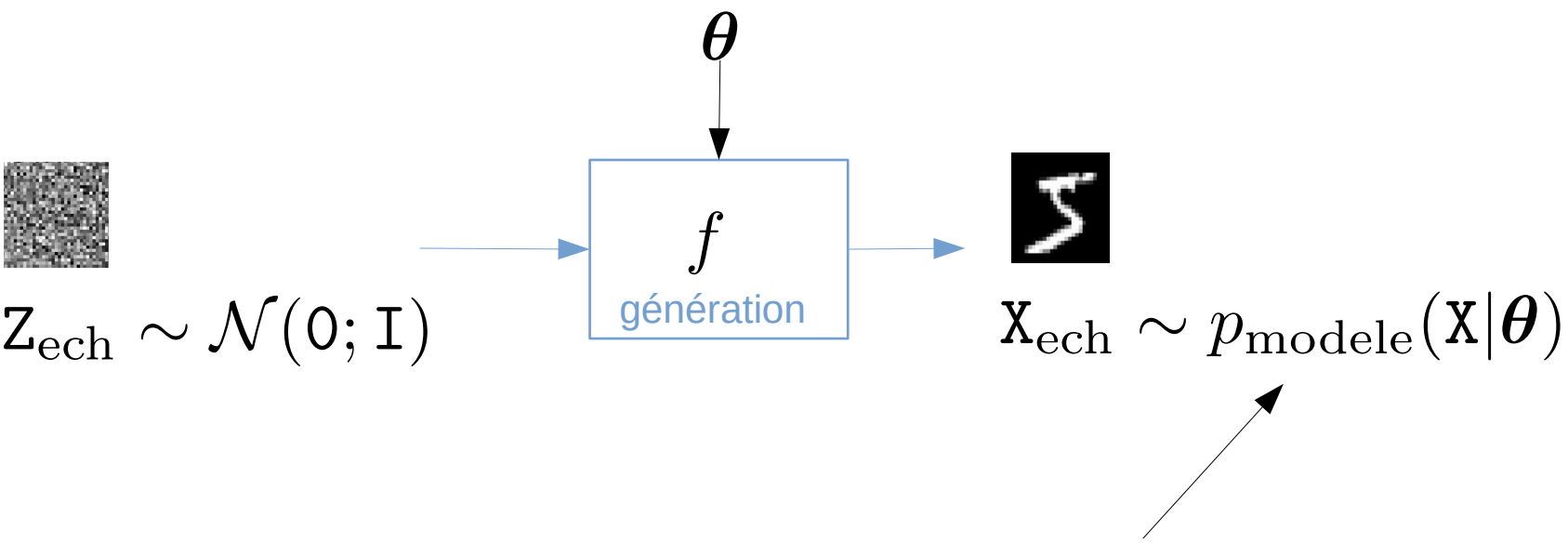


Réseau antagoniste



Terminologie : “Generative Adversarial Network” (GAN)

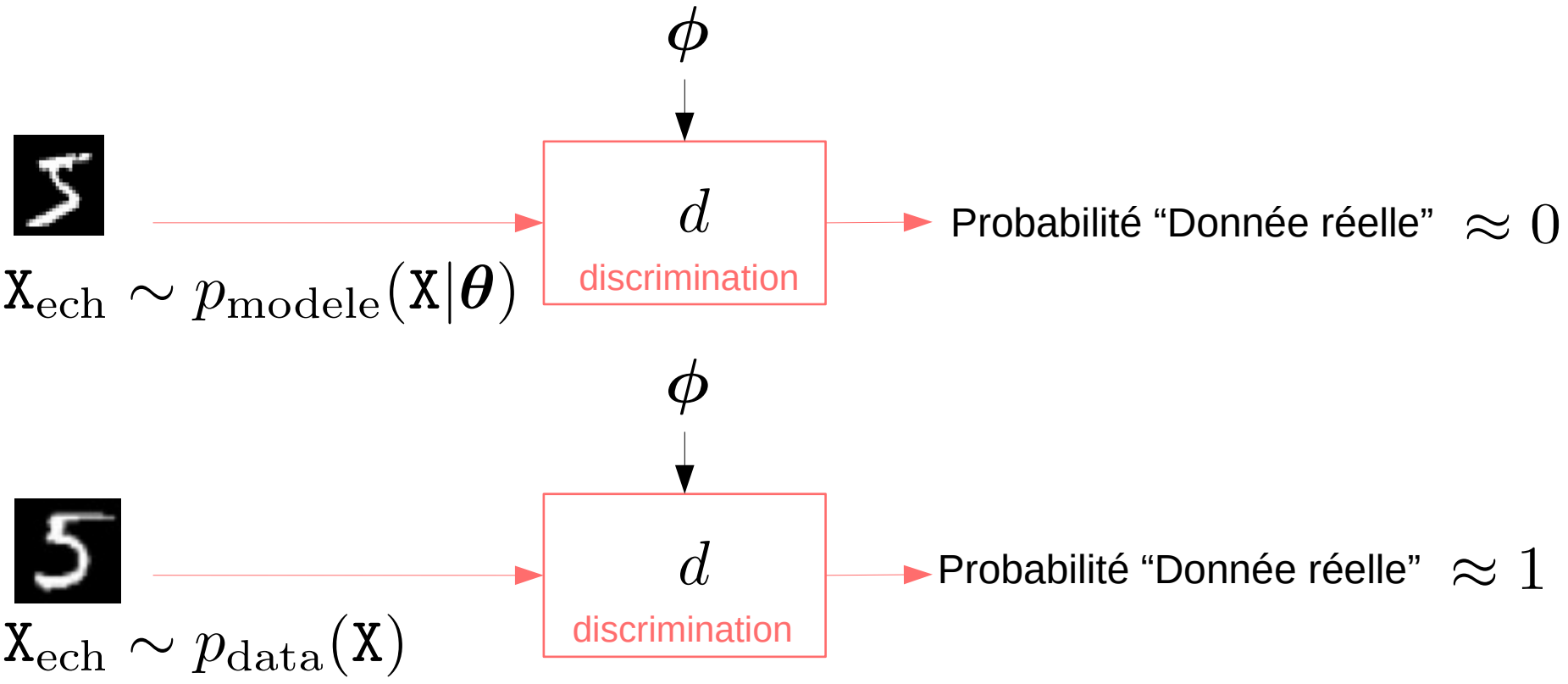
Réseau antagoniste : générateur



Définition implicite, le générateur n'est pas inversible

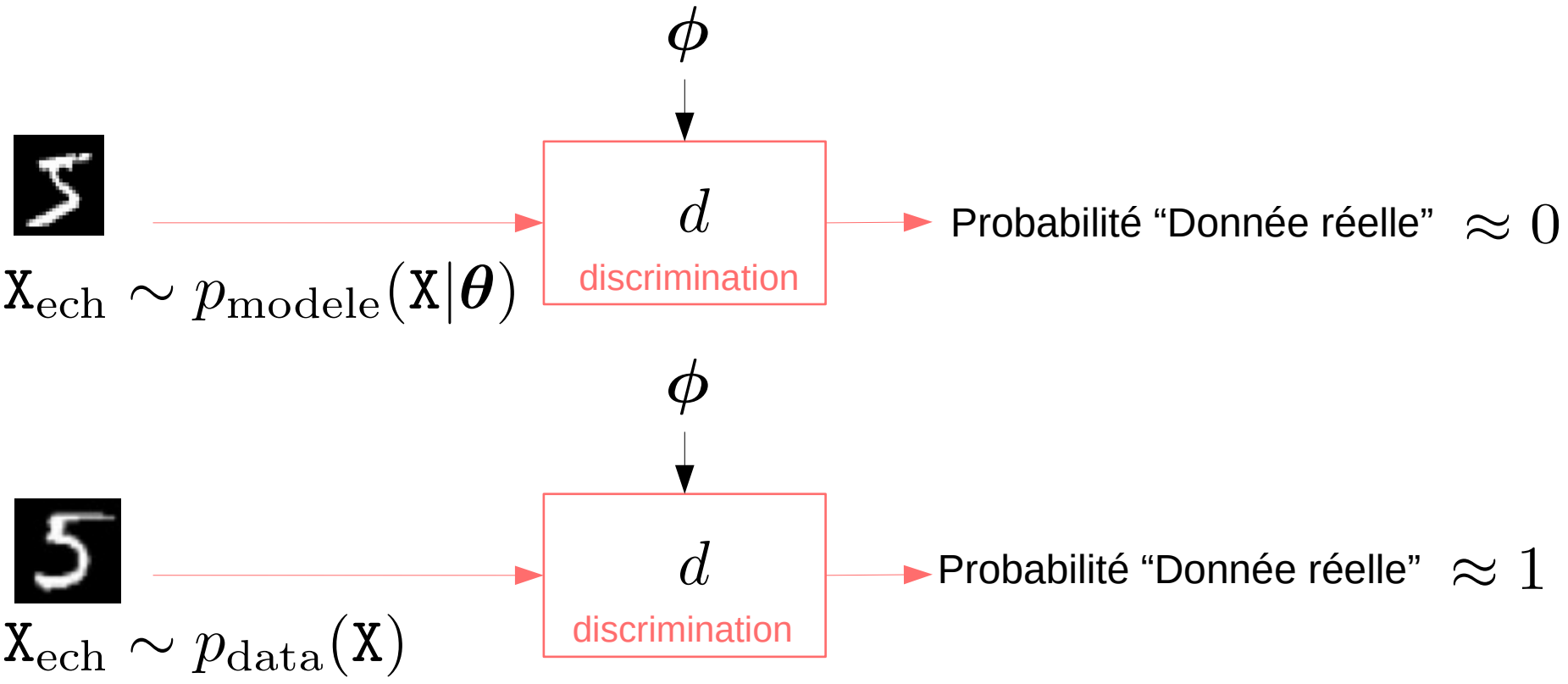
VI)

Réseau antagoniste : discriminateur



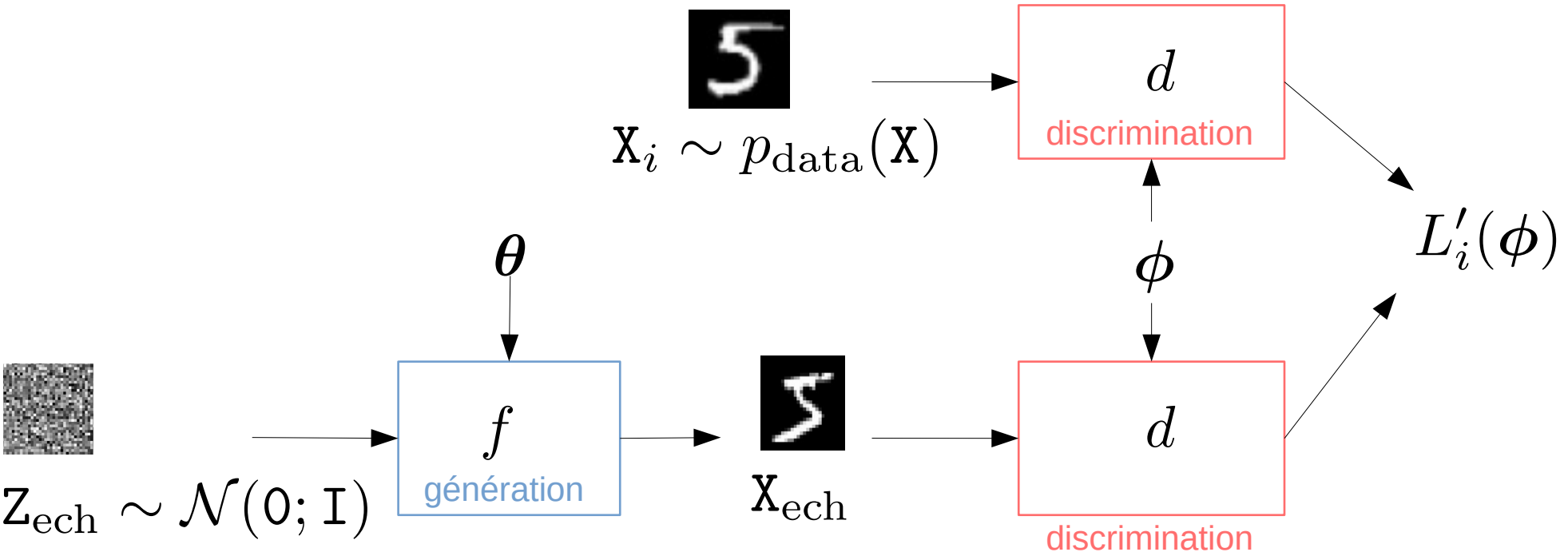
VI)

Réseau antagoniste : discriminateur



Réseau antagoniste : apprentissage

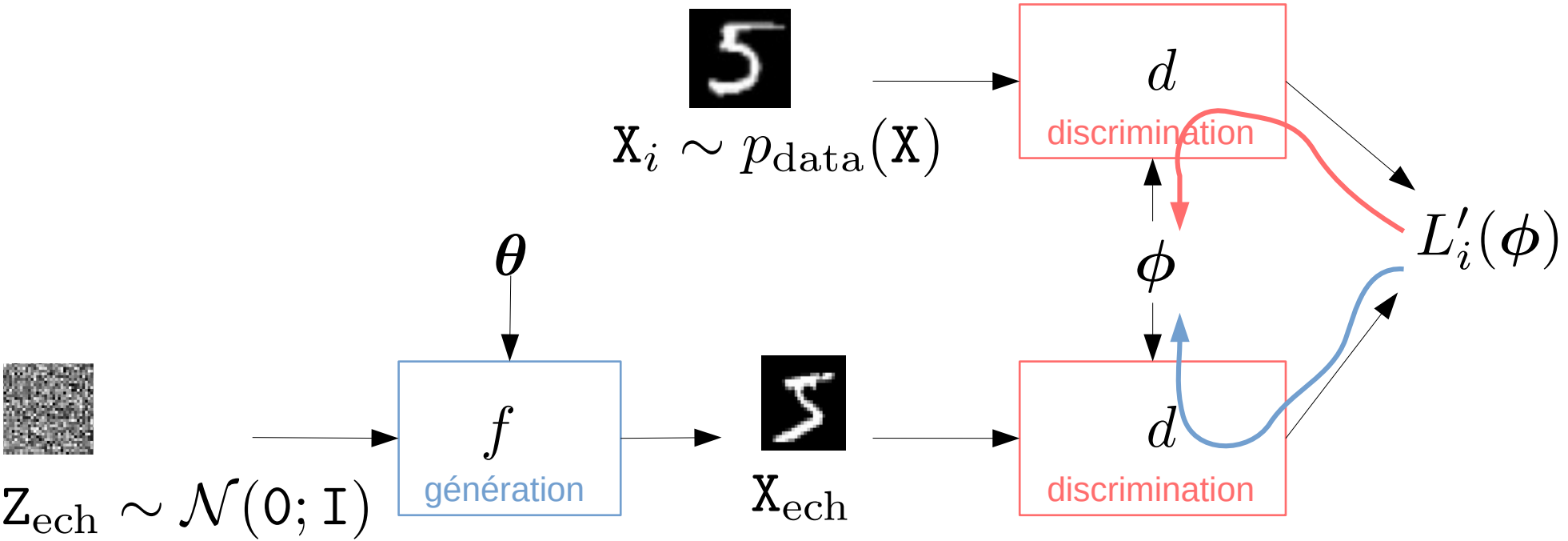
Pas de gradient sur les paramètres du discriminateur



Le discriminateur s'améliore pour mieux détecter les faux échantillons.

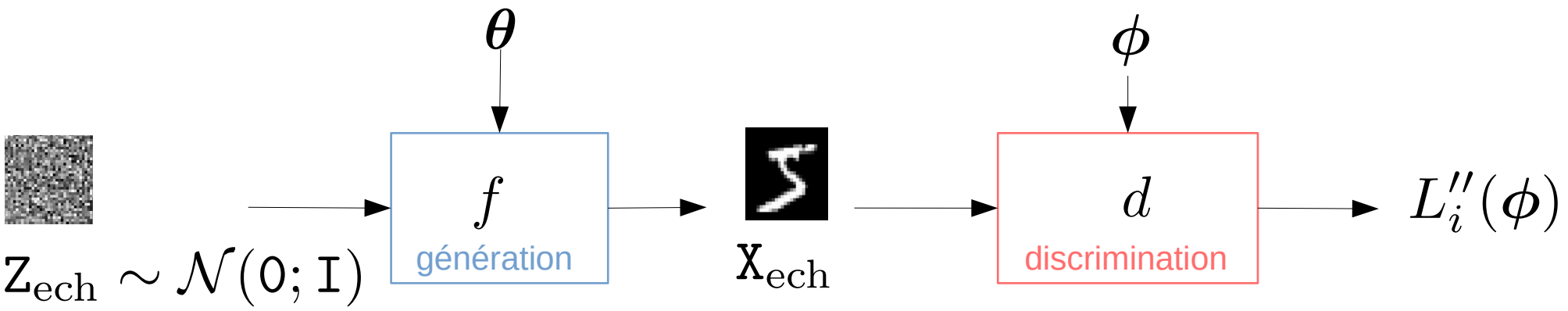
Réseau antagoniste : apprentissage

Pas de gradient sur les paramètres du discriminateur



Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du générateur

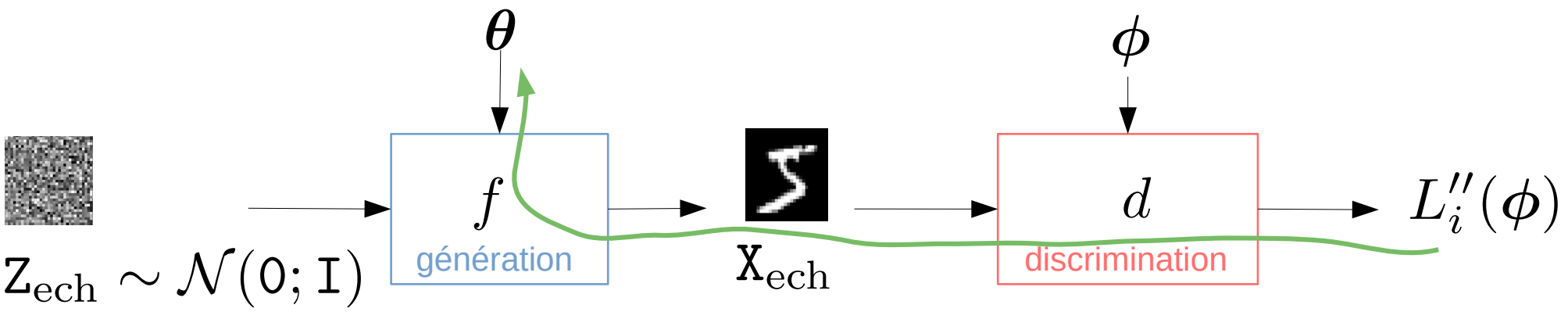


Le générateur s'améliore pour mieux tromper le discriminateur.

VI)

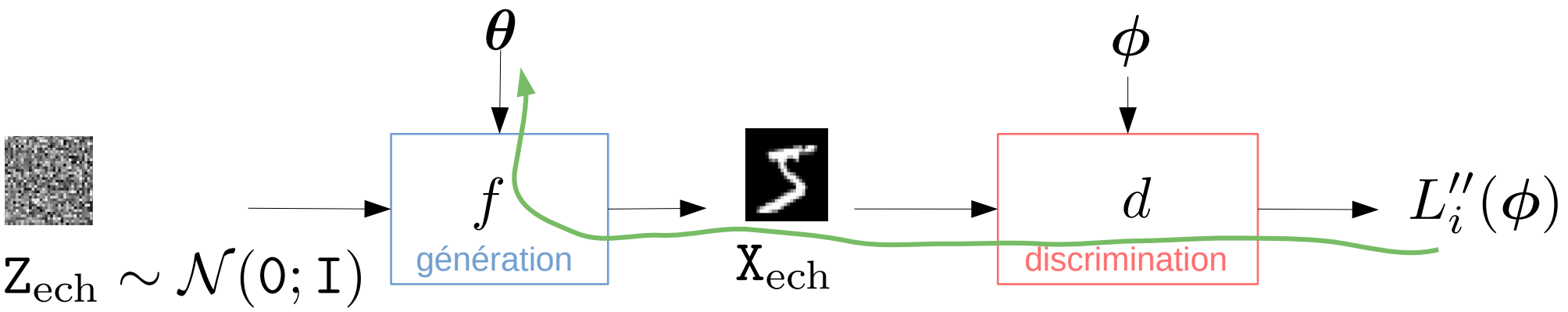
Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du générateur



Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du générateur



Problème d'apprentissage très difficile car il ne faut pas que l'un "écrase" l'autre.

Réseau antagoniste : apprentissage (formalisé)

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

Réseau antagoniste : apprentissage (formalisé)

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un GAN on choisit généralement une borne inférieur d'une divergence :

$$D(p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})||p_{\text{data}}(\mathtt{X})) \geq \max_{d(.)} \mathbb{E}_{p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})}(\ln(1-d(\mathtt{X})) + \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(\ln(d(\mathtt{X}))$$

discriminateur

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

Réseau antagoniste : apprentissage (formalisé)

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un GAN on choisit généralement une borne inférieur d'une divergence :

$$D(p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})||p_{\text{data}}(\mathtt{X})) \geq \max_{d(.)} \mathbb{E}_{p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})}(\ln(1-d(\mathtt{X})) + \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(\ln(d(\mathtt{X}))$$

discriminateur

Uniquement besoin de savoir échantillonner selon $p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})$

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

Réseau antagoniste : apprentissage (formalisé)

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un GAN on choisit généralement une borne inférieur d'une divergence :

$$D(p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})||p_{\text{data}}(\mathtt{X}))$$
$$\geq \max_{d(.)} \mathbb{E}_{p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})}(\ln(1-d(\mathtt{X})) + \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(\ln(d(\mathtt{X}))$$

Astuce de reparamétrisation ("Reparameterization trick")

$$= \max_{d(.)} \mathbb{E}_{\mathcal{N}(0,\mathtt{I})}(\ln(1-d(f(\mathtt{Z},\boldsymbol{\theta}))) + \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(\ln(d(\mathtt{X}))$$

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

Réseau antagoniste : apprentissage (formalisé)

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Pour un GAN on choisit généralement une borne inférieur d'une divergence :

$$D(p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})||p_{\text{data}}(\mathtt{X}))$$
$$\geq \max_{d(.)} \mathbb{E}_{p_{\text{m}}(\mathtt{X}|\boldsymbol{\theta})}(\ln(1-d(\mathtt{X})) + \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(\ln(d(\mathtt{X}))$$

Astuce de reparamétrisation ("Reparameterization trick")

$$= \max_{d(.)} \mathbb{E}_{\mathcal{N}(0,\mathtt{I})}(\ln(1-d(f(\mathtt{Z},\boldsymbol{\theta}))) + \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(\ln(d(\mathtt{X}))$$

En pratique, on va paramétrer le discriminateur par un réseau de neurones.

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

Réseau antagoniste : problème d'apprentissage

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{\mathcal{N}(0,\mathtt{I})}(\ln(1 - d(f(\mathtt{Z}, \boldsymbol{\theta}), \boldsymbol{\phi})) + \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(\ln(d(\mathtt{X}, \boldsymbol{\phi}))$$

Problème minmax → potentiellement beaucoup plus difficile à optimiser qu'un VAE ou qu'un NF

Réseau antagoniste : problème d'apprentissage

En pratique on alterne :

Un pas gradient pour optimiser $\boldsymbol{\phi}$

$$L'_i(\boldsymbol{\phi}) = -\ln(1 - d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \boldsymbol{\phi})) - \ln(d(\mathtt{X}_i, \boldsymbol{\phi}))$$

Objectif : Améliorer les paramètres du discriminateur afin qu'il prédise une valeur proche de 0 pour une donnée générée et proche de 1 pour une donnée réelle

Réseau antagoniste : problème d'apprentissage

En pratique on alterne :

Un pas gradient pour optimiser $\boldsymbol{\phi}$

$$L'_i(\boldsymbol{\phi}) = -\ln(1 - d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \boldsymbol{\phi})) - \ln(d(\mathtt{X}_i, \boldsymbol{\phi}))$$

Objectif : Améliorer les paramètres du discriminateur afin qu'il prédise une valeur proche de 0 pour une donnée générée et proche de 1 pour une donnée réelle

Suivi d'un pas de gradient pour optimiser $\boldsymbol{\theta}$

$$L''_i(\boldsymbol{\theta}) = \ln(1 - d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \boldsymbol{\phi}))$$

En pratique remplacé par :

$$-\ln(d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \boldsymbol{\phi}))$$

Objectif : Améliorer les paramètres du générateur afin que le discriminateur se trompe et prédise une valeur proche de 1

Réseau antagoniste : avantages et inconvénients

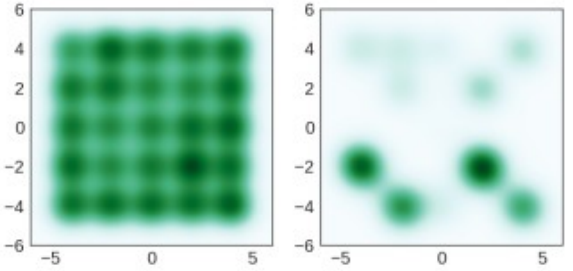
+ capable d’échantillonner efficacement
+ pas de contrainte particulière sur l’architecture
+ taille de Z non contrainte par celle de X

Réseau antagoniste : avantages et inconvénients

+ capable d'échantillonner efficacement
+ pas de contrainte particulière sur l'architecture
+ taille de Z non contrainte par celle de X

- pas d'expression de la (log-)probabilité d'une donnée
- problème minmax difficile à optimiser

** problèmes de convergence
** échantillonnage uniquement d'une partie de la distribution ("mode collapse")



(f) True Data (g) GAN