

Modèles génératifs profonds

Guillaume Bourmaud

PLAN

I. Introduction

II. Notion de divergence

III. Réseau inversible (NF)

IV. Auto-encodeur variationnel (VAE)

V. Modèle de diffusion (DDPM)

VI. Réseau antagoniste (GAN)

I) Introduction

Principe d'un modèle génératif

Base de données : $\{X_i\}_{i=1\dots N}$

$$\downarrow \\ \in \mathbb{R}^n$$

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

Principe d'un modèle génératif

Base de données : $\{X_i\}_{i=1\dots N}$

$$\downarrow \\ \in \mathbb{R}^n$$

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

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

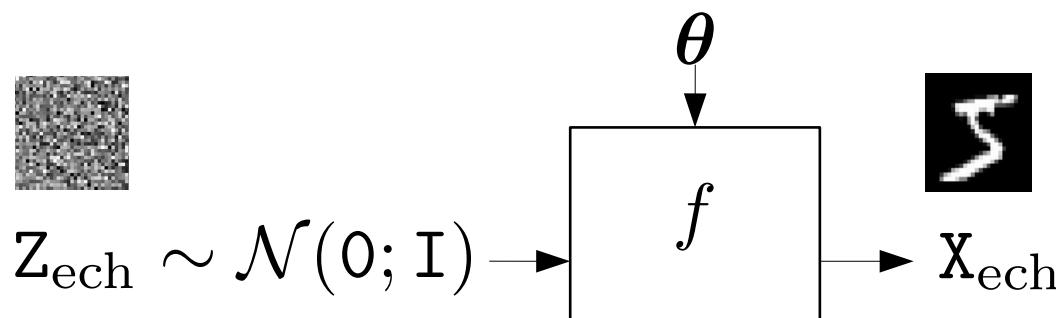
Principe d'un modèle génératif

Base de données : $\{X_i\}_{i=1 \dots N}$

$$\downarrow \\ \in \mathbb{R}^n$$

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

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



Principe d'un modèle génératif

Base de données : $\{X_i\}_{i=1 \dots N}$

$$\downarrow \\ \in \mathbb{R}^n$$

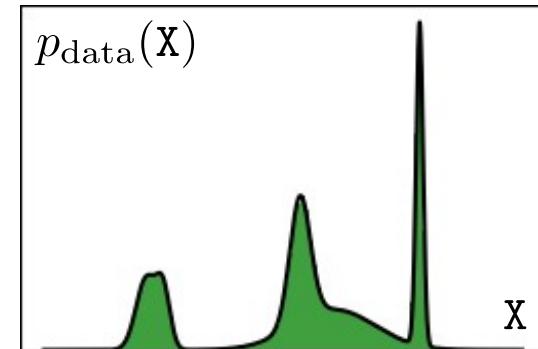
3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

Point de vue probabiliste

“est un échantillon de”

$$X_i \xrightarrow{\downarrow} p_{\text{data}}(X)$$

Densité de probabilité
inconnue



Principe d'un modèle génératif

Base de données : $\{X_i\}_{i=1 \dots N}$

$$\downarrow \\ \in \mathbb{R}^n$$

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

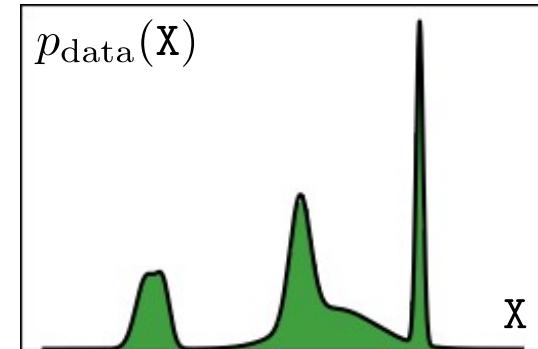
Point de vue probabiliste

“est un échantillon de”

$$X_i \xrightarrow{\downarrow} p_{\text{data}}(X)$$

Densité de probabilité
inconnue

On supposera les X_i i.i.d.



Exemple 1D

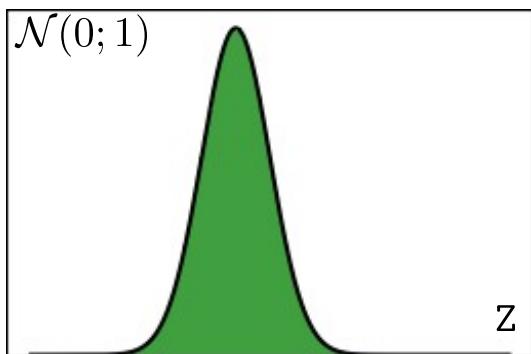
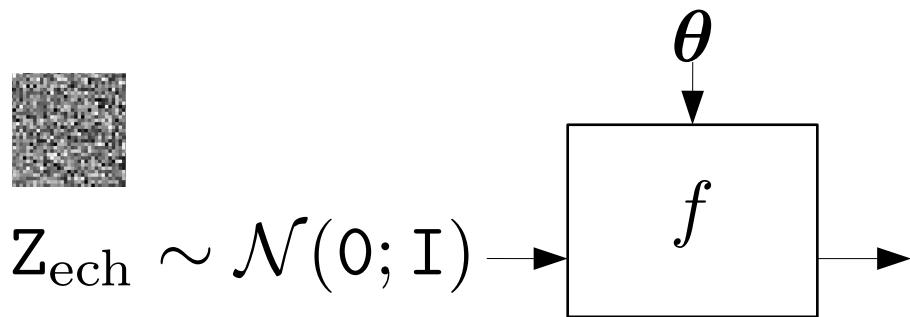
Principe d'un modèle génératif (suite)

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

I)

Principe d'un modèle génératif (suite)

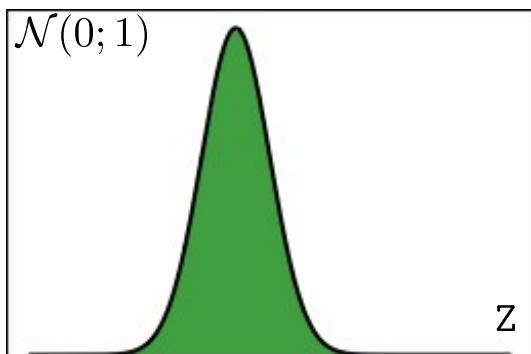
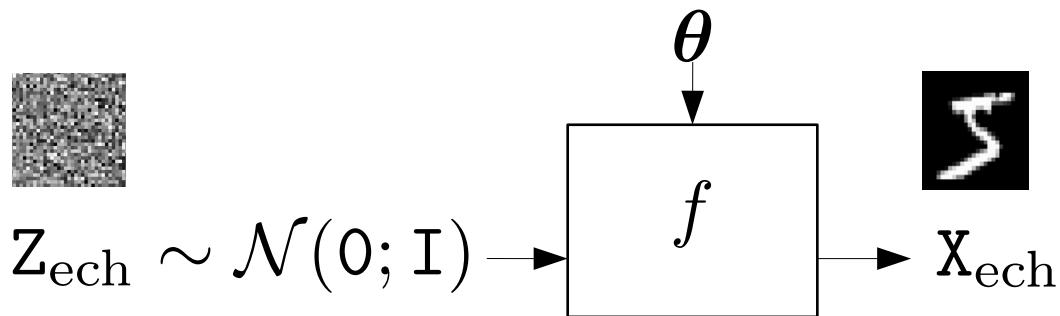
Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathbf{X})$



I)

Principe d'un modèle génératif (suite)

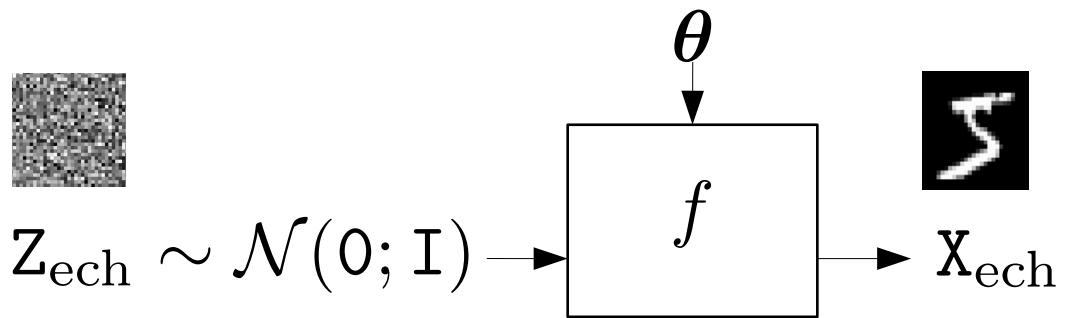
Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathbf{X})$



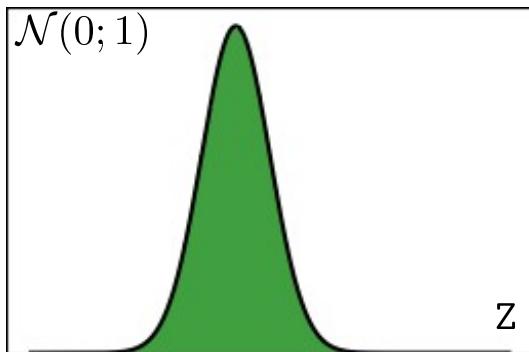
I)

Principe d'un modèle génératif (suite)

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



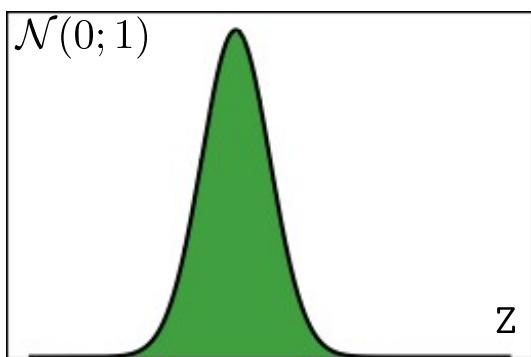
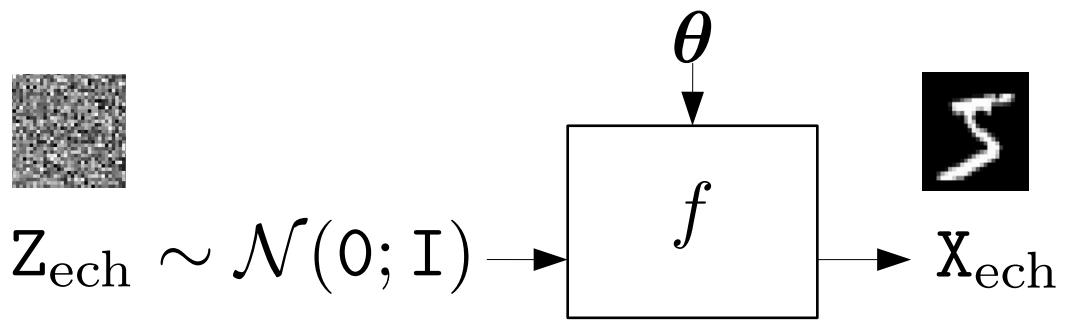
1) On tire $Z_{\text{ech}} \sim p(Z) = \mathcal{N}(0; \mathbf{I})$



I)

Principe d'un modèle génératif (suite)

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



- 1) On tire $Z_{\text{ech}} \sim p(Z) = \mathcal{N}(0; \mathbb{I})$
- 2) On transforme Z_{ech}

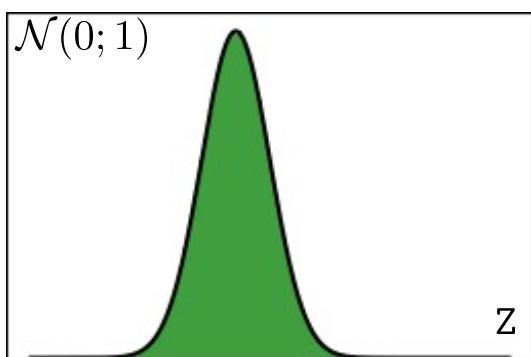
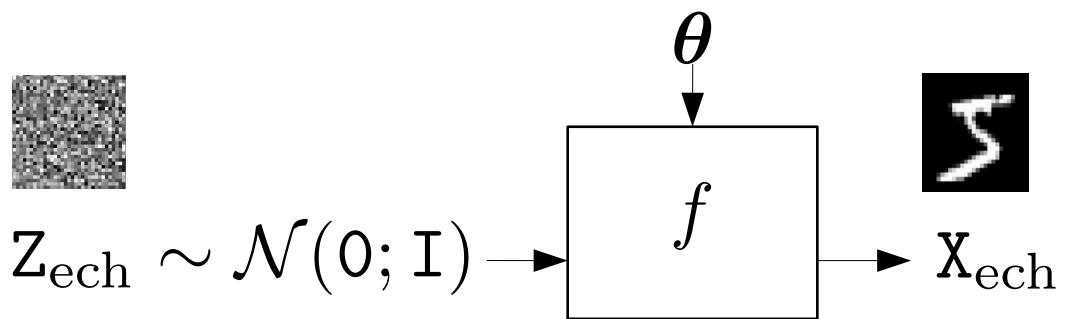
$$X_{\text{ech}} = f(Z_{\text{ech}}; \theta)$$

Ce qui définit **implicitement** $p_m(\mathbf{X}|\theta)$

I)

Principe d'un modèle génératif (suite)

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



- 1) On tire $Z_{\text{ech}} \sim p(Z) = \mathcal{N}(0; \mathbf{I})$

- 2) On transforme Z_{ech}

$$X_{\text{ech}} = f(Z_{\text{ech}}; \theta)$$

Ce qui définit **implicitement** $p_m(\mathbf{X}|\theta)$

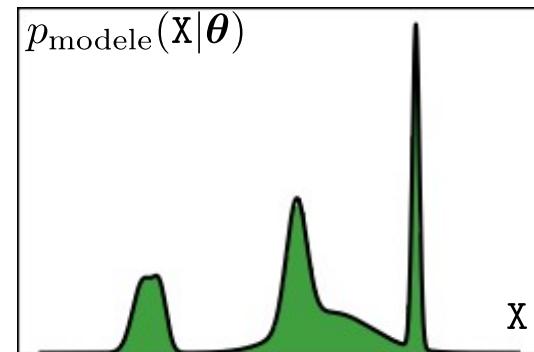
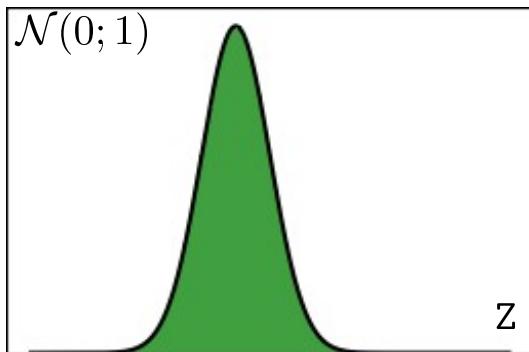
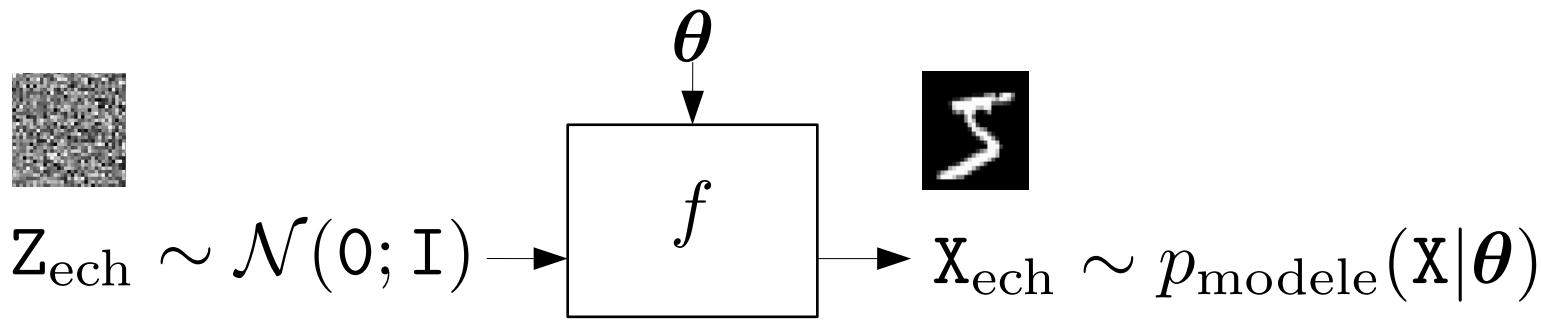
- 3) On a donc

$$X_{\text{ech}} \sim p_{\text{modele}}(\mathbf{X}|\theta)$$

I)

Principe d'un modèle génératif (suite)

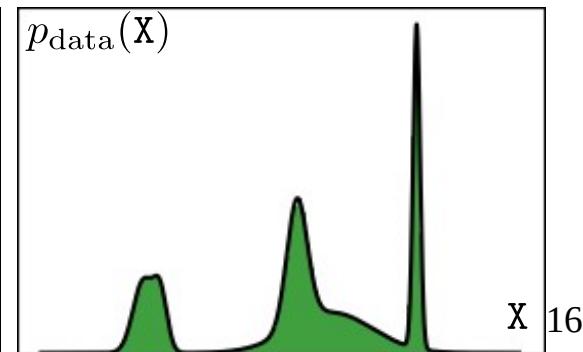
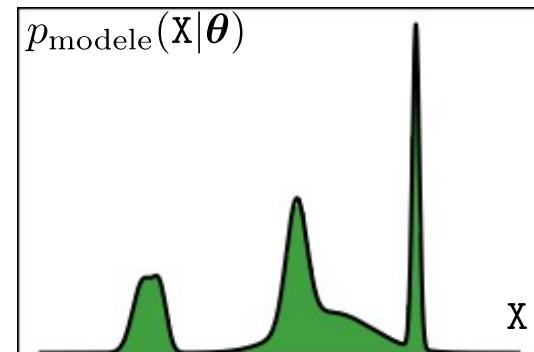
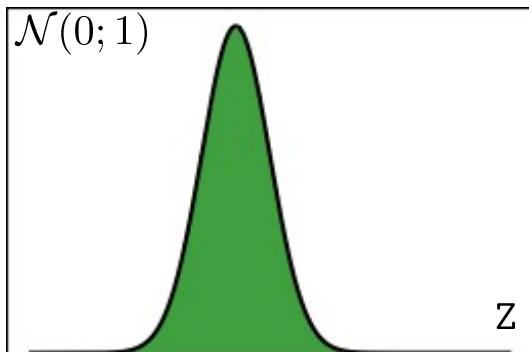
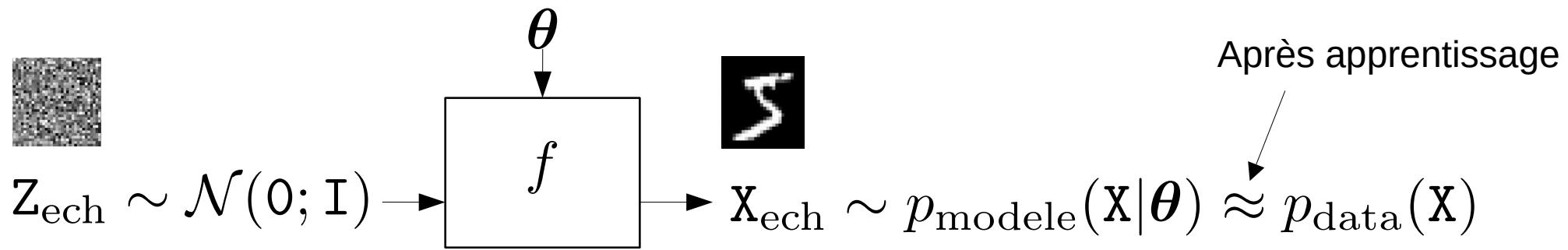
Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathbf{X})$



I)

Principe d'un modèle génératif (suite)

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



Difficulté de la tâche d'apprentissage

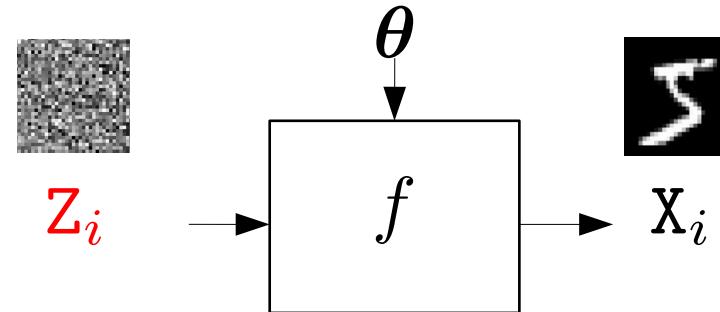
Base de données **non-étiquetées** : $\{X_i\}_{i=1\dots N}$

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

I)

Difficulté de la tâche d'apprentissage

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

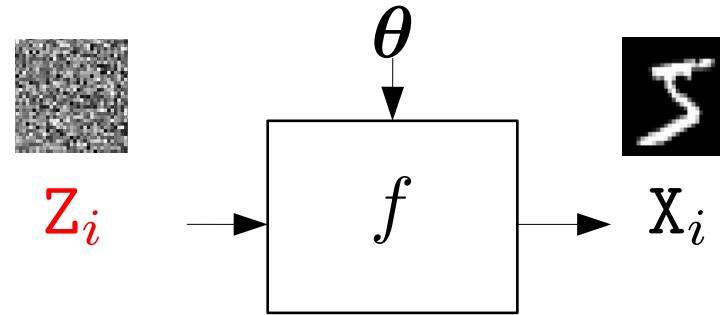


3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

Terminologie : Z_i = variable "latente"

Difficulté de la tâche d'apprentissage

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



3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	1	0	6	9	2	3

On ne dispose pas de couples $\{Z_i, X_i\}_{i=1\dots N}$

→ **problème d'apprentissage non-supervisé**

Terminologie : Z_i = variable “latente”

I)

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

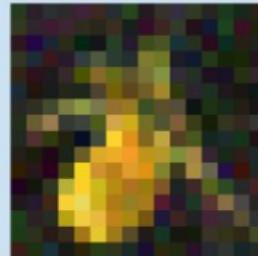
(a) Inpainting



(c) Gaussian deblur



(b) Super-resolution



(d) Motion deblur



Principe d'un modèle génératif conditionnel

Base de données **étiquetées** : $\{Y_i, X_i\}_{i=1\dots N}$



Exemple : Colorisation d'image

Principe d'un modèle génératif conditionnel

Base de données **étiquetées** : $\{Y_i, X_i\}_{i=1\dots N}$



Exemple : Colorisation d'image

Point de vue probabiliste : $X_i \sim p_{\text{data}}(X|Y_i)$ **Inconnue**

Échantillon d'une distribution a posteriori 23

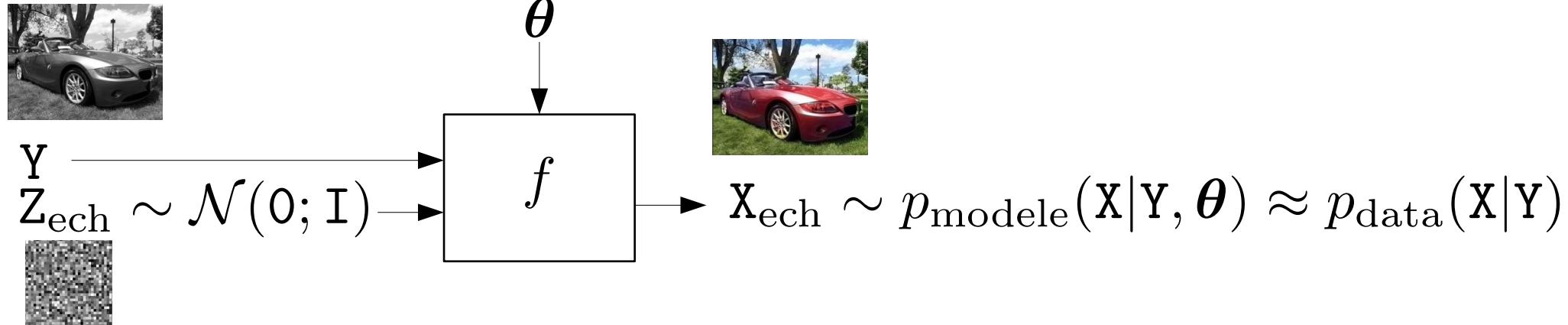
Principe d'un modèle génératif conditionnel (suite)

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

I)

Principe d'un modèle génératif conditionnel (suite)

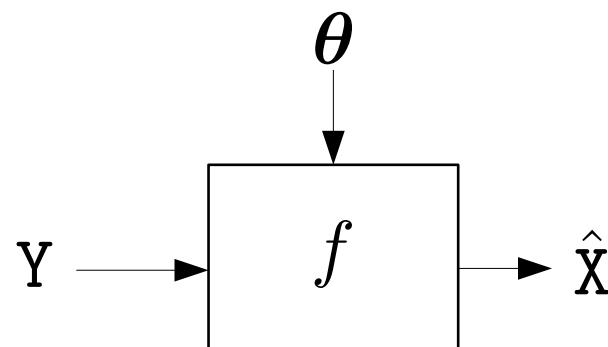
Objectif : Optimiser les paramètres d'un réseau de neurones afin de générer, à partir d'un Y , de **nouveaux échantillons** de $p_{\text{data}}(X|Y)$



I)

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour la régression

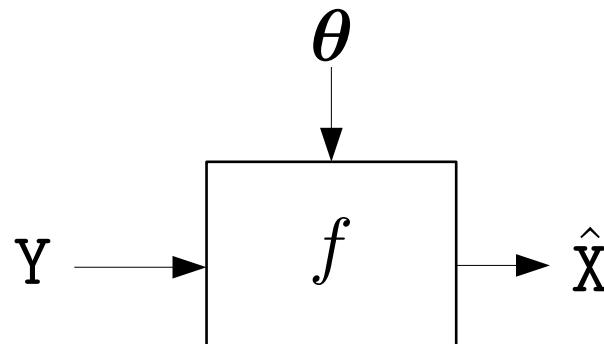


Régresseur

I)

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour la régression



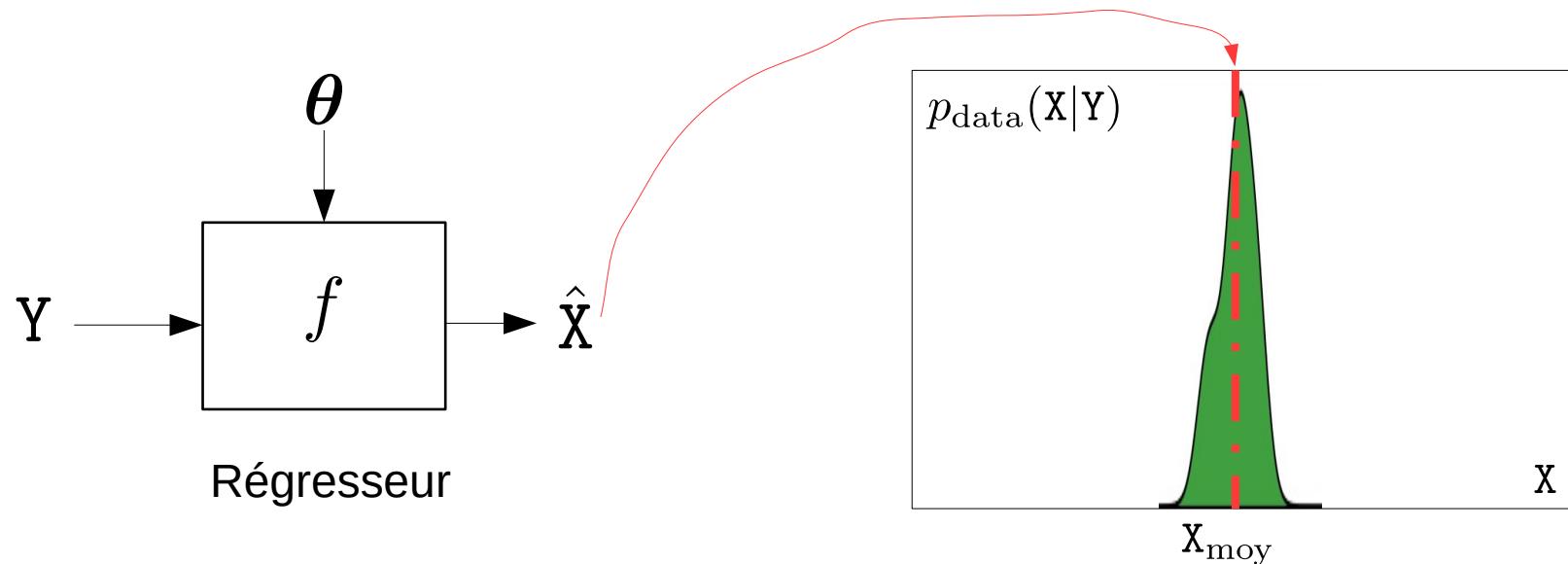
Régresseur

$$L(\boldsymbol{\theta}) = \sum_i \|\mathbf{x}_i - f(\mathbf{Y}; \boldsymbol{\theta})\|^2$$

Pour un même \mathbf{Y} le réseau apprend à prédire la moyenne des \mathbf{x}_i

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour la régression

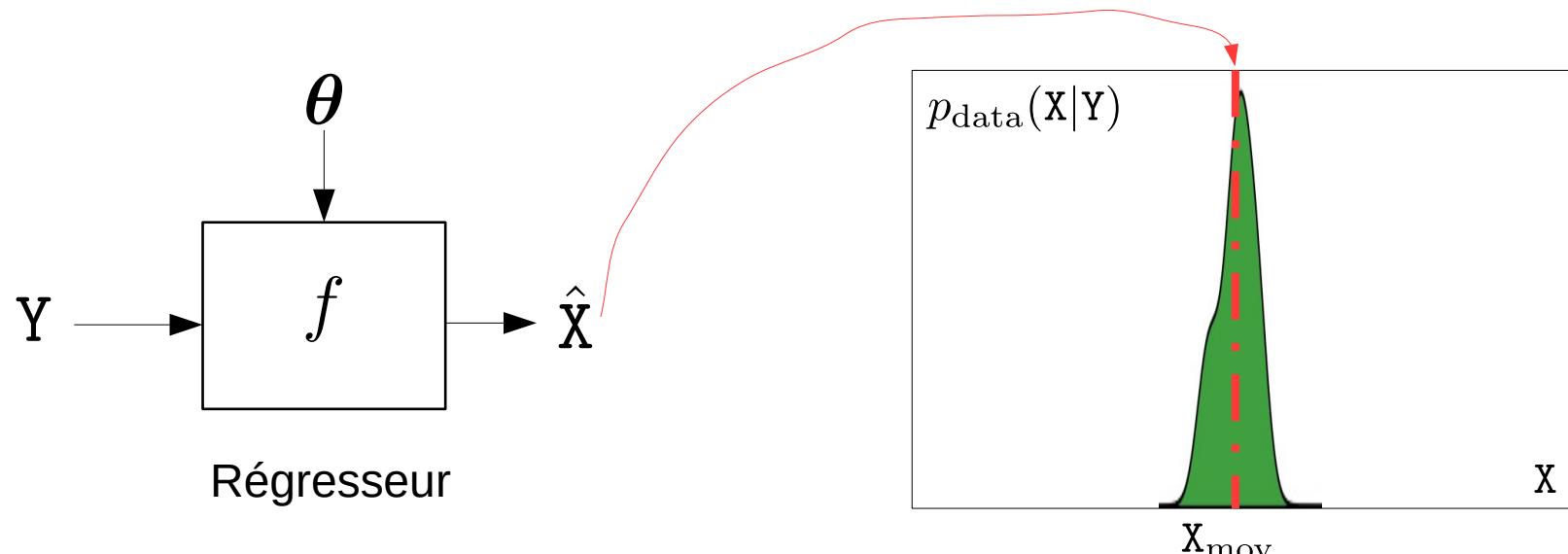


$$L(\theta) = \sum_i \|\mathbf{x}_i - f(\mathbf{y}; \theta)\|^2$$

Pour un même \mathbf{Y} le réseau apprend à prédire la moyenne des \mathbf{X}_i

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour la régression



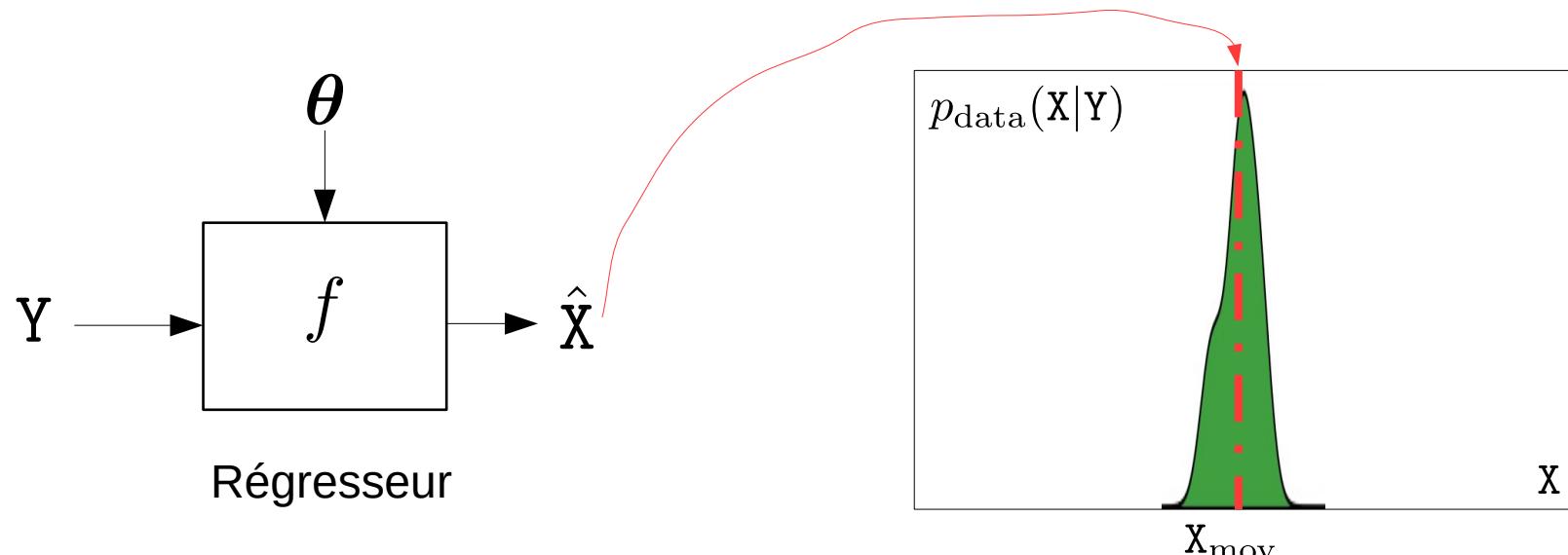
Ici, une prédiction telle que $\hat{X} \approx X_{\text{moy}}$ est satisfaisante car $p_{\text{data}}(X|Y)$ est

unimodale et piquée → tous les échantillons de $p_{\text{data}}(X|Y)$ seraient quasi-identiques

I)

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour la régression

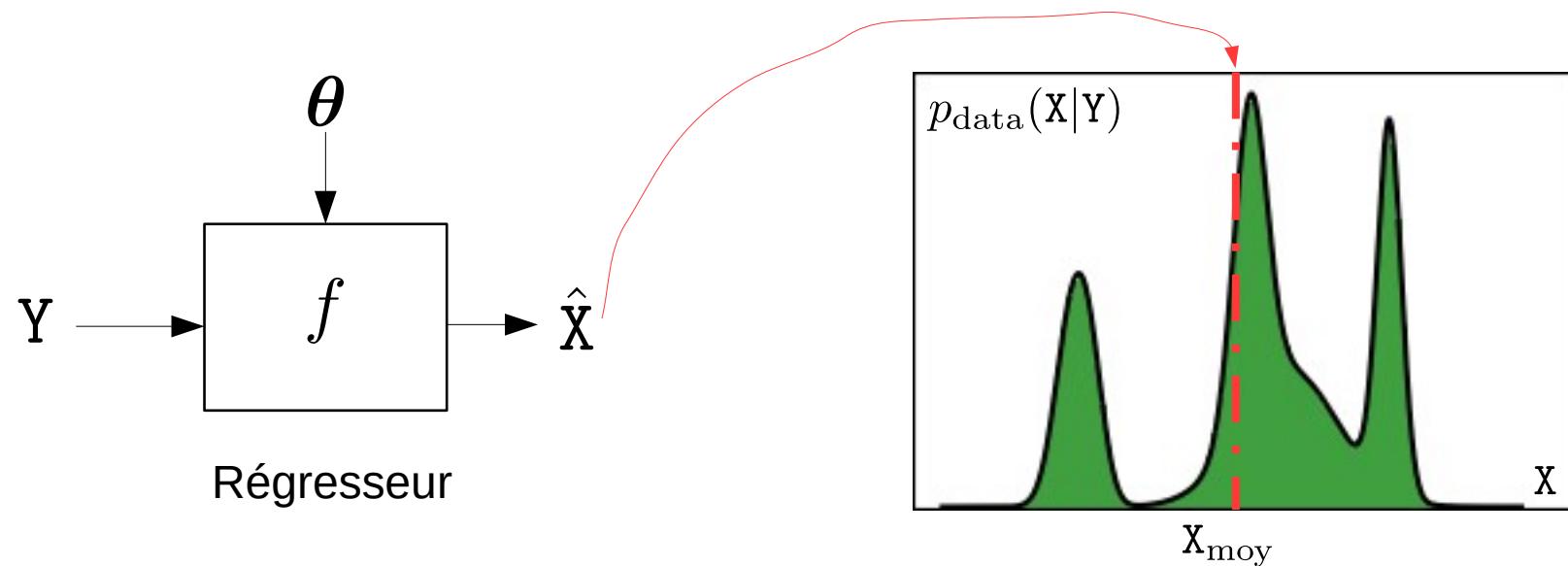


Ici, une prédiction telle que $\hat{X} \approx X_{\text{moy}}$ est satisfaisante car $p_{\text{data}}(X|Y)$ est

unimodale et piquée → tous les échantillons de $p_{\text{data}}(X|Y)$ seraient quasi-identiques
 Ici savoir échantillonner $p_{\text{data}}(X|Y)$ n'est pas très utile...

Différence vis-à-vis de l'apprentissage supervisé

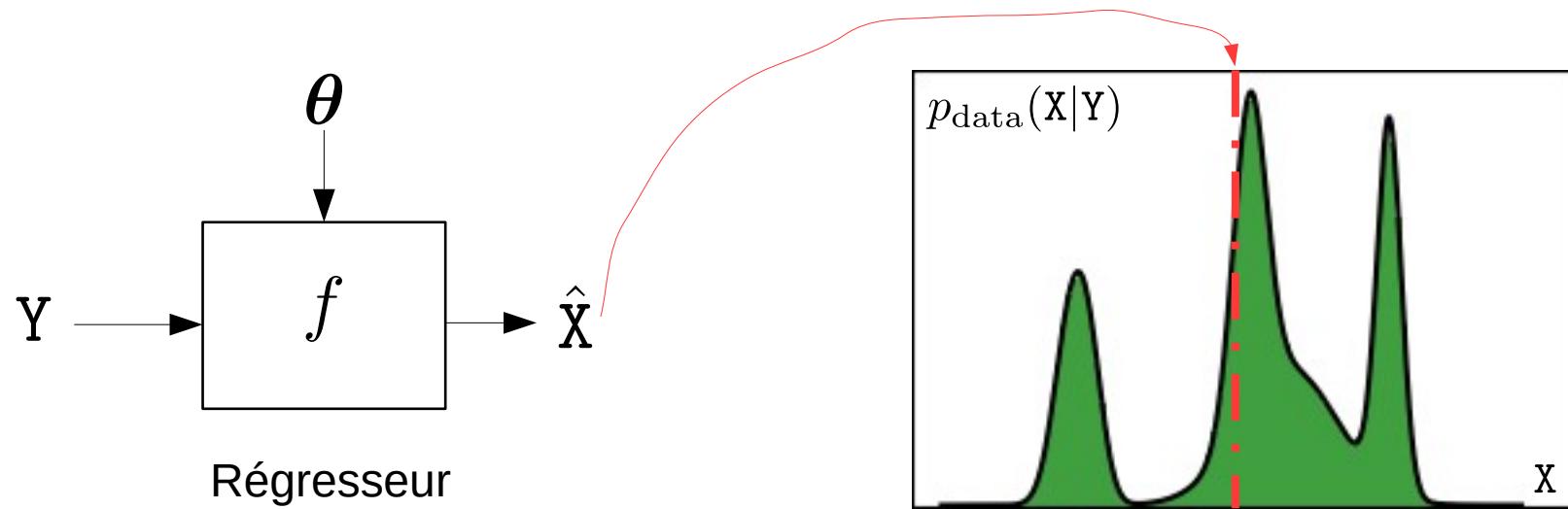
Rappel apprentissage supervisé pour la régression



I)

Différence vis-à-vis de l'apprentissage supervisé

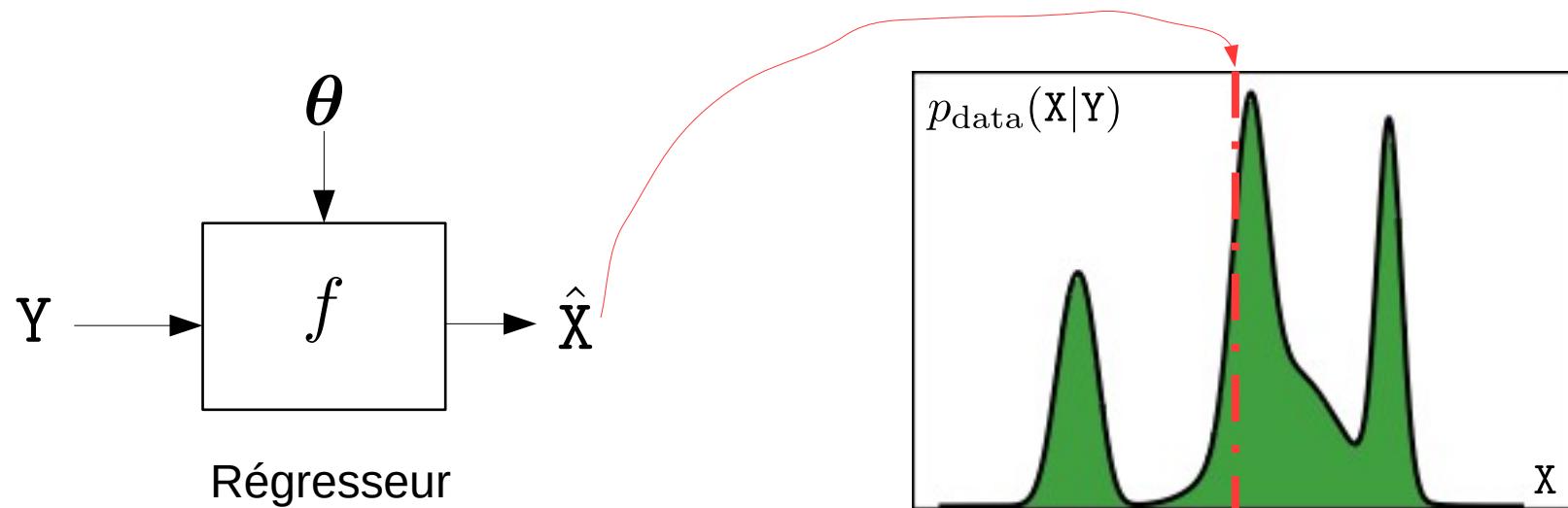
Rappel apprentissage supervisé pour la régression



Ici, une prédiction telle que $\hat{X} \approx X_{\text{moy}}$ n'est satisfaisante car $p_{\text{data}}(X|Y)$ est **multimodale** → grande variabilité des échantillons de $p_{\text{data}}(X|Y)$

Différence vis-à-vis de l'apprentissage supervisé

Rappel apprentissage supervisé pour la régression



Ici, une prédiction telle que $\hat{X} \approx X_{\text{moy}}$ n'est satisfaisante car $p_{\text{data}}(X|Y)$ est

multimodale → grande variabilité des échantillons de $p_{\text{data}}(X|Y)$
 Ici savoir échantillonner $p_{\text{data}}(X|Y)$ est très utile.

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** $p_{\text{data}}(\mathbf{X}|\mathbf{Y})$

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

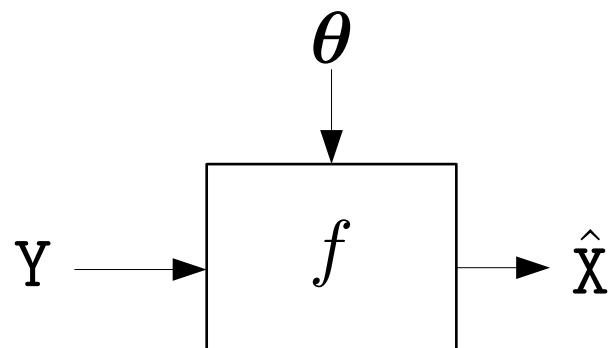
Question : Que se passe-t-il si on entraîne un régresseur pour prédire un “5” ?

“5”

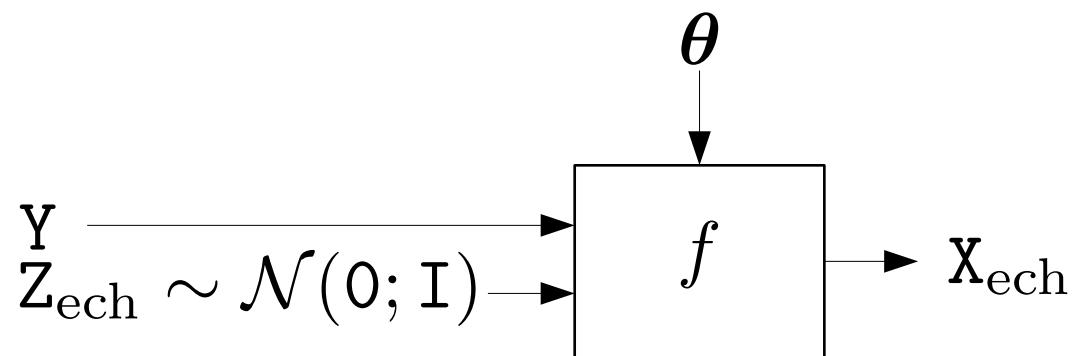


Nécessité d'apprendre à échantillonner

Différence vis-à-vis de l'apprentissage supervisé



Supervisé → Régresseur

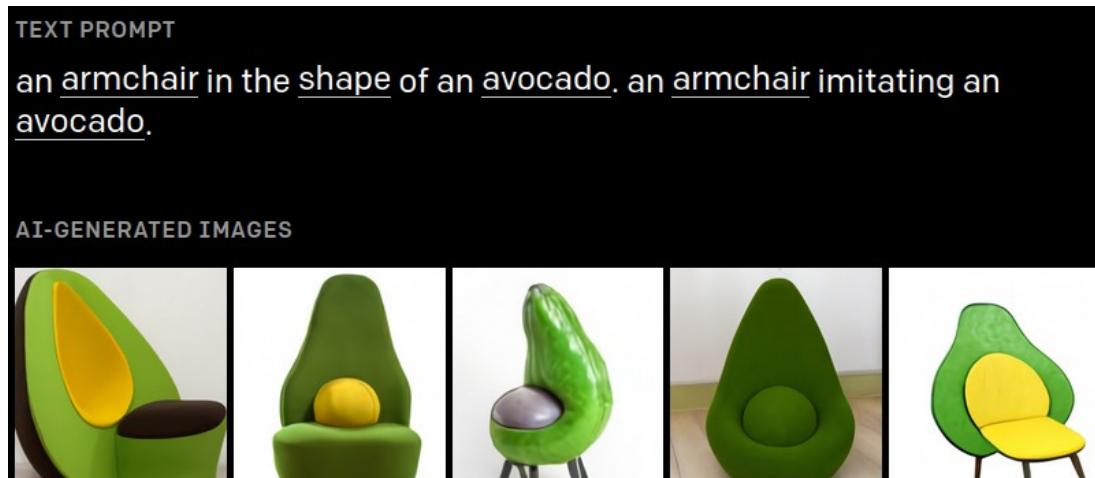


Modèle génératif → Échantillonneur

I)

Exemple d'utilisation

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



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

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

Notion de divergence

Nécessité de pouvoir comparer deux densités de probabilité, par exemple pour savoir si (pour une valeur de θ) $p_{\text{modele}}(X|\theta)$ "ressemble" à $p_{\text{data}}(X)$

Notion de divergence

Nécessité de pouvoir comparer deux densités de probabilité, par exemple pour savoir si (pour une valeur de θ) $p_{\text{modele}}(X|\theta)$ "ressemble" à $p_{\text{data}}(X)$

Densités de probabilité

$$D(p||q) \geq 0$$

scalaire

Divergence

$$D(p||q) = 0 \text{ ssi } p = q$$

Notion de divergence (suite)

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

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

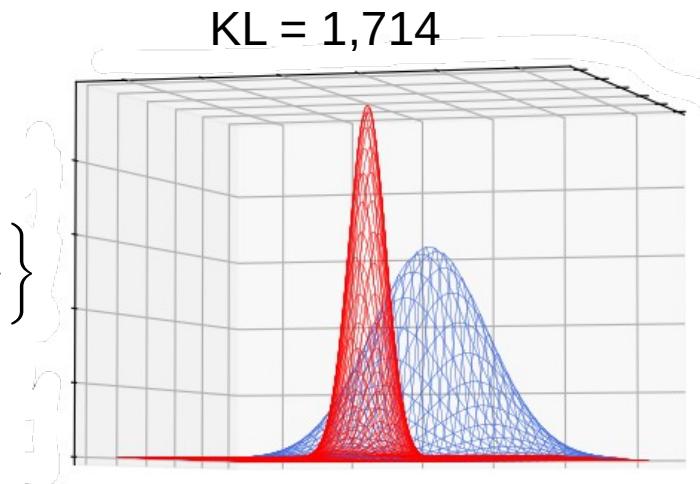
Notion de divergence (suite)

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

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

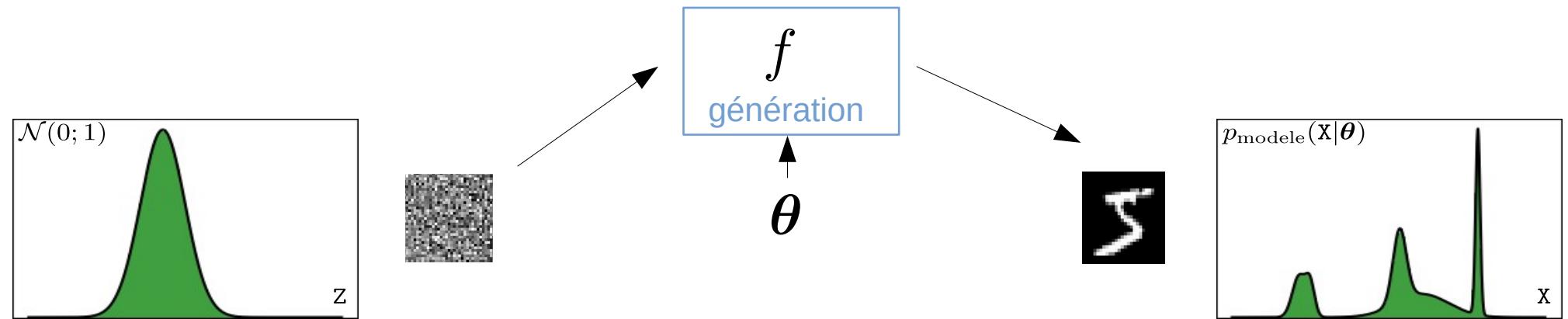
Exemple : Expression analytique pour deux gaussiennes
 $\mathcal{N}_0(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ et $\mathcal{N}_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$

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

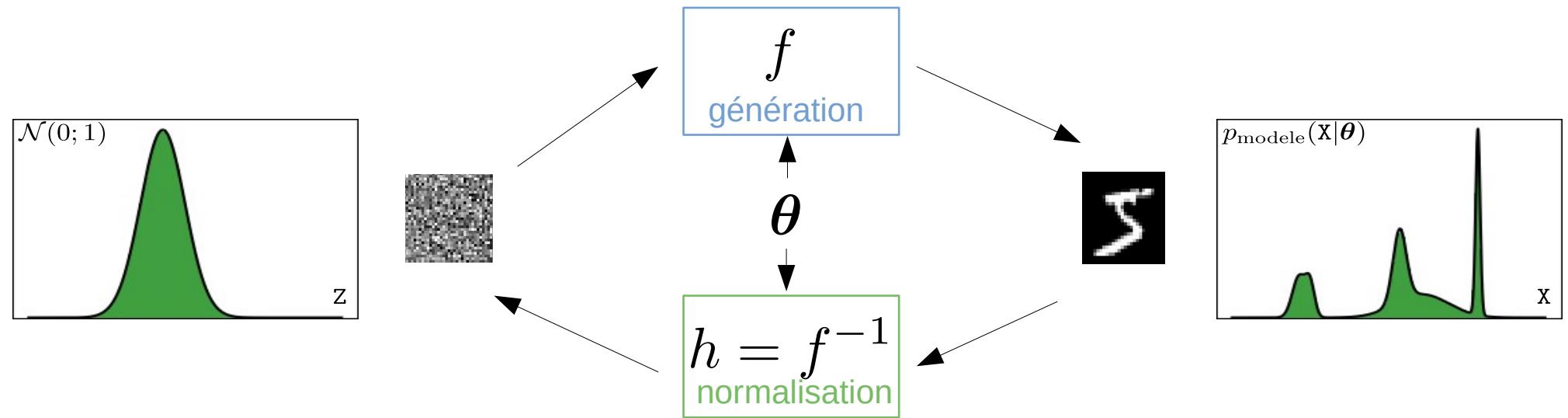


III) Réseau invisible (NF)

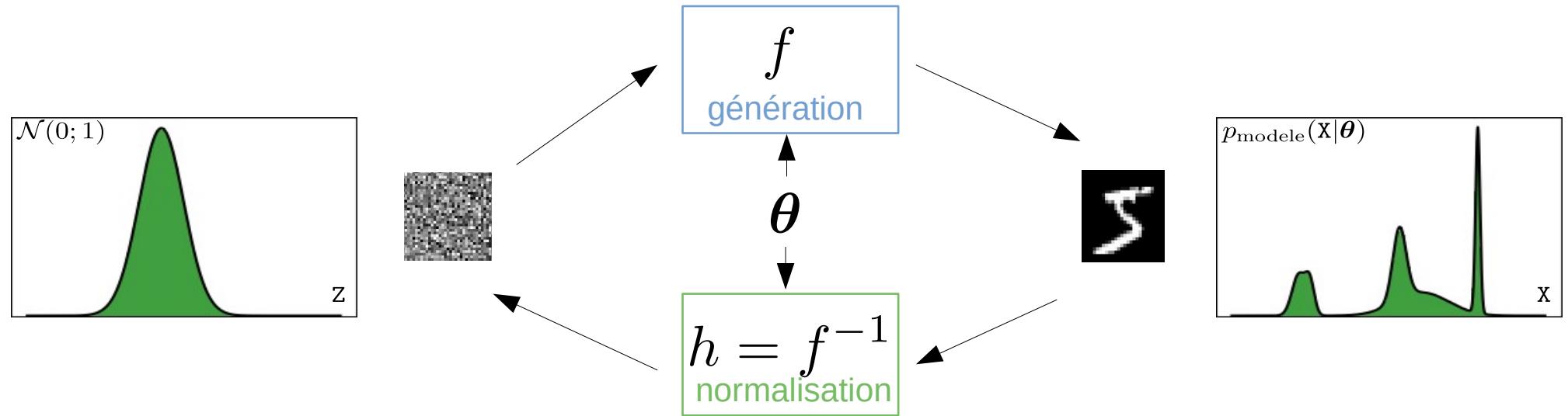
Réseau inversible : idée générale



Réseau inversible : idée générale



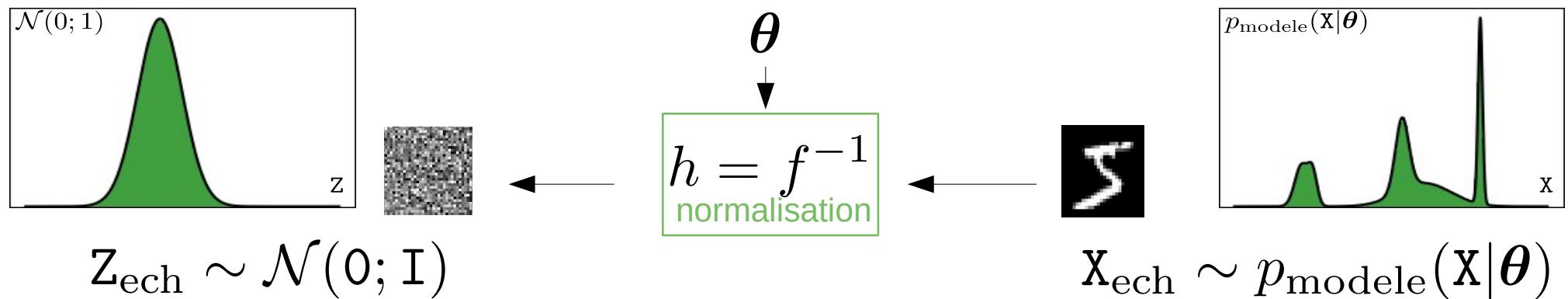
Réseau inversible : idée générale



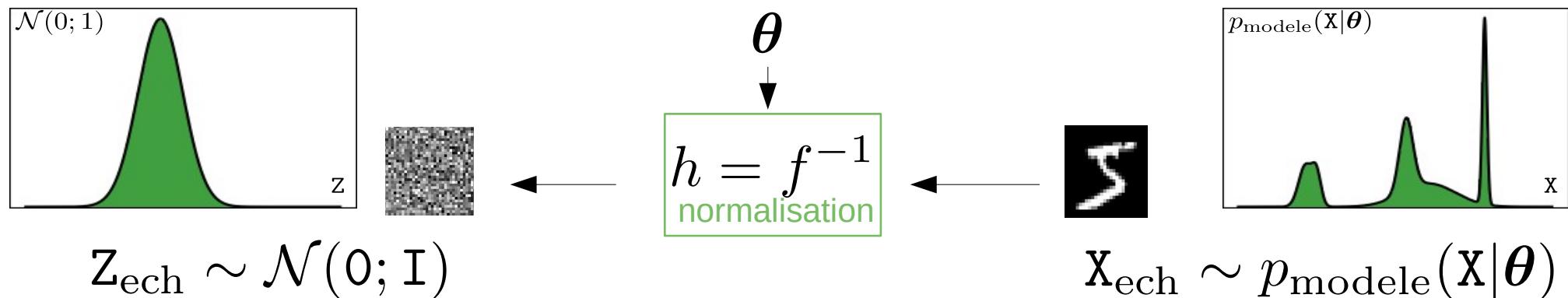
Ne pas disposer de couples $\{Z_i, X_i\}_{i=1\dots N}$ n'est plus un problème.

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

Réseau inversible : formalisme

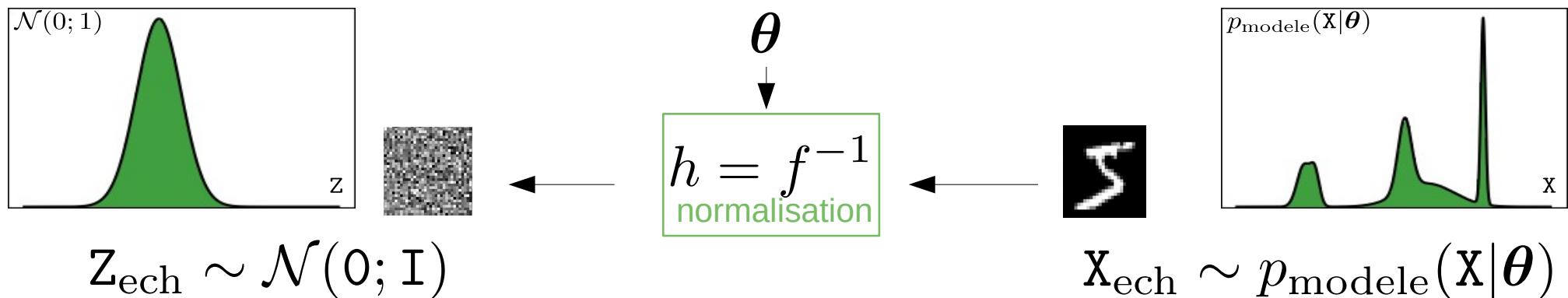


Réseau inversible : formalisme



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

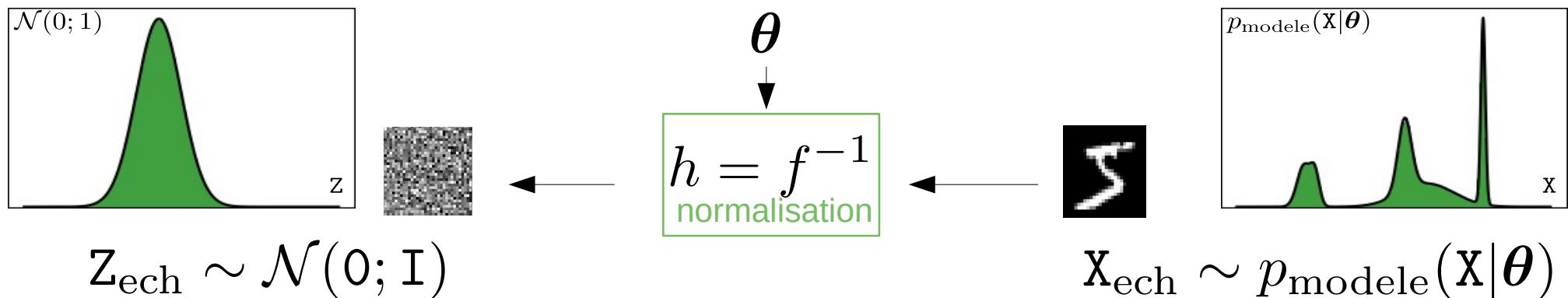
Réseau inversible : formalisme



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

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

Réseau invisible : formalisme (suite)



$$p_{\text{modele}}(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{N}(h(\mathbf{x}; \boldsymbol{\theta}); 0, \mathbf{I}) |\det J_h(\mathbf{x})|$$

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

Réseau invisible : apprentissage

Objectif : optimiser θ pour que $p_{\text{modele}}(\mathbf{x}|\theta) \approx p_{\text{data}}(\mathbf{x})$

Réseau invisible : apprentissage

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

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

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

Réseau invisible : apprentissage

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

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

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

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

Réseau invisible : apprentissage

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

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

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

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

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

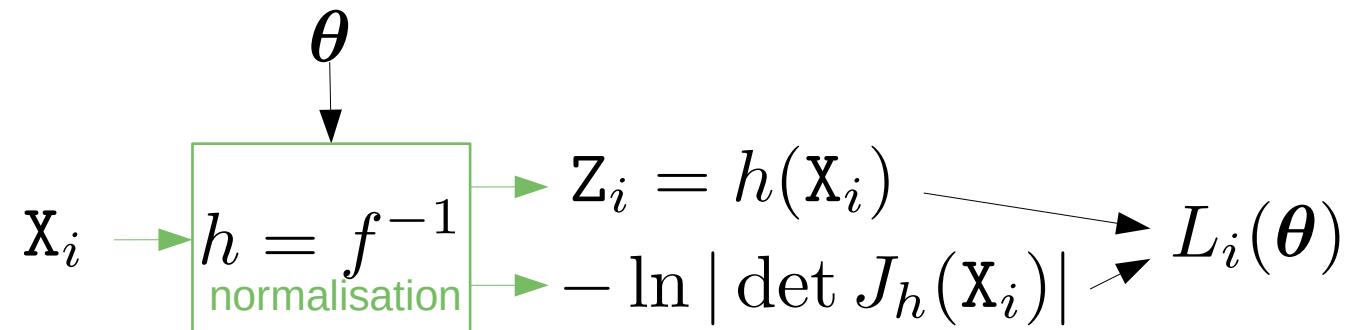
Réseau inversible : apprentissage (suite)

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

Essaye de transformer \mathbf{x}_i
proche du tenseur “zéro”

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

Descente de gradient



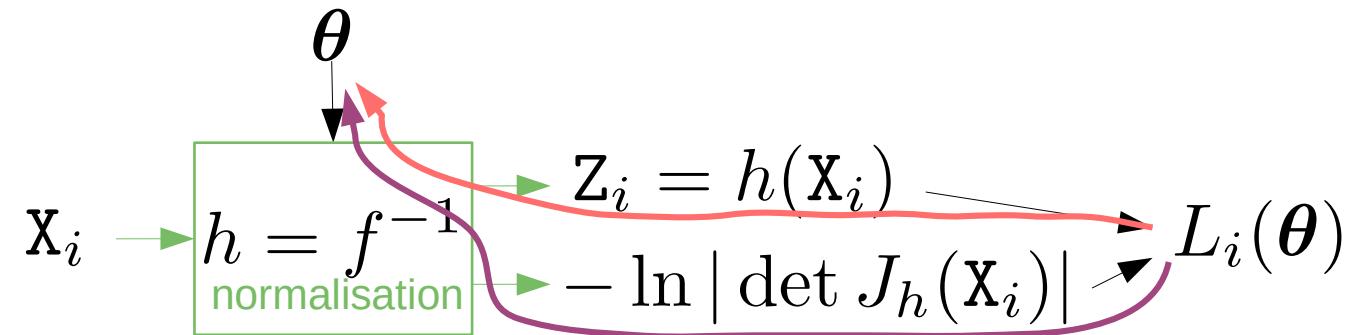
Réseau inversible : apprentissage (suite)

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

Essaye de transformer \mathbf{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 invisible : architecture

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

1. avoir la même taille en entrée et en sortie
2. être invisible, 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”

$$\begin{aligned} X_1 &= Z_1 \\ X_2 &= \exp(s_\theta(Z_1)) \odot Z_2 + m_\theta(Z_1) \end{aligned}$$

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

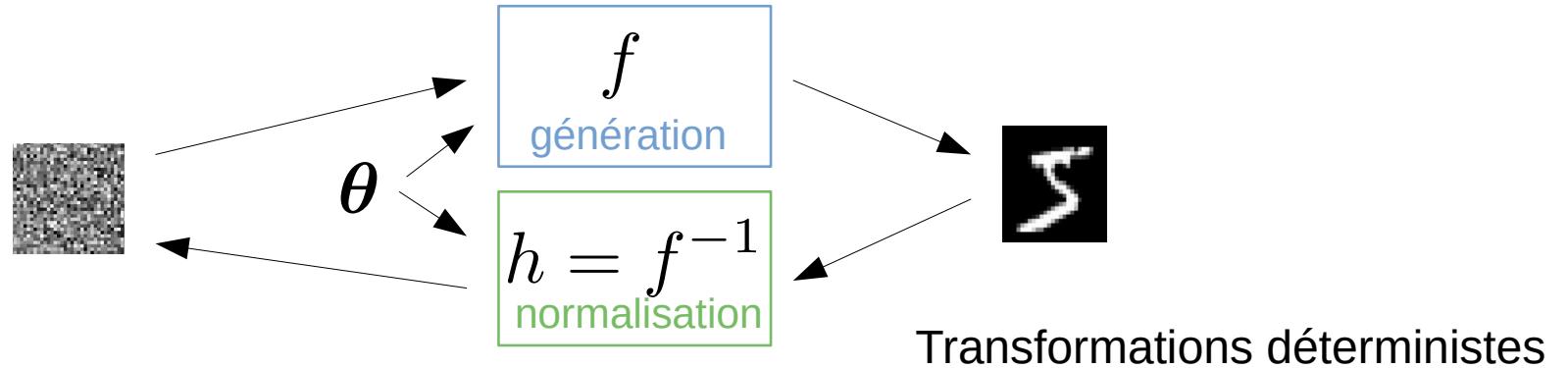
- + capable d'échantillonner efficacement
- + capable de calculer la (log-)probabilité d'une donnée
- + apprentissage possible par maximum de vraisemblance

- 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)

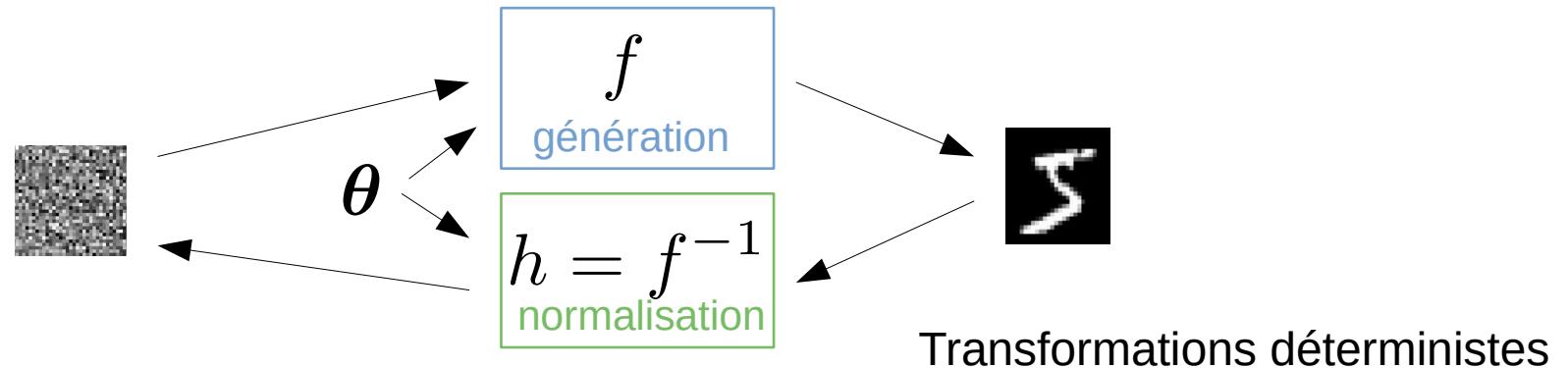
Auto-encodeur variationnel : idée générale

Réseau
inversible

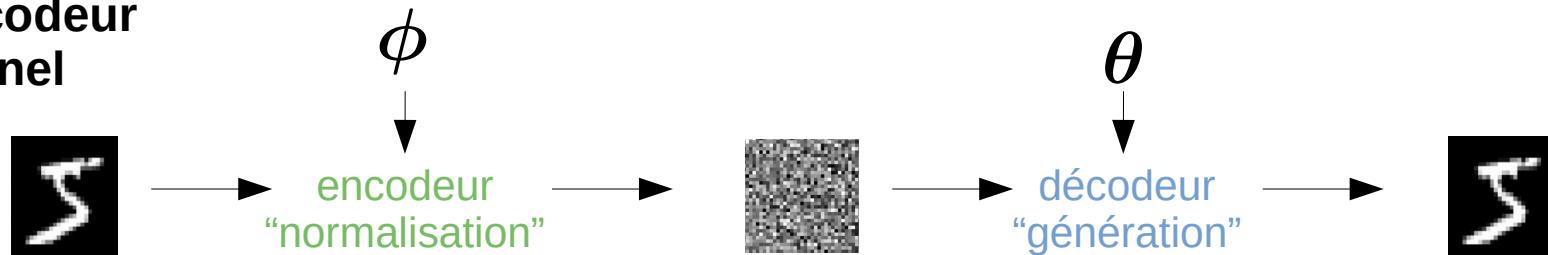


Auto-encodeur variationnel : idée générale

**Réseau
inversible**



**Auto-encodeur
variationnel**



Terminologie : “Variational Auto-Encoder” (VAE)

76

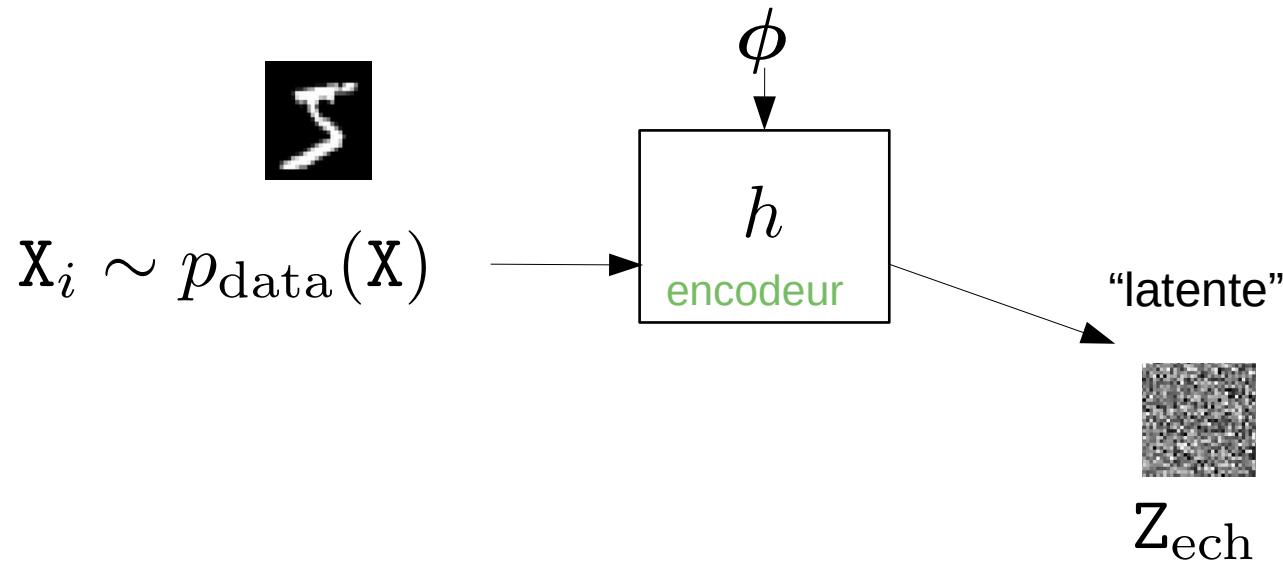
Transformations stochastiques

Auto-encodeur variationnel : principe

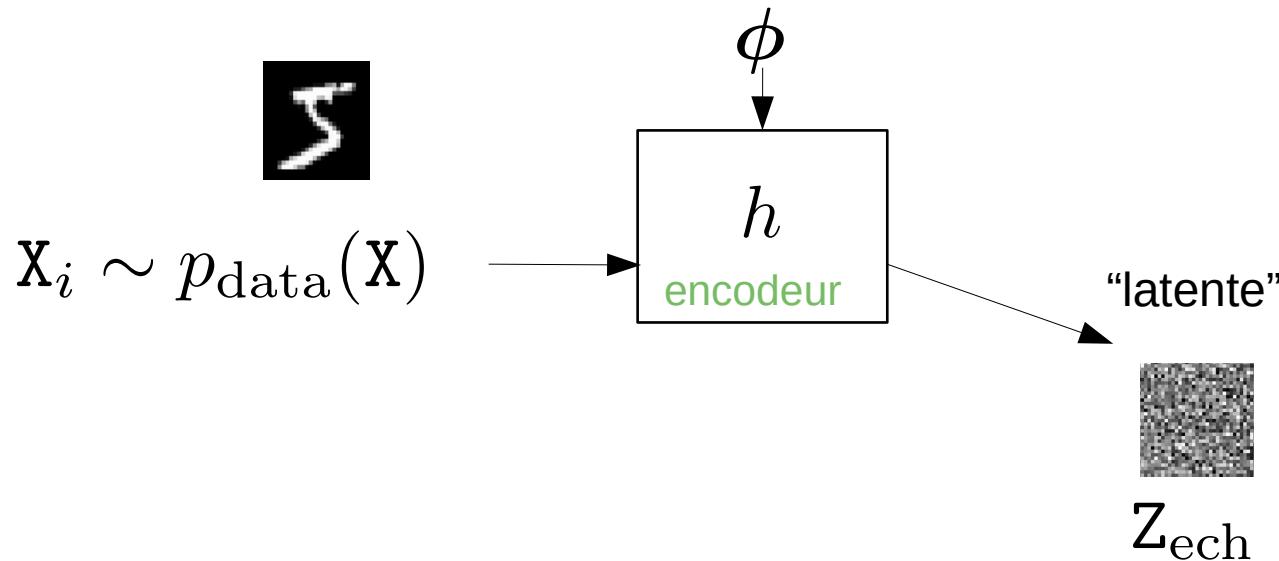


$$\mathbf{X}_i \sim p_{\text{data}}(\mathbf{X})$$

Auto-encodeur variationnel : principe

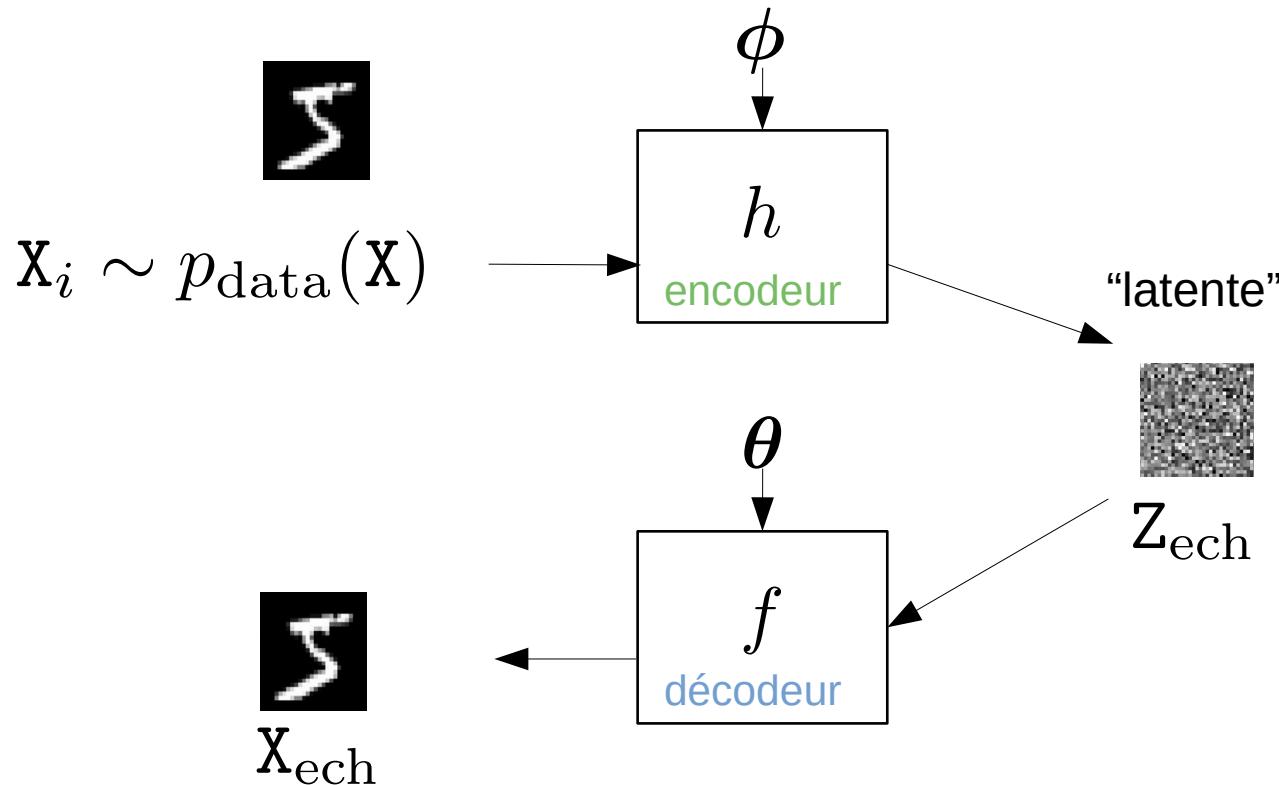


Auto-encodeur variationnel : principe

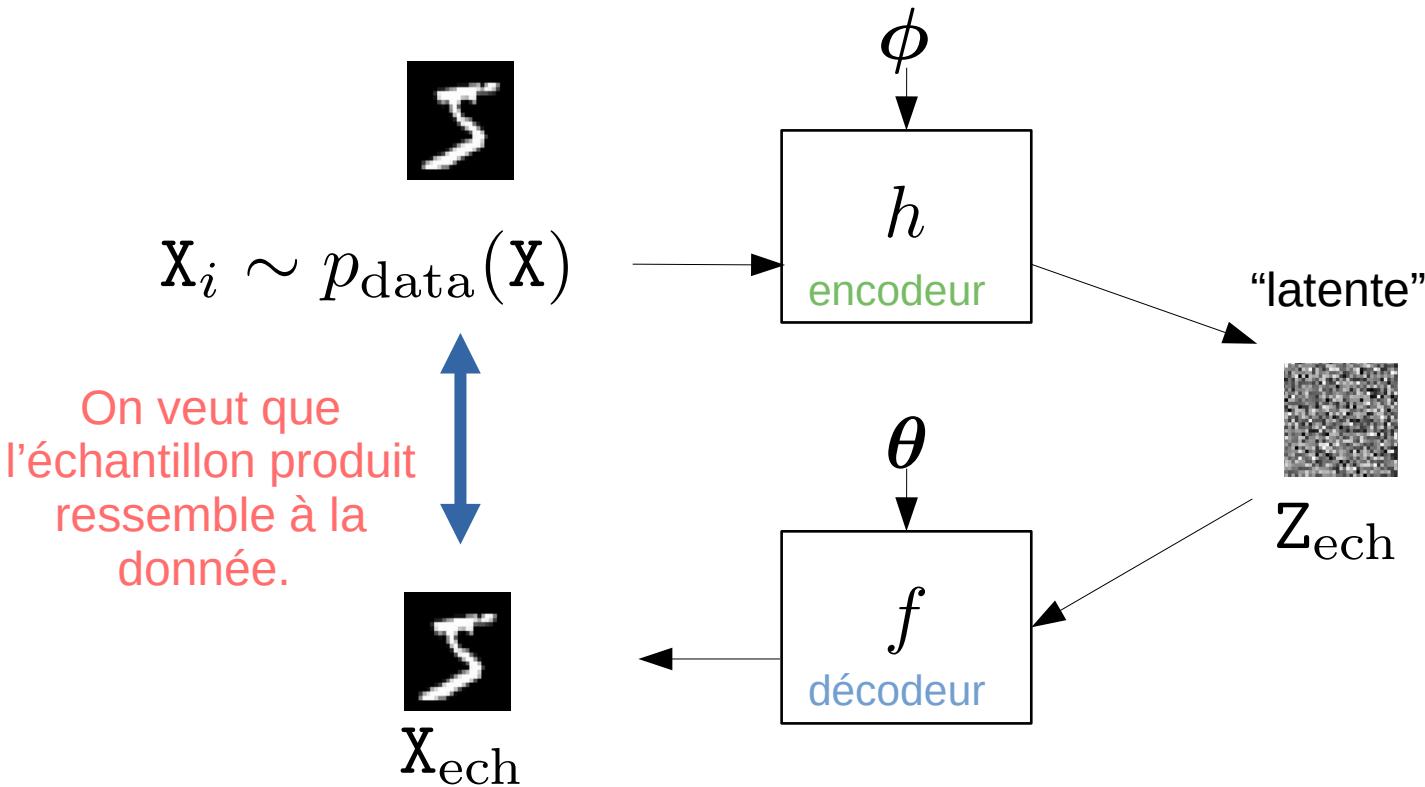


L'espace latent est en général plus petit que l'entrée, il “comprime” ou “résume” l'information

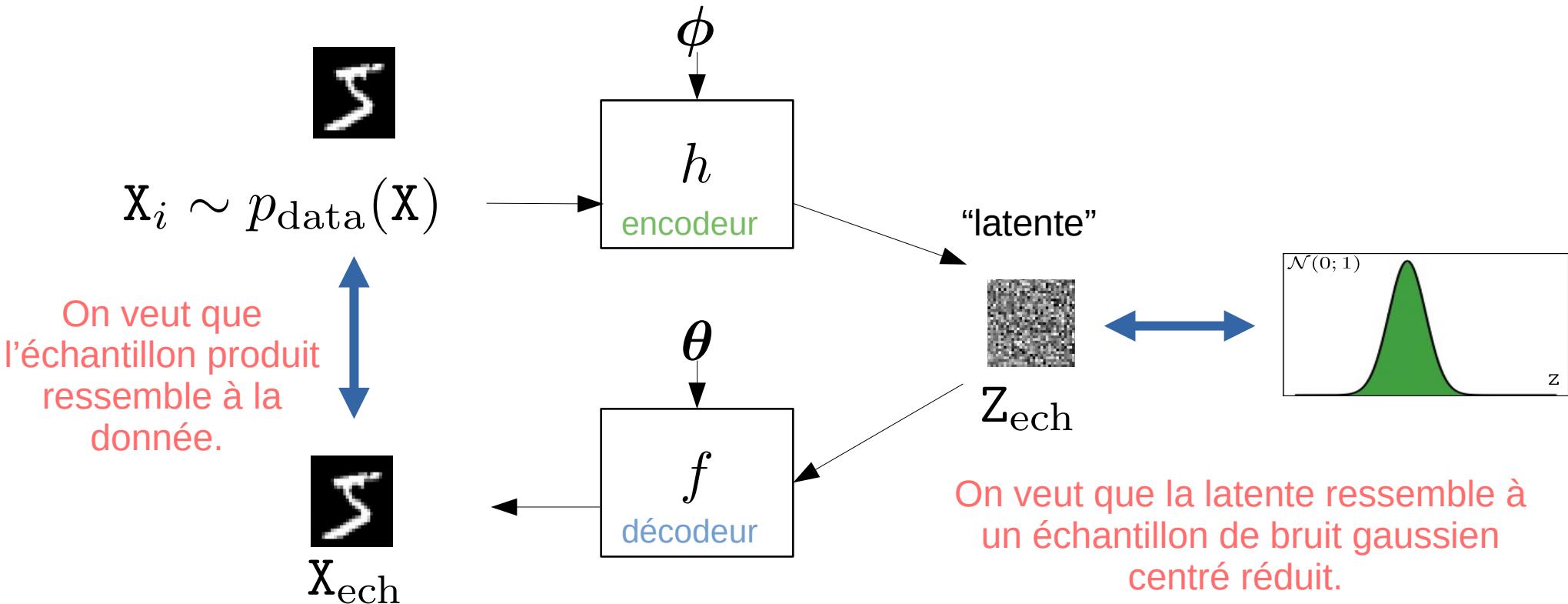
Auto-encodeur variationnel : principe



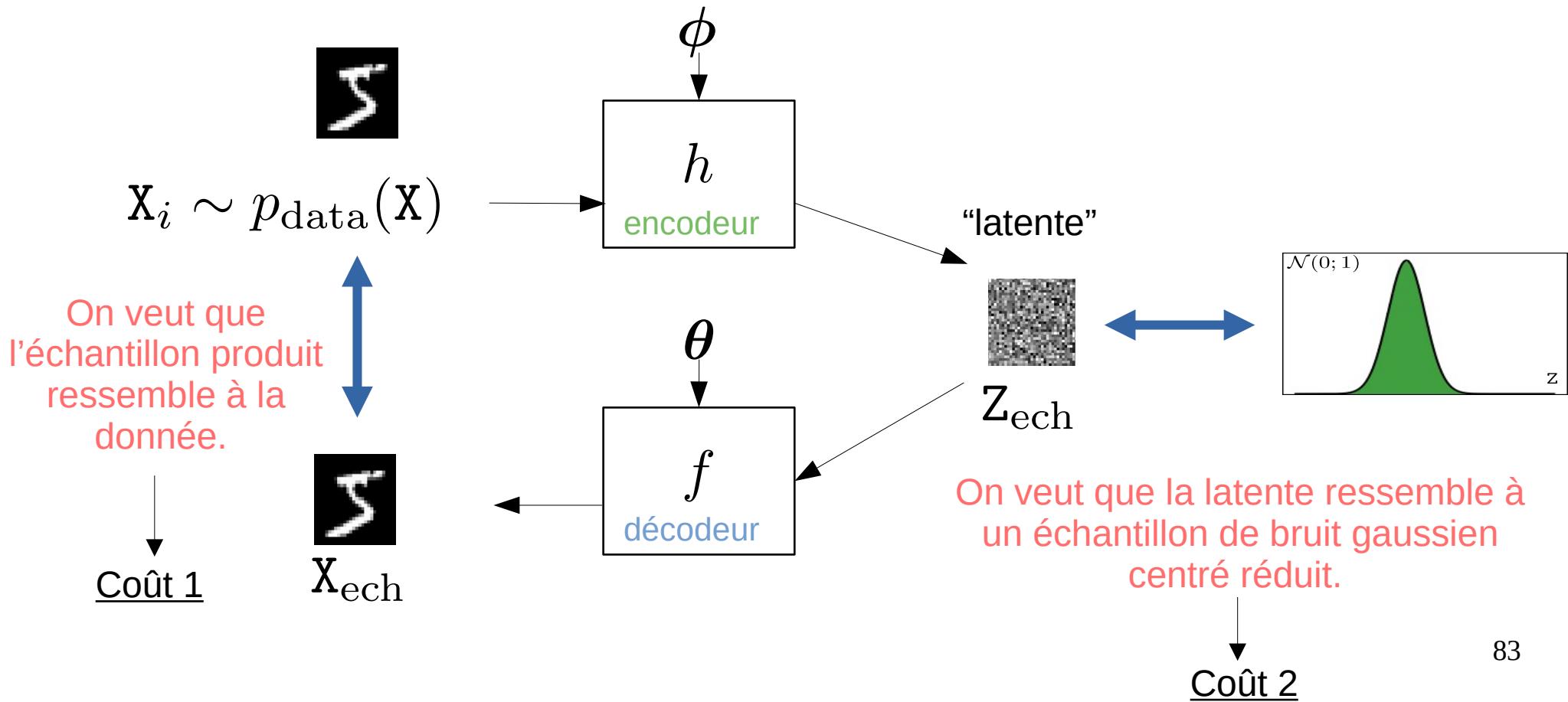
Auto-encodeur variationnel : principe



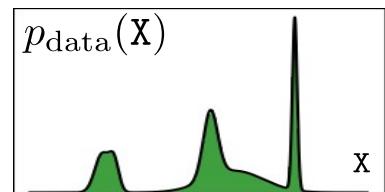
Auto-encodeur variationnel : principe



Auto-encodeur variationnel : principe

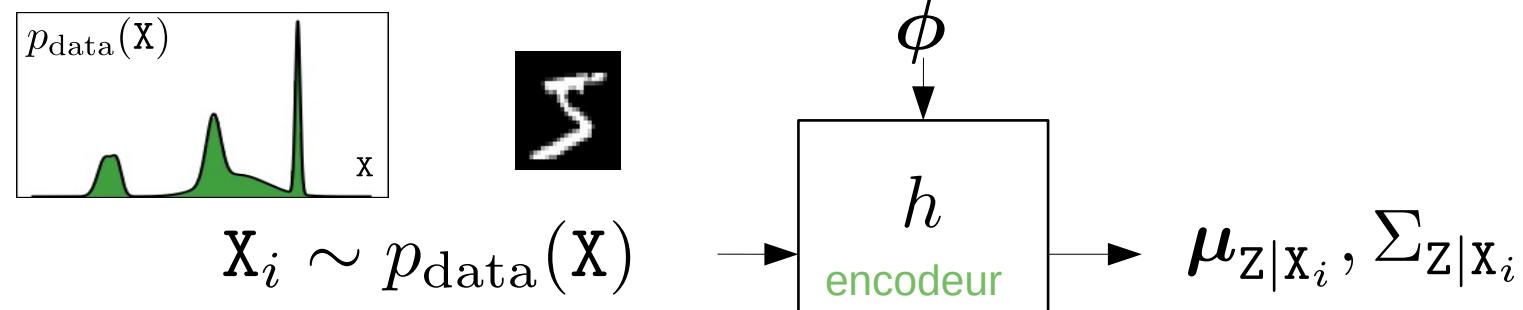


Auto-encodeur variationnel : encodeur



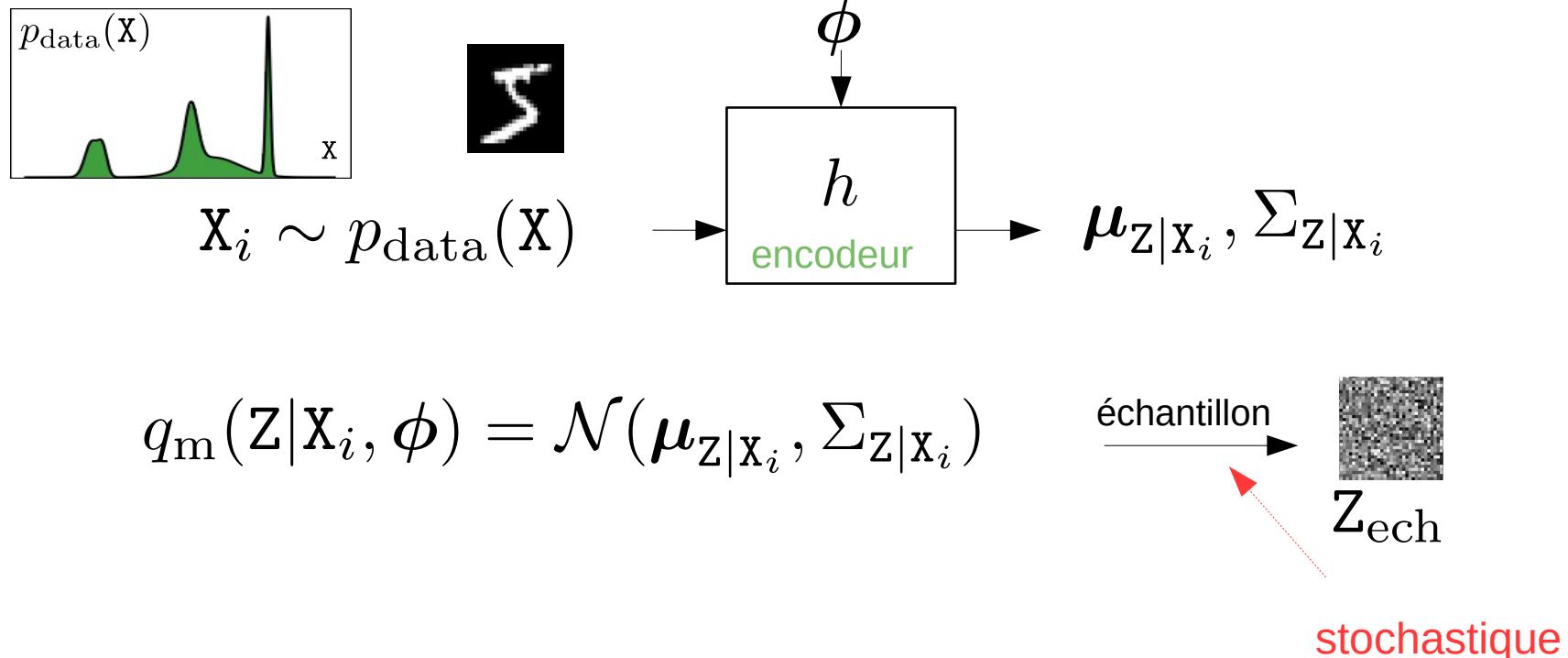
$$\mathbf{X}_i \sim p_{\text{data}}(\mathbf{x})$$

Auto-encodeur variationnel : encodeur

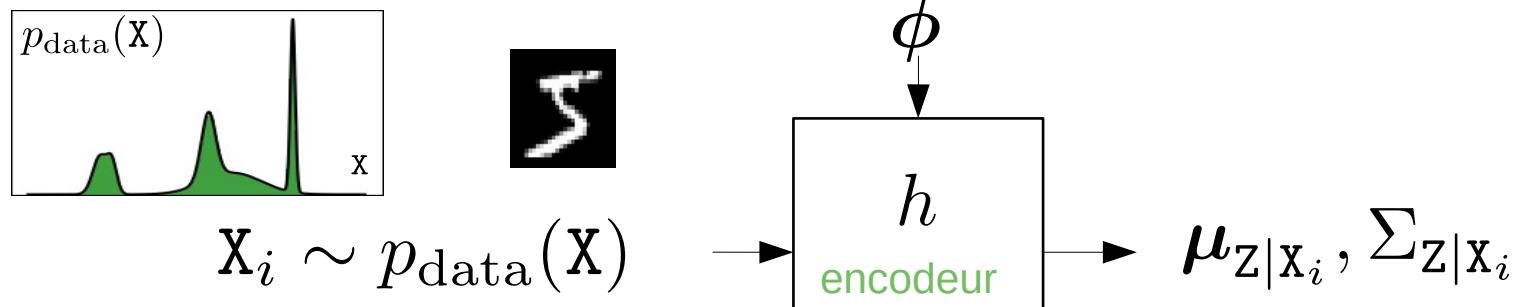


$$q_m(\mathbf{z}|\mathbf{X}_i, \phi) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}|\mathbf{X}_i}, \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{X}_i})$$

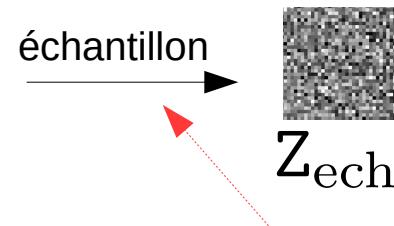
Auto-encodeur variationnel : encodeur



Auto-encodeur variationnel : encodeur



$$q_m(Z|X_i, \phi) = \mathcal{N}(\mu_{Z|x_i}, \Sigma_{Z|x_i})$$



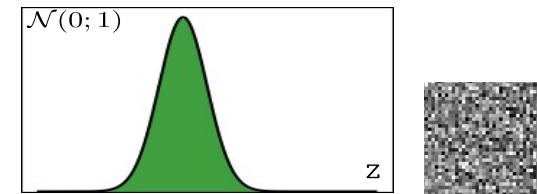
Échantillonnage de l'encodeur

$$Z_{\text{ech}} = \mu_{Z|x_i} + \Sigma_{Z|x_i}^{1/2} \epsilon_{\text{ech}}$$

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

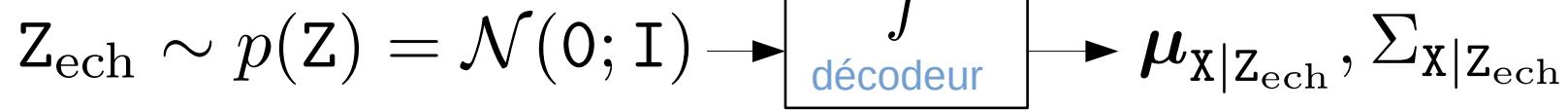
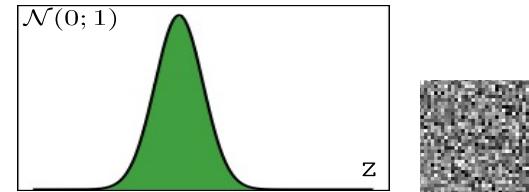
stochastique

Auto-encodeur variationnel : décodeur



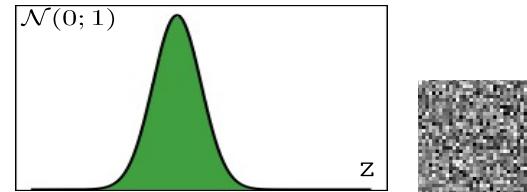
$$\mathbf{z}_{\text{ech}} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{0}; \mathbf{I})$$

Auto-encodeur variationnel : décodeur

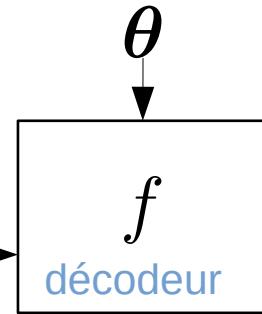


$$p_m(x|z_{\text{ech}}, \theta) = \mathcal{N}(\mu_{X|Z_{\text{ech}}}, \Sigma_{X|Z_{\text{ech}}})$$

Auto-encodeur variationnel : décodeur



$Z_{\text{ech}} \sim p(z) = \mathcal{N}(0; I) \rightarrow$



$\mu_{X|Z_{\text{ech}}}, \Sigma_{X|Z_{\text{ech}}}$

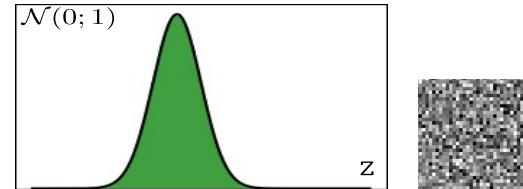
$p_m(x|Z_{\text{ech}}, \theta) = \mathcal{N}(\mu_{X|Z_{\text{ech}}}, \Sigma_{X|Z_{\text{ech}}}) \xrightarrow{\text{échantillon}}$



X_{ech}

stochastique

Auto-encodeur variationnel : décodeur



$$z_{\text{ech}} \sim p(z) = \mathcal{N}(0; I) \rightarrow \begin{array}{c} \theta \\ \downarrow \\ f \\ \text{décodeur} \end{array} \rightarrow \mu_{x|z_{\text{ech}}}, \Sigma_{x|z_{\text{ech}}}$$

$$p_m(x|z_{\text{ech}}, \theta) = \mathcal{N}(\mu_{x|z_{\text{ech}}}, \Sigma_{x|z_{\text{ech}}}) \xrightarrow{\text{échantillon}} \begin{array}{c} \text{échantillon} \\ \xrightarrow{} \\ \mathfrak{x}_{\text{ech}} \end{array}$$

Échantillonnage du décodeur

$$\begin{aligned} x_{\text{ech}} &= \mu_{x|z_{\text{ech}}} + \Sigma_{x|z_{\text{ech}}}^{1/2} \epsilon_{\text{ech}} \\ \epsilon_{\text{ech}} &\sim \mathcal{N}(0; I) \end{aligned}$$

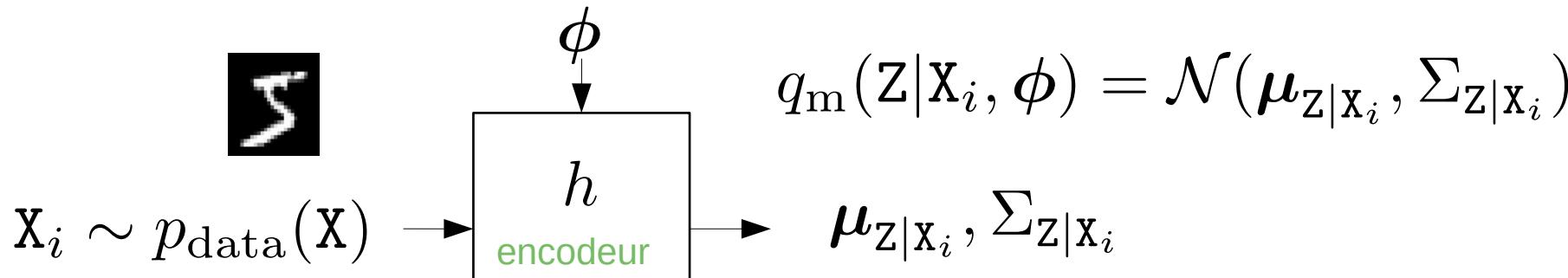
stochastique

Auto-encodeur variationnel : principe détaillé

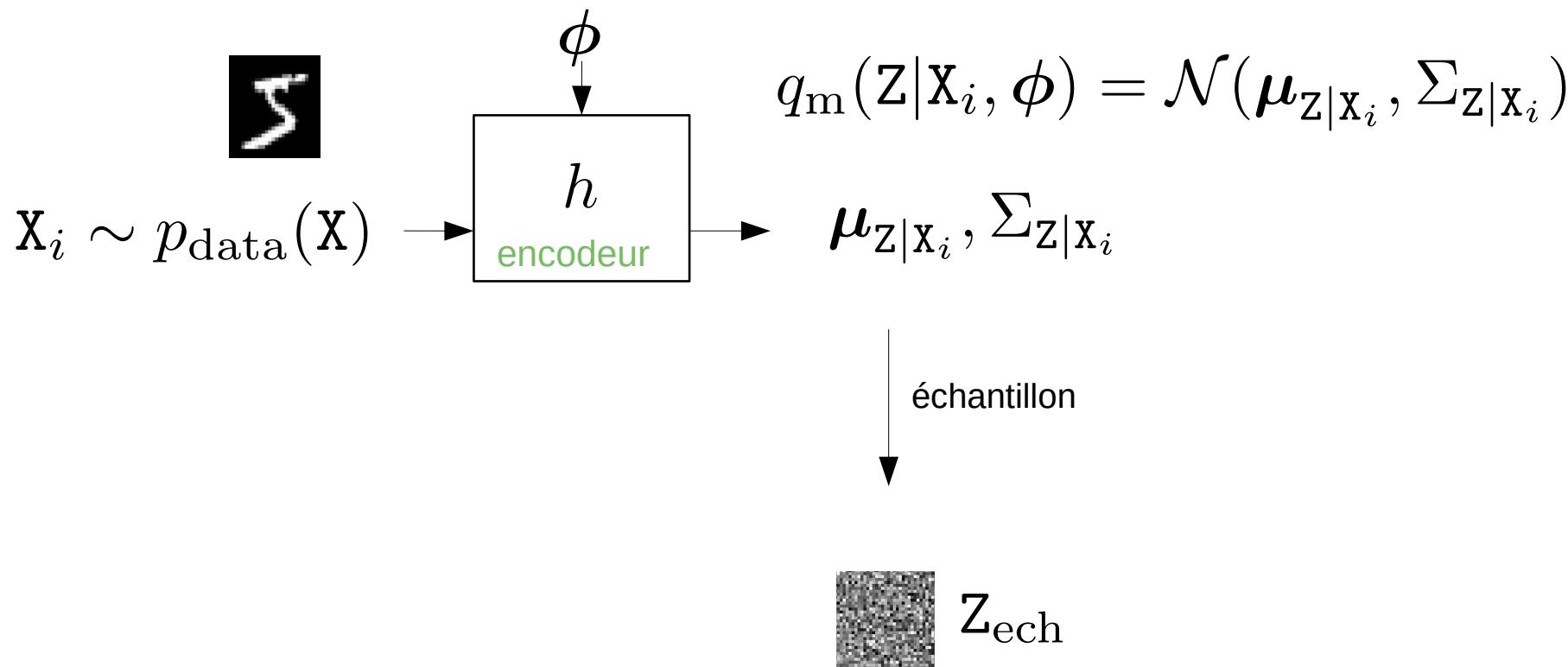


$$\mathbf{x}_i \sim p_{\text{data}}(\mathbf{x})$$

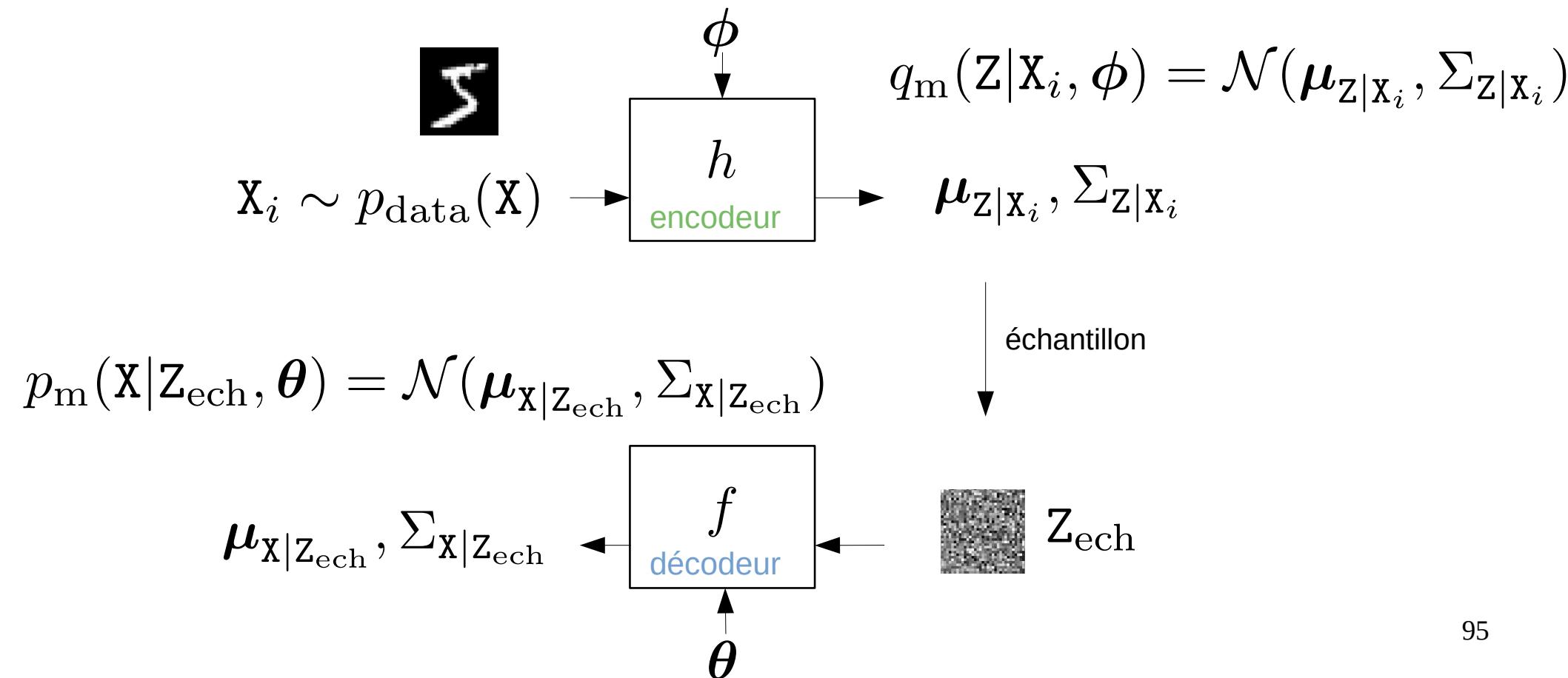
Auto-encodeur variationnel : principe détaillé



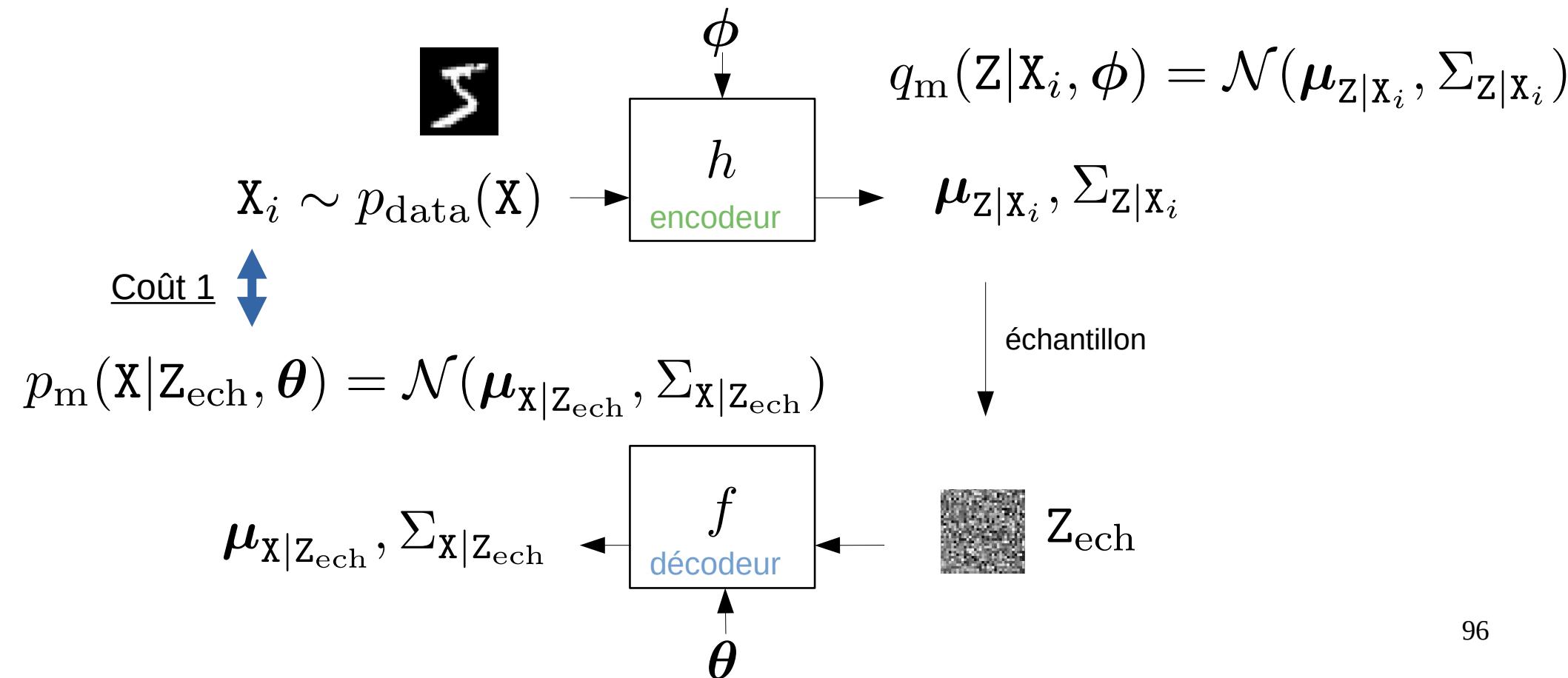
Auto-encodeur variationnel : principe détaillé



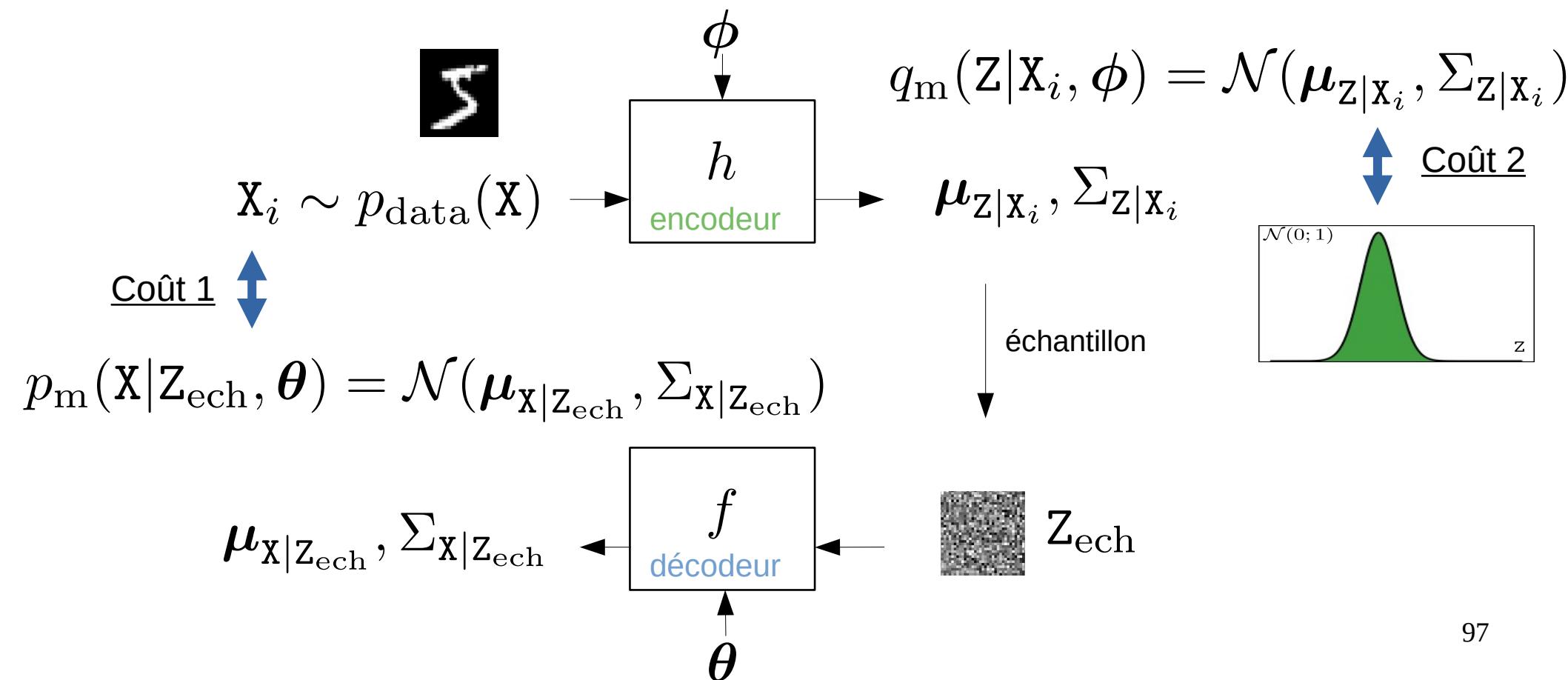
Auto-encodeur variationnel : principe détaillé



Auto-encodeur variationnel : principe détaillé



Auto-encodeur variationnel : principe détaillé



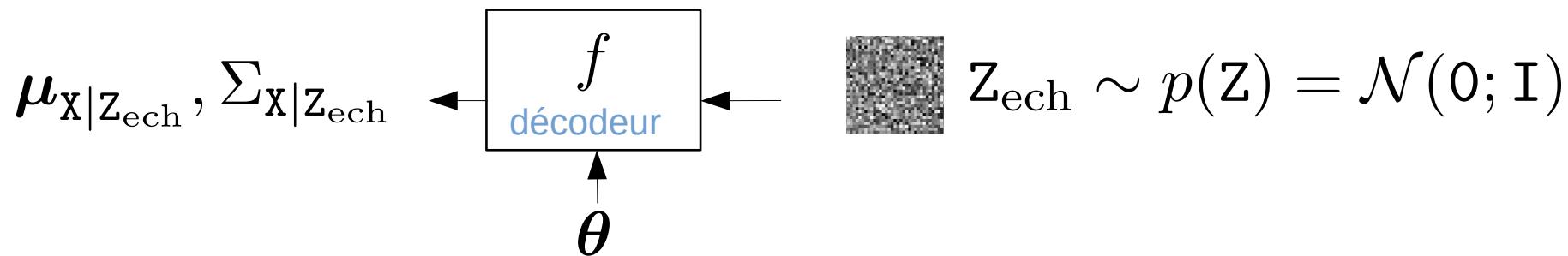
Auto-encodeur variationnel : formalisme

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

Auto-encodeur variationnel : formalisme

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

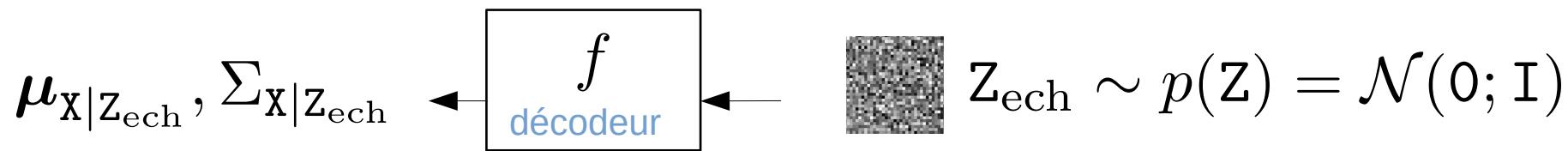
$$p_m(\mathbf{X}|Z_{\text{ech}}, \theta) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}|Z_{\text{ech}}}, \Sigma_{\mathbf{X}|Z_{\text{ech}}})$$



Auto-encodeur variationnel : formalisme

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

$$p_m(\mathbf{X}|Z_{\text{ech}}, \theta) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}|Z_{\text{ech}}}, \Sigma_{\mathbf{X}|Z_{\text{ech}}})$$

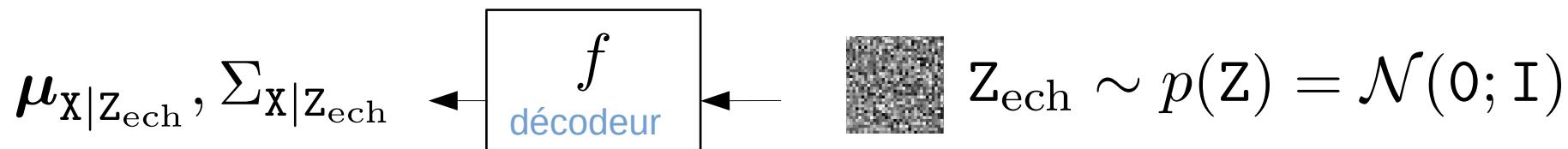


$$p_m(\mathbf{X}|\theta) = \int p_m(\mathbf{X}, Z|\theta) dZ = \int p(Z)p_m(\mathbf{X}|Z, \theta) dZ$$

Auto-encodeur variationnel : formalisme

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

$$p_m(\mathbf{X}|Z_{\text{ech}}, \theta) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}|Z_{\text{ech}}}, \Sigma_{\mathbf{X}|Z_{\text{ech}}})$$



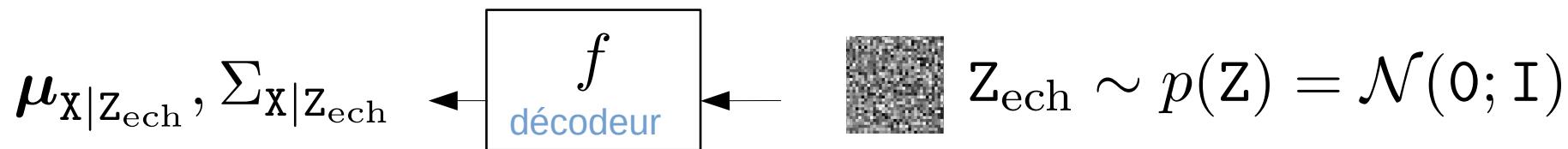
$$p_m(\mathbf{X}|\theta) = \int p_m(\mathbf{X}, Z|\theta) dZ = \int p(Z)p_m(\mathbf{X}|Z, \theta) dZ$$

Intégrale en grande dimension, trop coûteux à calculer !

Auto-encodeur variationnel : formalisme

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

$$p_m(\mathbf{X}|Z_{\text{ech}}, \theta) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}|Z_{\text{ech}}}, \Sigma_{\mathbf{X}|Z_{\text{ech}}})$$



$$p_m(\mathbf{X}|\theta) = \int p_m(\mathbf{X}, \mathbf{Z}|\theta) d\mathbf{Z} = \int p(\mathbf{Z}) p_m(\mathbf{X}|\mathbf{Z}, \theta) d\mathbf{Z}$$

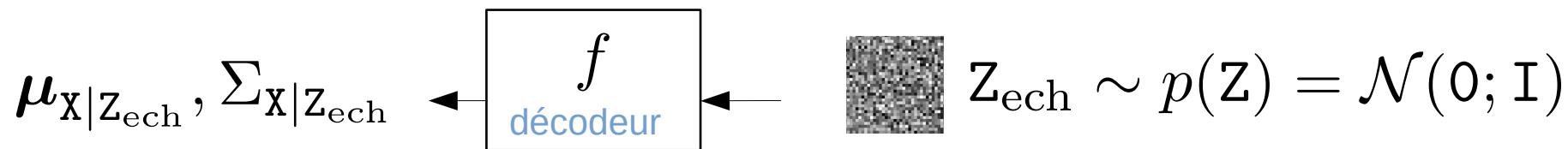
Intégrale en grande dimension, trop coûteux à calculer !

→ On ne peut pas calculer $p_m(\mathbf{X}|\theta)$ (on sait juste l'échantillonner)

Auto-encodeur variationnel : formalisme

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

$$p_m(\mathbf{X}|Z_{\text{ech}}, \theta) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{X}|Z_{\text{ech}}}, \Sigma_{\mathbf{X}|Z_{\text{ech}}})$$



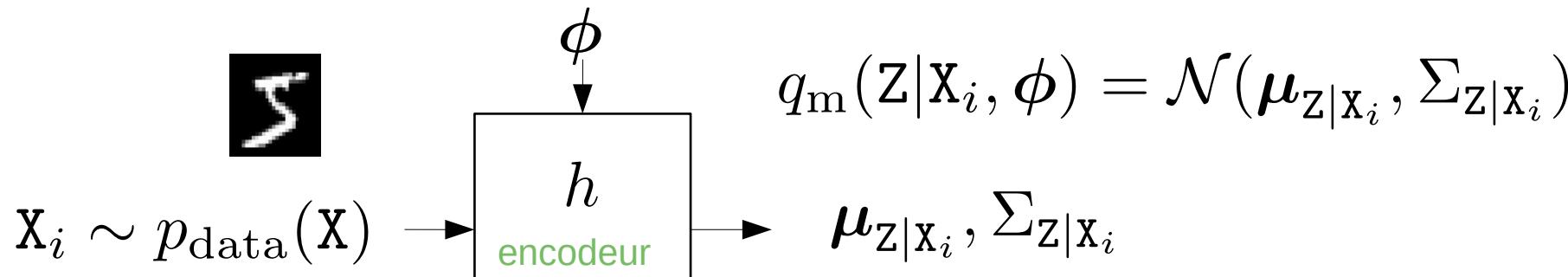
$$p_m(\mathbf{X}|\theta) = \int p_m(\mathbf{X}, \mathbf{Z}|\theta) d\mathbf{Z} = \int p(\mathbf{Z}) p_m(\mathbf{X}|\mathbf{Z}, \theta) d\mathbf{Z}$$

Intégrale en grande dimension, trop coûteux à calculer !

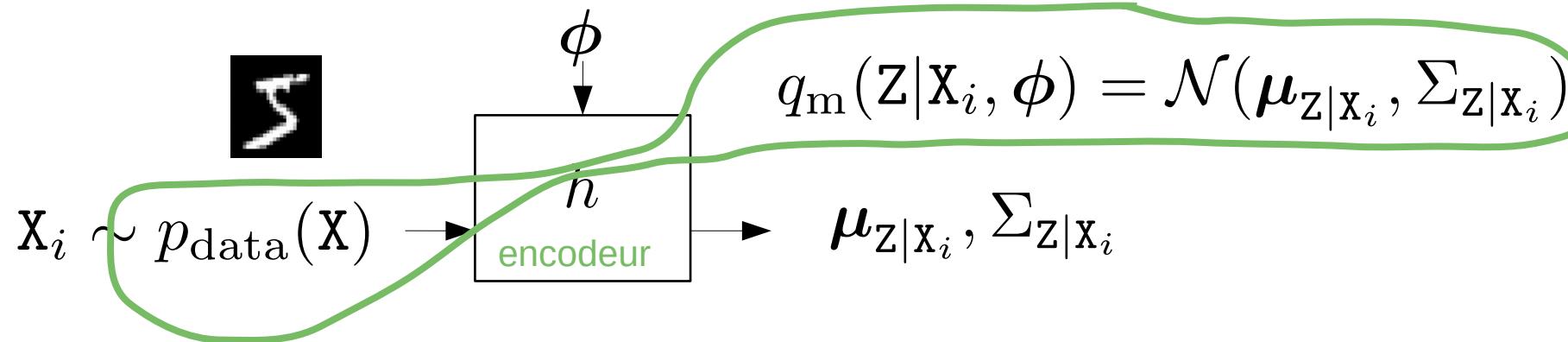
→ On ne peut pas calculer $p_m(\mathbf{X}|\theta)$ (on sait juste l'échantillonner)

$$\cancel{KL(p_{\text{data}}(\mathbf{X}) || p_{\text{modele}}(\mathbf{X}|\theta))}$$

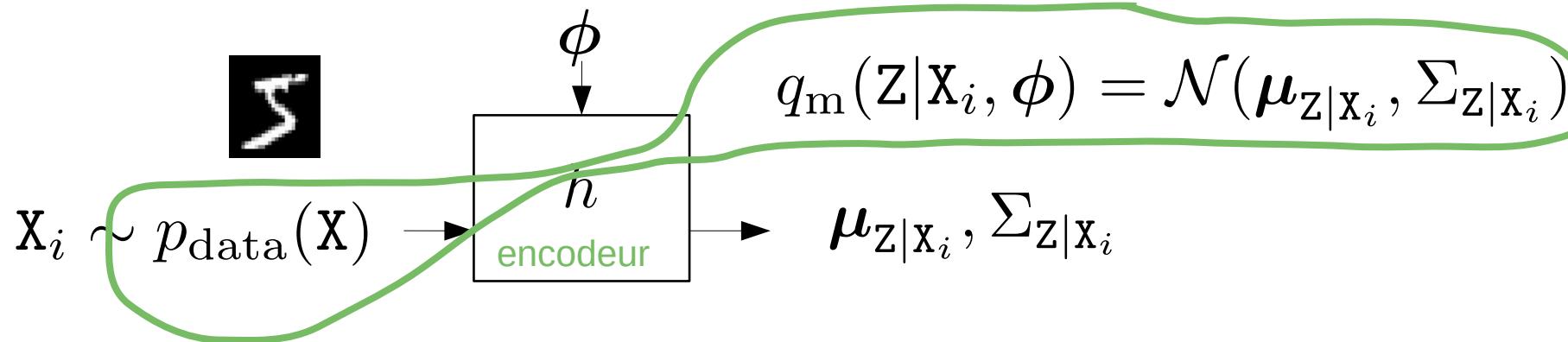
Auto-encodeur variationnel : formalisme (suite)



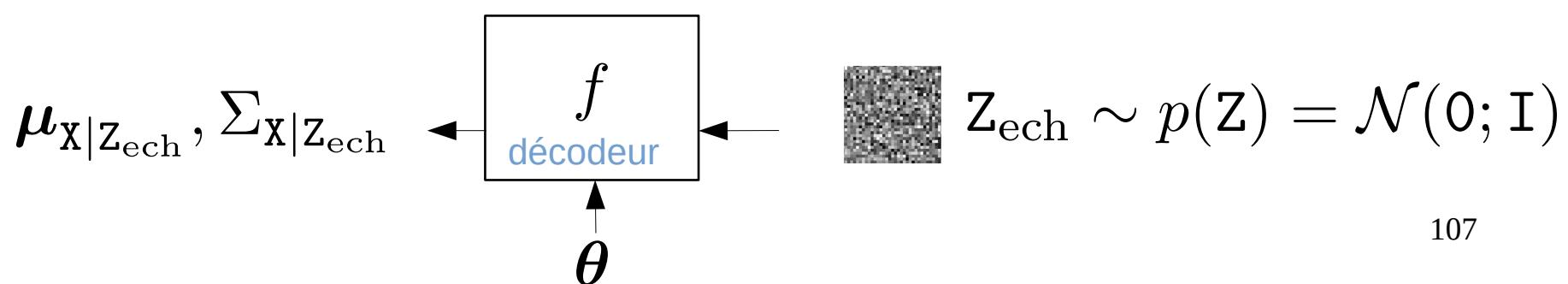
Auto-encodeur variationnel : formalisme (suite)



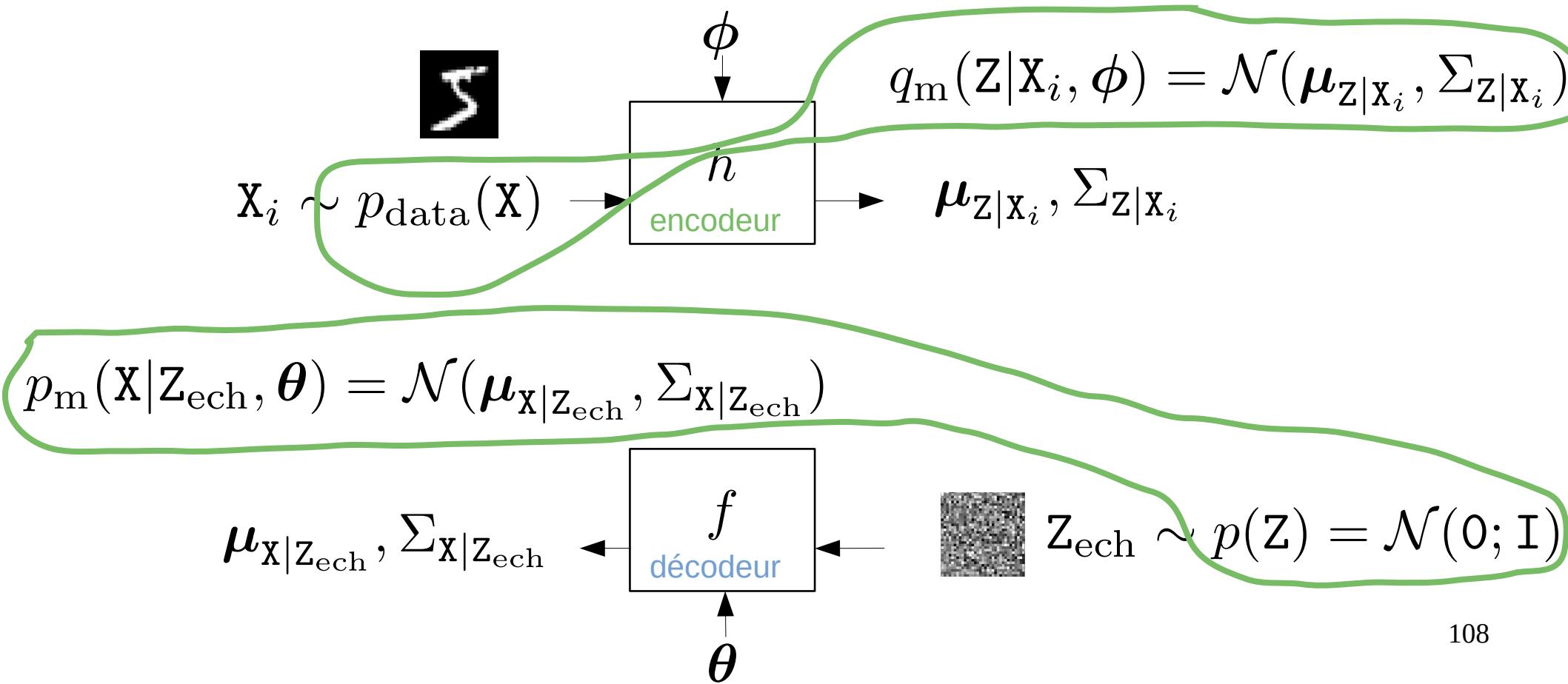
Auto-encodeur variationnel : formalisme (suite)



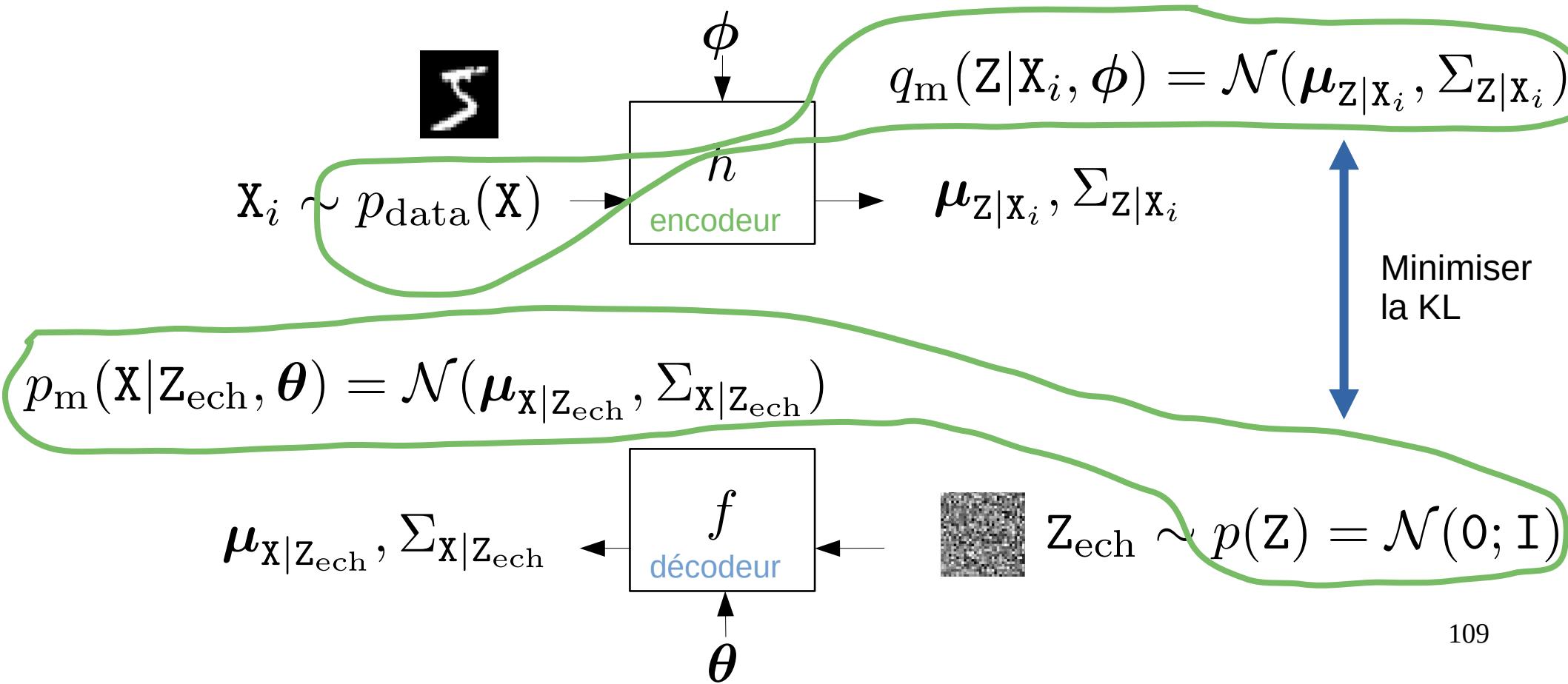
$$p_m(\mathbf{x}|\mathbf{z}_{\text{ech}}, \theta) = \mathcal{N}(\mu_{\mathbf{x}|\mathbf{z}_{\text{ech}}}, \Sigma_{\mathbf{x}|\mathbf{z}_{\text{ech}}})$$



Auto-encodeur variationnel : formalisme (suite)



Auto-encodeur variationnel : formalisme (suite)



Auto-encodeur variationnel : formalisme (suite)

Nouvel objectif : optimiser θ et ϕ pour que

$$p_m(x, z|\theta) = p(z)p_m(x|z, \theta) \approx q_m(x, z|\phi) = p_{\text{data}}(x)q_m(z|x, \phi)$$

Auto-encodeur variationnel : formalisme (suite)

Nouvel objectif : optimiser θ et ϕ pour que

$$p_m(x, z|\theta) = p(z)p_m(x|z, \theta) \approx q_m(x, z|\phi) = p_{\text{data}}(x)q_m(z|x, \phi)$$

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

$$KL(q_m(x, z|\phi)||p_m(x, z|\theta))$$

Auto-encodeur variationnel : formalisme (suite)

Nouvel objectif : optimiser θ et ϕ pour que

$$p_m(x, z|\theta) = p(z)p_m(x|z, \theta) \approx q_m(x, z|\phi) = p_{\text{data}}(x)q_m(z|x, \phi)$$

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

$$\begin{aligned} & KL(q_m(x, z|\phi) || p_m(x, z|\theta)) \\ &= \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})} \left(-\ln \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \right) \end{aligned}$$

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

Auto-encodeur variationnel : formalisme (suite)

Nouvel objectif : optimiser θ et ϕ pour que

$$p_m(x, z|\theta) = p(z)p_m(x|z, \theta) \approx q_m(x, z|\phi) = p_{\text{data}}(x)q_m(z|x, \phi)$$

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

$$\begin{aligned} & KL(q_m(x, z|\phi) || p_m(x, z|\theta)) \\ &= \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})} \left(-\ln \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \right) \quad \text{"Erreur de reconstruction" :} \\ &\quad \text{Incite la sortie du décodeur à ressembler aux } X \end{aligned}$$

$$+ \mathbb{E}_{p_{\text{data}}(x)} (KL(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) || \mathcal{N}(0, I))) \quad \text{"Régularisation" :}$$

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

Auto-encodeur variationnel : formalisme (suite)

Nouvel objectif : optimiser θ et ϕ pour que

$$p_m(x, z|\theta) = p(z)p_m(x|z, \theta) \approx q_m(x, z|\phi) = p_{\text{data}}(x)q_m(z|x, \phi)$$

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

$$\begin{aligned}
 & KL(q_m(x, z|\phi) || p_m(x, z|\theta)) \\
 &= \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})} \left(-\ln \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \right) \quad \text{"Erreur de reconstruction" :} \\
 &\quad \text{Incite la sortie du décodeur à ressembler aux } X \\
 &+ \mathbb{E}_{p_{\text{data}}(x)} (KL(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) || \mathcal{N}(0, I))) \quad \text{"Régularisation" :} \\
 &+ \text{cst}_{\phi, \theta} \quad \text{Incite la sortie de l'encodeur à être centrée réduite} \rightarrow \text{logique car lorsqu'on générera des } z, \text{ on les tirera selon une gaussienne centrée réduite}
 \end{aligned}$$

Auto-encodeur variationnel : formalisme (suite)

Nouvel objectif : optimiser θ et ϕ pour que

$$p_m(x, z|\theta) = p(z)p_m(x|z, \theta) \approx q_m(x, z|\phi) = p_{\text{data}}(x)q_m(z|x, \phi)$$

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

$$\begin{aligned} & KL(q_m(x, z|\phi) || p_m(x, z|\theta)) \\ &= \mathbb{E}_{p_{\text{data}}(x)} \mathbb{E}_{\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})} \left(-\ln \mathcal{N}(\mu_{x|z}, \Sigma_{x|z}) \right) \quad \text{"Erreur de reconstruction"} : \\ &\quad \text{Incite la sortie du décodeur à ressembler aux } X \\ &+ \mathbb{E}_{p_{\text{data}}(x)} (KL(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) || \mathcal{N}(0, I))) \quad \text{"Régularisation"} : \\ &\quad \text{Incite la sortie de l'encodeur à être centrée réduite} \rightarrow \text{logique car lorsqu'on générera des } z, \text{ on les tirera selon une gaussienne centrée réduite} \\ &+ \text{cst}_{\phi, \theta} \end{aligned}$$

Auto-encodeur variationnel : apprentissage

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

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

Auto-encodeur variationnel : apprentissage

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

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

Problème : ϕ intervient dans cette espérance (dans le calcul de ces moyenne et covariance)

Auto-encodeur variationnel : apprentissage

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

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

Astuce de reparamétrisation (“Reparameterization trick”)

$$= \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{\mathcal{N}(0, \mathbf{I})} \left(-\ln \mathcal{N}(\boldsymbol{\mu}_{x|(\boldsymbol{\mu}_{z|x_i} + \Sigma_{z|x_i}^{1/2} \boldsymbol{\epsilon})}, \Sigma_{x|(\boldsymbol{\mu}_{z|x_i} + \Sigma_{z|x_i}^{1/2} \boldsymbol{\epsilon})}) \right) + KL(\mathcal{N}(\boldsymbol{\mu}_{z|x_i}, \Sigma_{z|x_i}) || \mathcal{N}(0, \mathbf{I})) \quad 119$$

Auto-encodeur variationnel : apprentissage (suite)

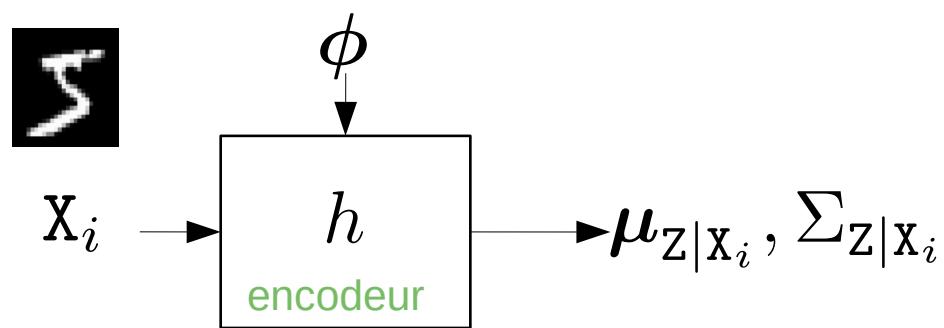
$$L_i(\theta, \phi) =$$



X_i →

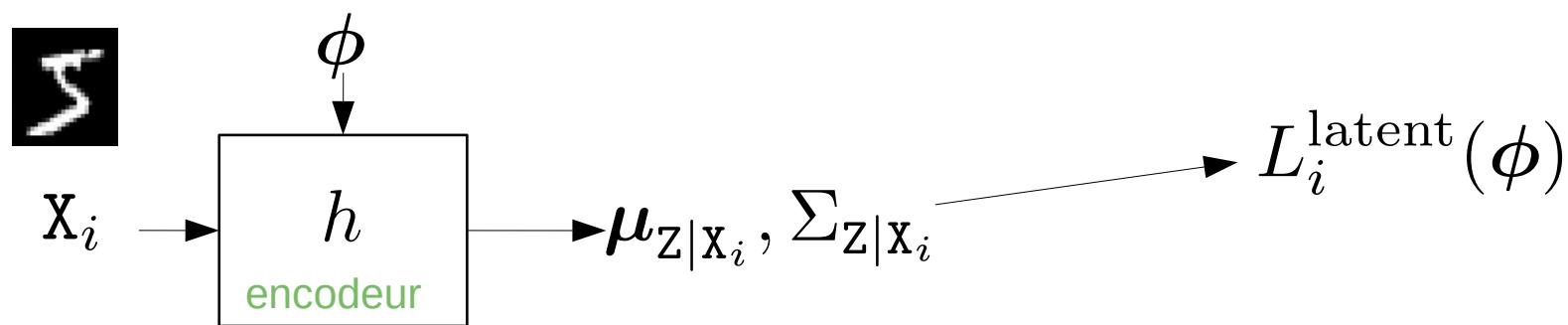
Auto-encodeur variationnel : apprentissage (suite)

$$L_i(\theta, \phi) =$$



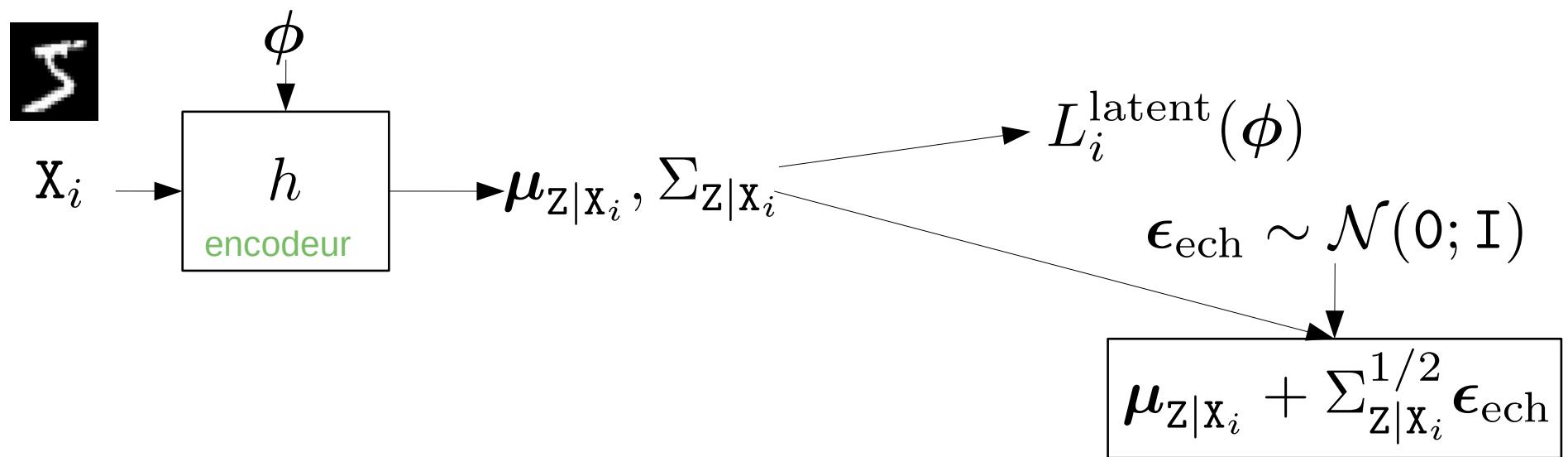
Auto-encodeur variationnel : apprentissage (suite)

$$L_i(\theta, \phi) = KL(\mathcal{N}(\mu_{z|x_i}, \Sigma_{z|x_i}) || \mathcal{N}(0, I))$$



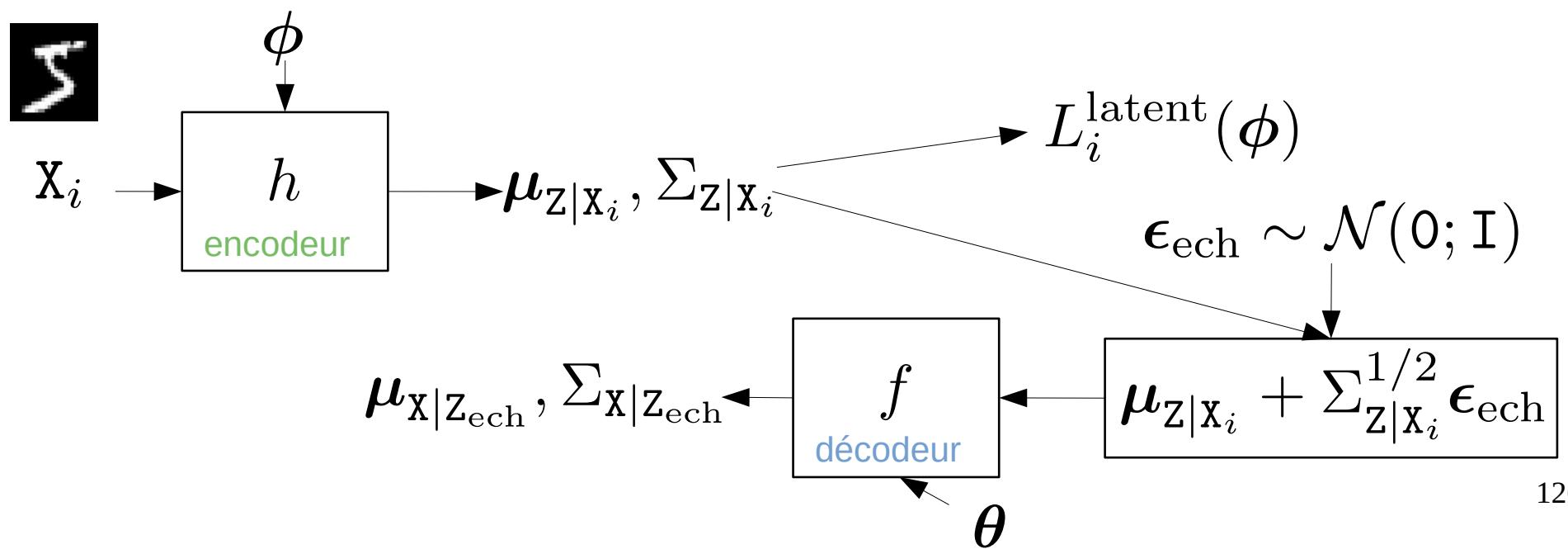
Auto-encodeur variationnel : apprentissage (suite)

$$L_i(\theta, \phi) = KL(\mathcal{N}(\mu_{z|x_i}, \Sigma_{z|x_i}) || \mathcal{N}(0, I))$$



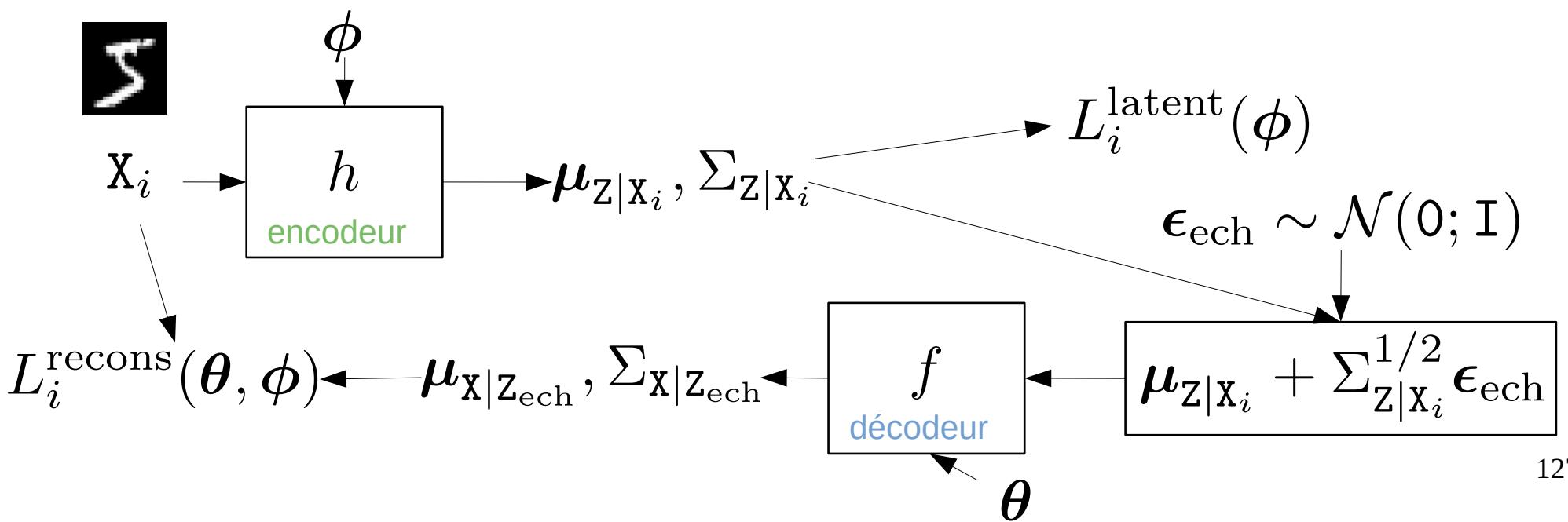
Auto-encodeur variationnel : apprentissage (suite)

$$L_i(\theta, \phi) = KL(\mathcal{N}(\mu_{z|x_i}, \Sigma_{z|x_i}) || \mathcal{N}(0, I))$$



Auto-encodeur variationnel : apprentissage (suite)

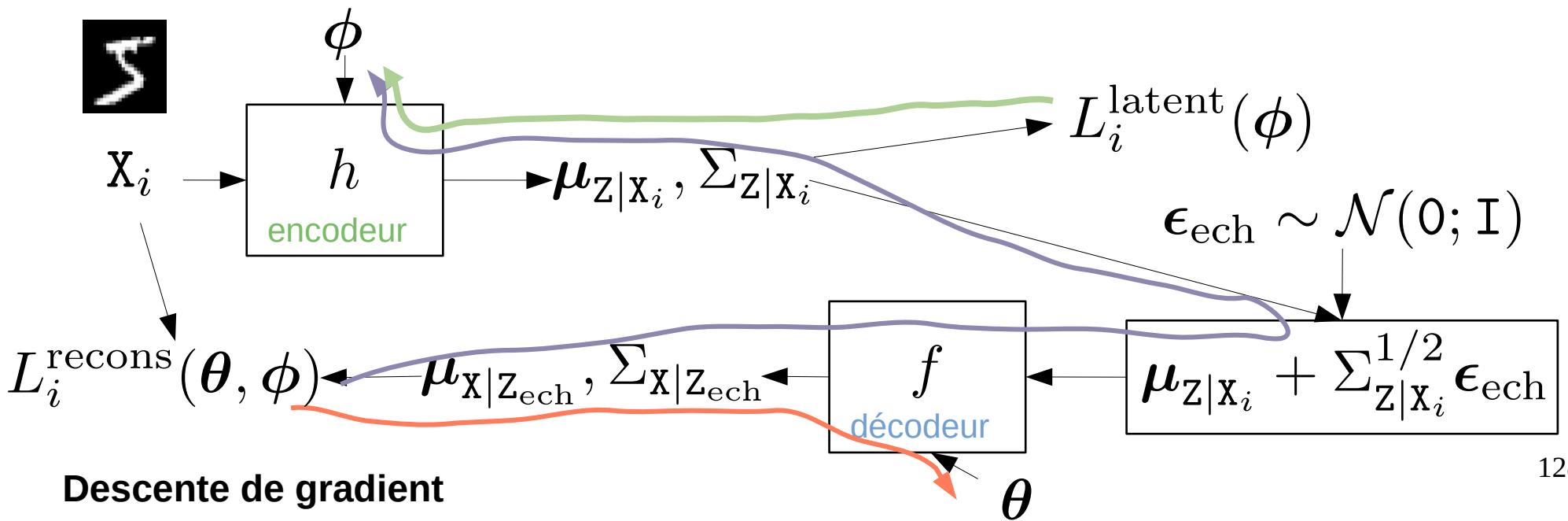
$$\begin{aligned}
 L_i(\theta, \phi) &= KL(\mathcal{N}(\mu_{z|x_i}, \Sigma_{z|x_i}) || \mathcal{N}(0, \mathbf{I})) \\
 &\quad - \ln \mathcal{N}(\mu_{x|(\mu_{z|x_i} + \Sigma_{z|x_i}^{1/2} \epsilon_{\text{ech}})}, \Sigma_{x|(\mu_{z|x_i} + \Sigma_{z|x_i}^{1/2} \epsilon_{\text{ech}})})
 \end{aligned}$$



Auto-encodeur variationnel : apprentissage (suite)

$$L_i(\theta, \phi) = KL(\mathcal{N}(\mu_{z|x_i}, \Sigma_{z|x_i}) || \mathcal{N}(0, I))$$

$$- \ln \mathcal{N}(\mu_{x|(\mu_{z|x_i} + \Sigma_{z|x_i}^{1/2} \epsilon_{\text{ech}})}, \Sigma_{x|(\mu_{z|x_i} + \Sigma_{z|x_i}^{1/2} \epsilon_{\text{ech}})})$$



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) Modèle de diffusion (DDPM)

v)

Modèle de diffusion : idée générale

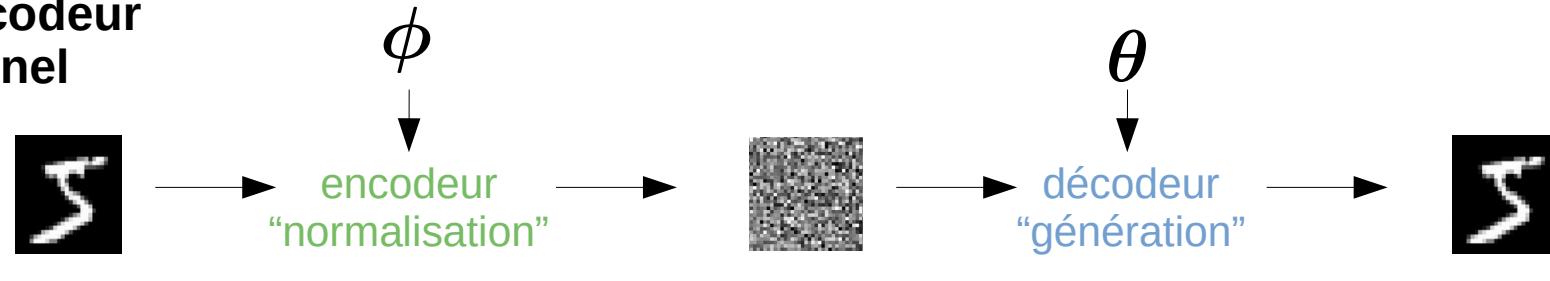
**Auto-encodeur
variationnel**



Transformations stochastiques

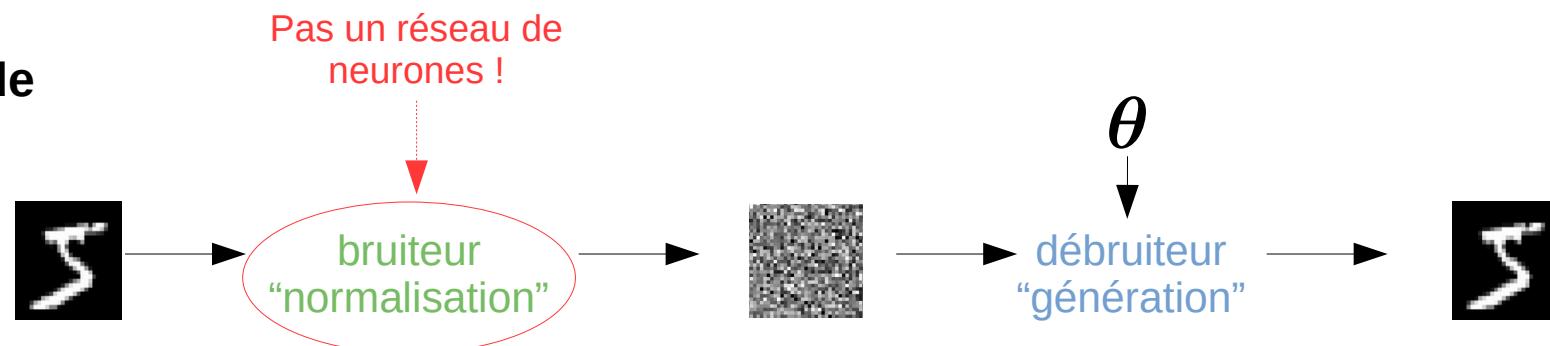
Modèle de diffusion : idée générale

Auto-encodeur variationnel



Transformations stochastiques

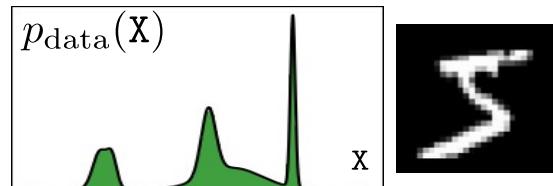
Modèle de diffusion



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

v)

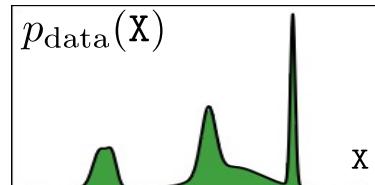
Bruiteur (Diffusion)



$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

v)

Bruiteur (Diffusion)

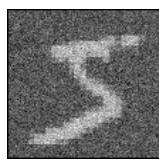
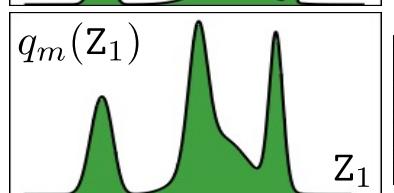
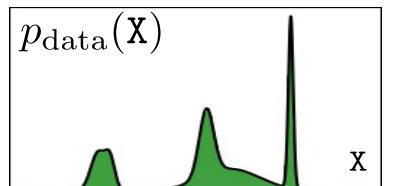


$$X_{\text{ech}} \sim p_{\text{data}}(x)$$



$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; I)$$

Bruiteur (Diffusion)



$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

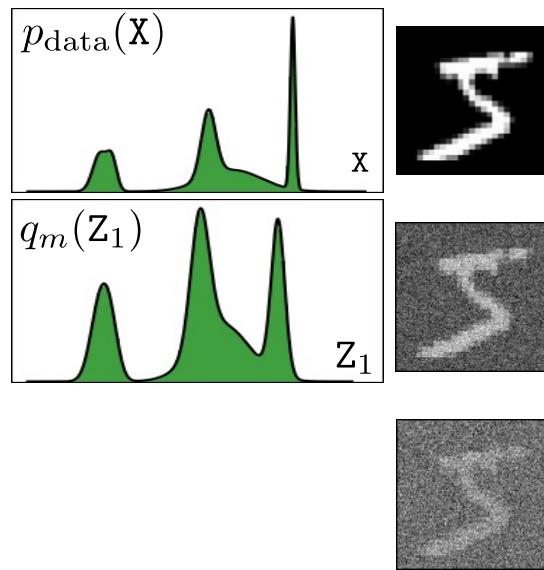


$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; I)$$



On a légèrement “empâté” $p_{\text{data}}(x)$

Bruiteur (Diffusion)

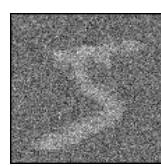
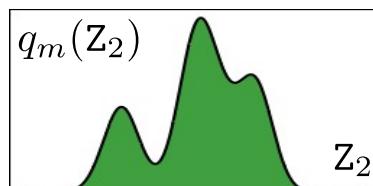
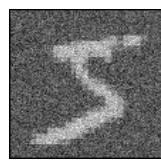
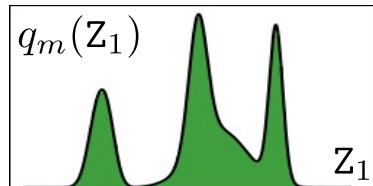
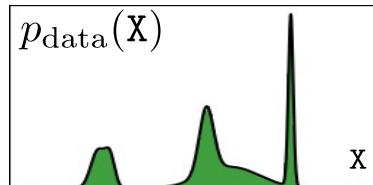


$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; I)$$

$$Z_2 = \sqrt{1 - \beta_2} Z_1 + \sqrt{\beta_2} \epsilon_2 \leftarrow \epsilon_2 \sim \mathcal{N}(0; I)$$

Bruiteur (Diffusion)



$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

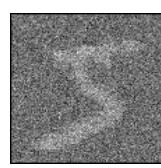
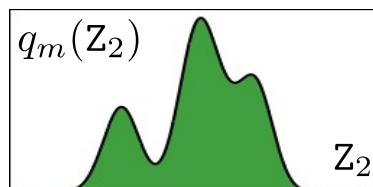
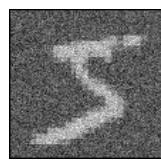
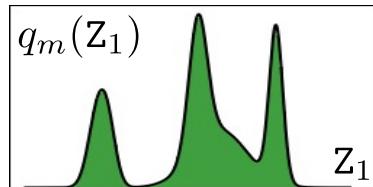
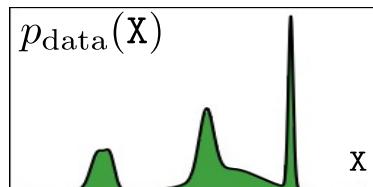
$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; I)$$

$$Z_2 = \sqrt{1 - \beta_2} Z_1 + \sqrt{\beta_2} \epsilon_2 \leftarrow \epsilon_2 \sim \mathcal{N}(0; I)$$

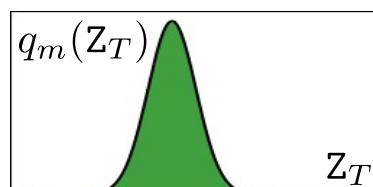


On a encore un peu plus légèrement “empâté” $p_{\text{data}}(X)$

Bruiteur (Diffusion)



⋮



$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

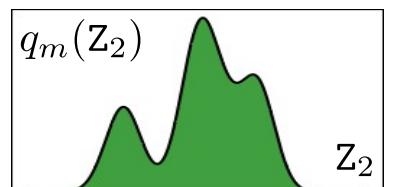
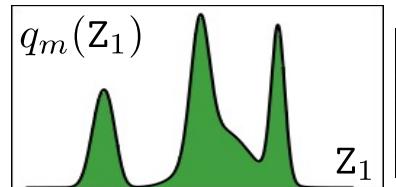
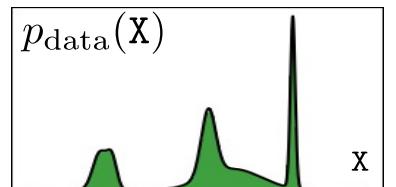
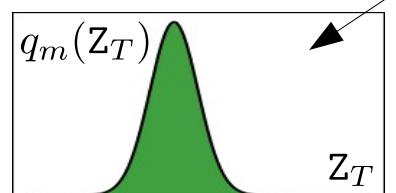
$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; I)$$

$$Z_2 = \sqrt{1 - \beta_2} Z_1 + \sqrt{\beta_2} \epsilon_2 \leftarrow \epsilon_2 \sim \mathcal{N}(0; I)$$

⋮

$$Z_T = \sqrt{1 - \beta_T} Z_{T-1} + \sqrt{\beta_T} \epsilon_T \leftarrow \epsilon_T \sim \mathcal{N}(0; I)$$

Bruiteur (Diffusion)


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


$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

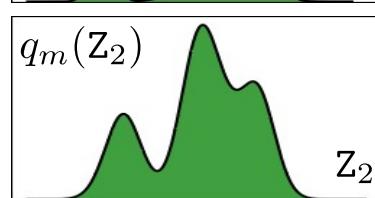
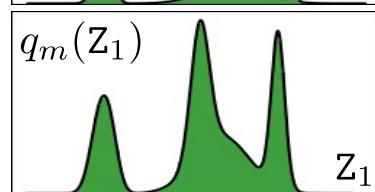
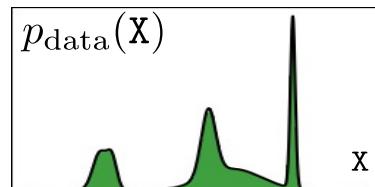
$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; \mathbf{I})$$

$$Z_2 = \sqrt{1 - \beta_2} Z_1 + \sqrt{\beta_2} \epsilon_2 \leftarrow \epsilon_2 \sim \mathcal{N}(0; \mathbf{I})$$

 \vdots

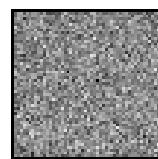
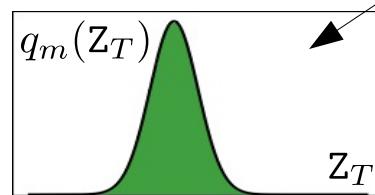
$$Z_T = \sqrt{1 - \beta_T} Z_{T-1} + \sqrt{\beta_T} \epsilon_T \leftarrow \epsilon_T \sim \mathcal{N}(0; \mathbf{I})$$

Bruiteur (Diffusion)



:

$$\approx \mathcal{N}(0; \mathbf{I})$$



$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; \mathbf{I})$$

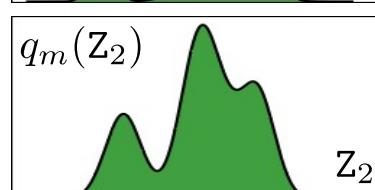
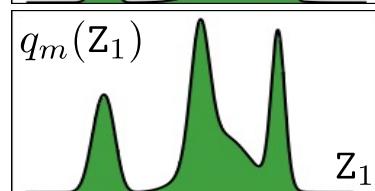
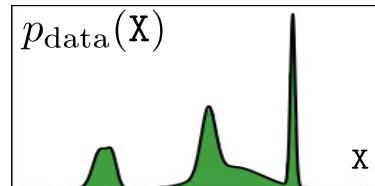
$$Z_2 = \sqrt{1 - \beta_2} Z_1 + \sqrt{\beta_2} \epsilon_2 \leftarrow \epsilon_2 \sim \mathcal{N}(0; \mathbf{I})$$

Il faut choisir les $\{\beta_t\}_{t=1 \dots T}$ de telle sorte que

$$\prod_{k=1}^T (1 - \beta_k) \rightarrow 0$$

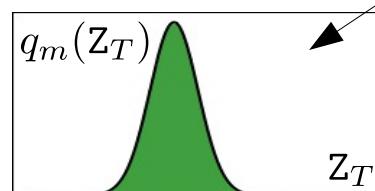
$$Z_T = \sqrt{1 - \beta_T} Z_{T-1} + \sqrt{\beta_T} \epsilon_T \leftarrow \epsilon_T \sim \mathcal{N}(0; \mathbf{I})$$

Bruiteur (Diffusion)



:

$$\approx \mathcal{N}(0; \mathbf{I})$$



$$X_{\text{ech}} \sim p_{\text{data}}(x)$$

$$Z_1 = \sqrt{1 - \beta_1} X_{\text{ech}} + \sqrt{\beta_1} \epsilon_1 \leftarrow \epsilon_1 \sim \mathcal{N}(0; \mathbf{I})$$

$$Z_2 = \sqrt{1 - \beta_2} Z_1 + \sqrt{\beta_2} \epsilon_2 \leftarrow \epsilon_2 \sim \mathcal{N}(0; \mathbf{I})$$

Il faut choisir les $\{\beta_t\}_{t=1 \dots T}$ de telle sorte que

$$\prod_{k=1}^T (1 - \beta_k) \rightarrow 0$$

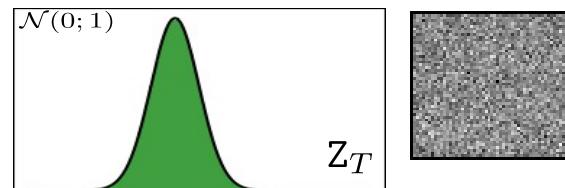
$$Z_T = \sqrt{1 - \beta_T} Z_{T-1} + \sqrt{\beta_T} \epsilon_T \leftarrow \epsilon_T \sim \mathcal{N}(0; \mathbf{I})$$

142

Remarque : On préserve la taille de X tout le temps, on ne peut pas la réduire.

v)

Débruiteur

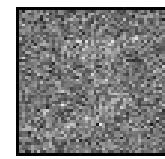
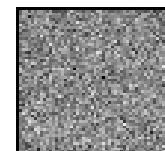
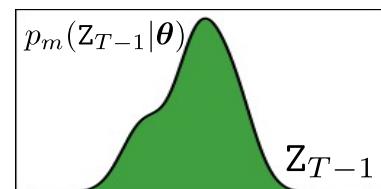
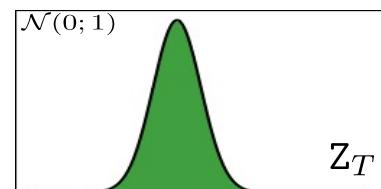


$$\mathbf{z}_T = \mathbf{n}_T$$



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

Débruiteur



$$z_T = n_T$$



$$n_T \sim \mathcal{N}(0; I)$$

$$z_{T-1} = \mu_{\theta_{T-1}}(z_T) + \sigma_{T-1} n_{T-1}$$

$$n_{T-1} \sim \mathcal{N}(0; I)$$

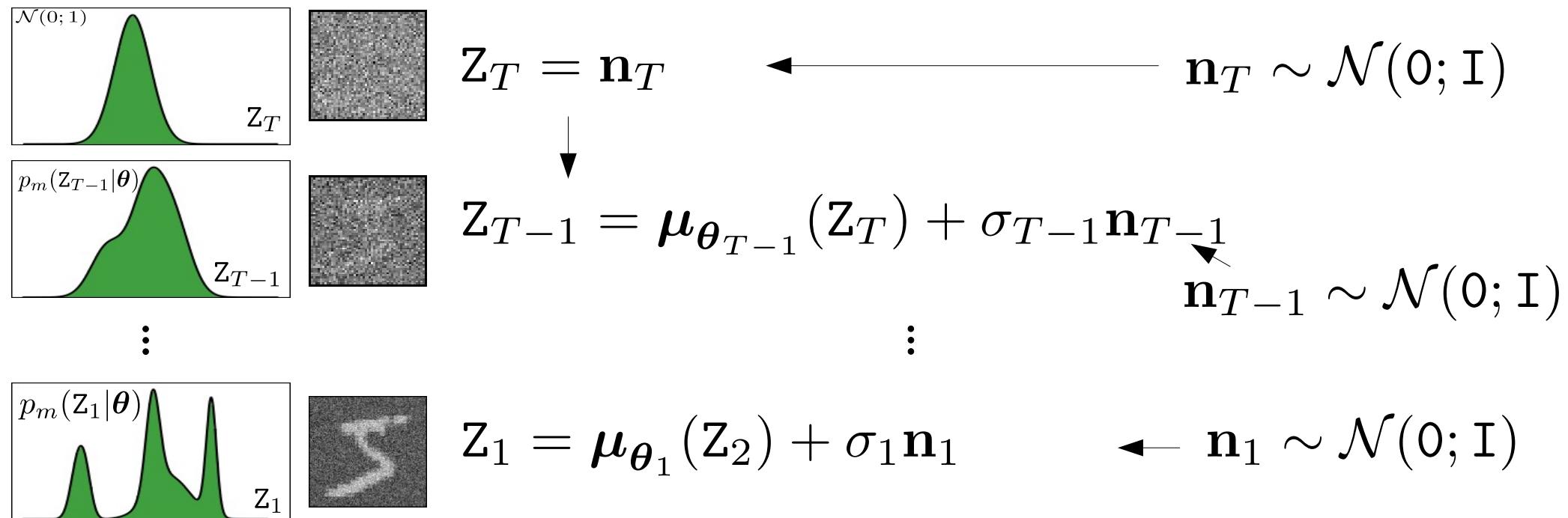
Fonction paramétrique
(réseau de neurones)



Paramètre
généralement fixé
manuellement

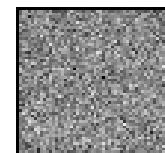
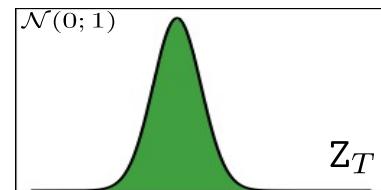


Débruiteur



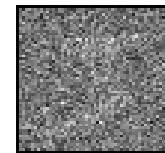
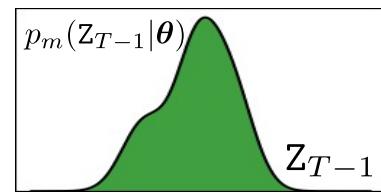
v)

Débruiteur



$$Z_T = \mathbf{n}_T$$

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

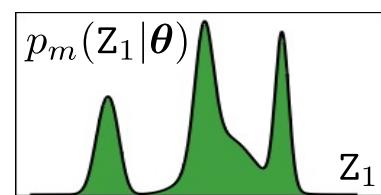


$$Z_{T-1} = \mu_{\theta_{T-1}}(Z_T) + \sigma_{T-1} \mathbf{n}_{T-1}$$

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

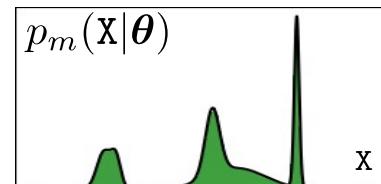
⋮

⋮



$$Z_1 = \mu_{\theta_1}(Z_2) + \sigma_1 \mathbf{n}_1$$

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



$$x = \mu_{\theta_0}(Z_1) + \sigma_0 \mathbf{n}_0$$

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

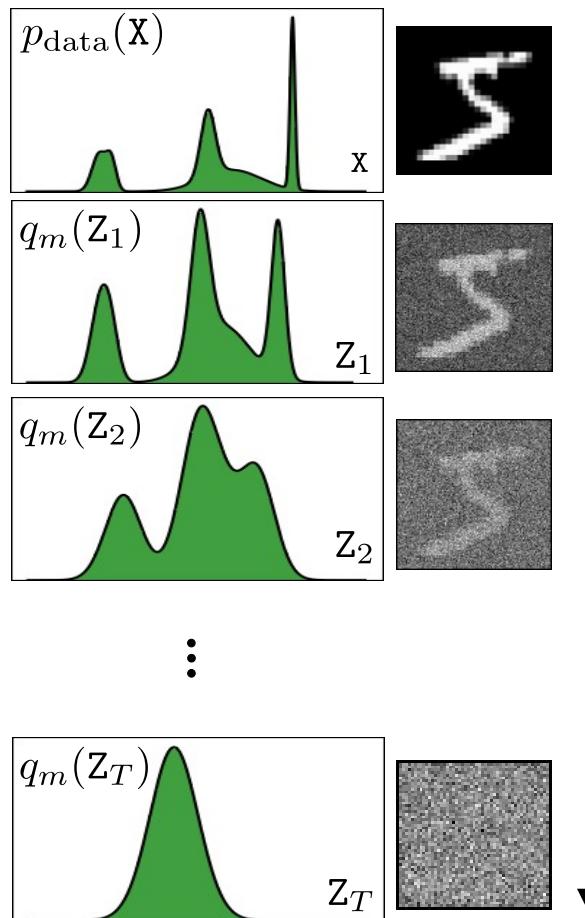
v)

Modèle de diffusion : apprentissage

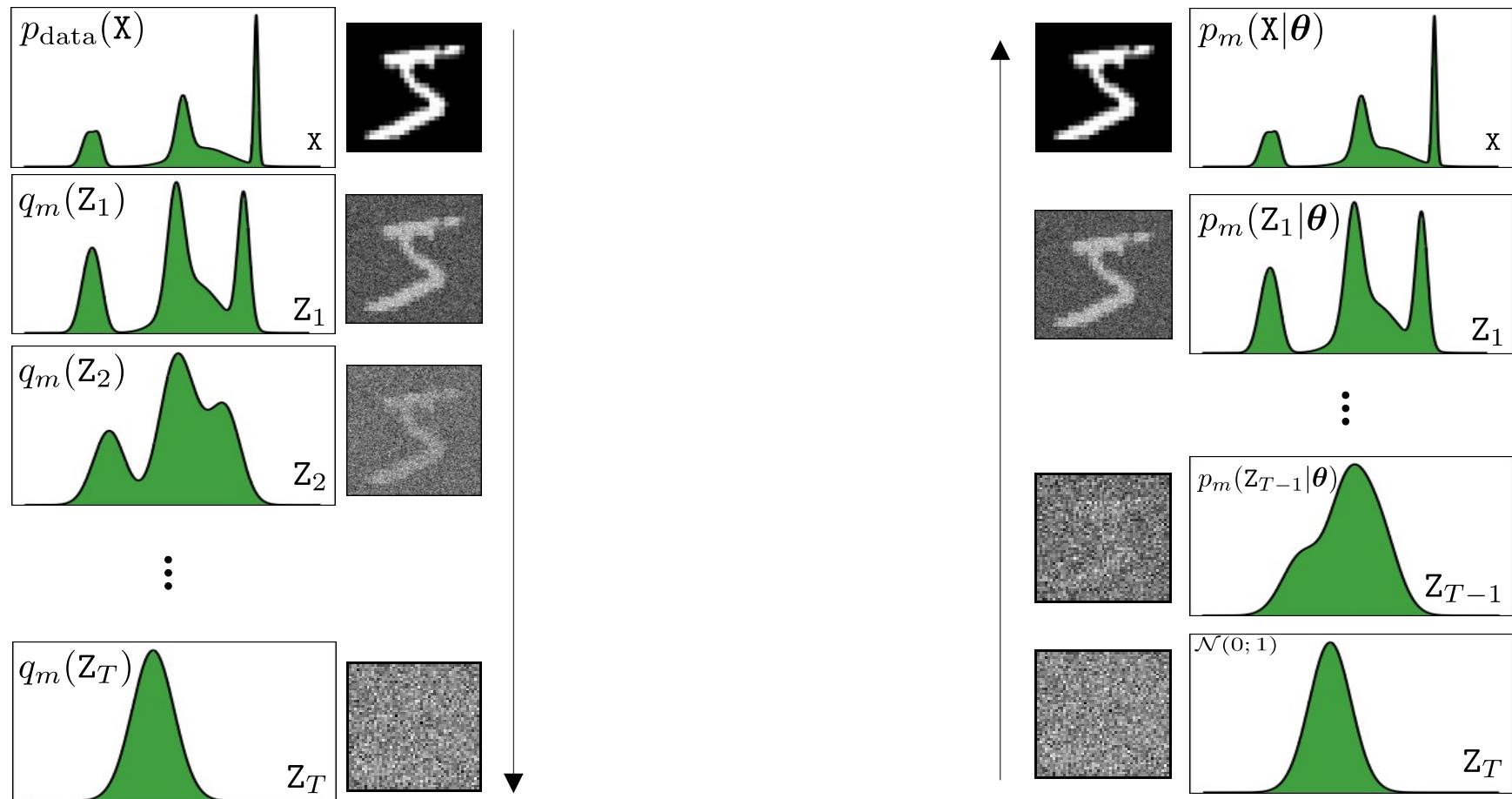
Comme pour le VAE, on ne peut pas calculer $p_{\text{modele}}(\mathbf{X}|\boldsymbol{\theta})$

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

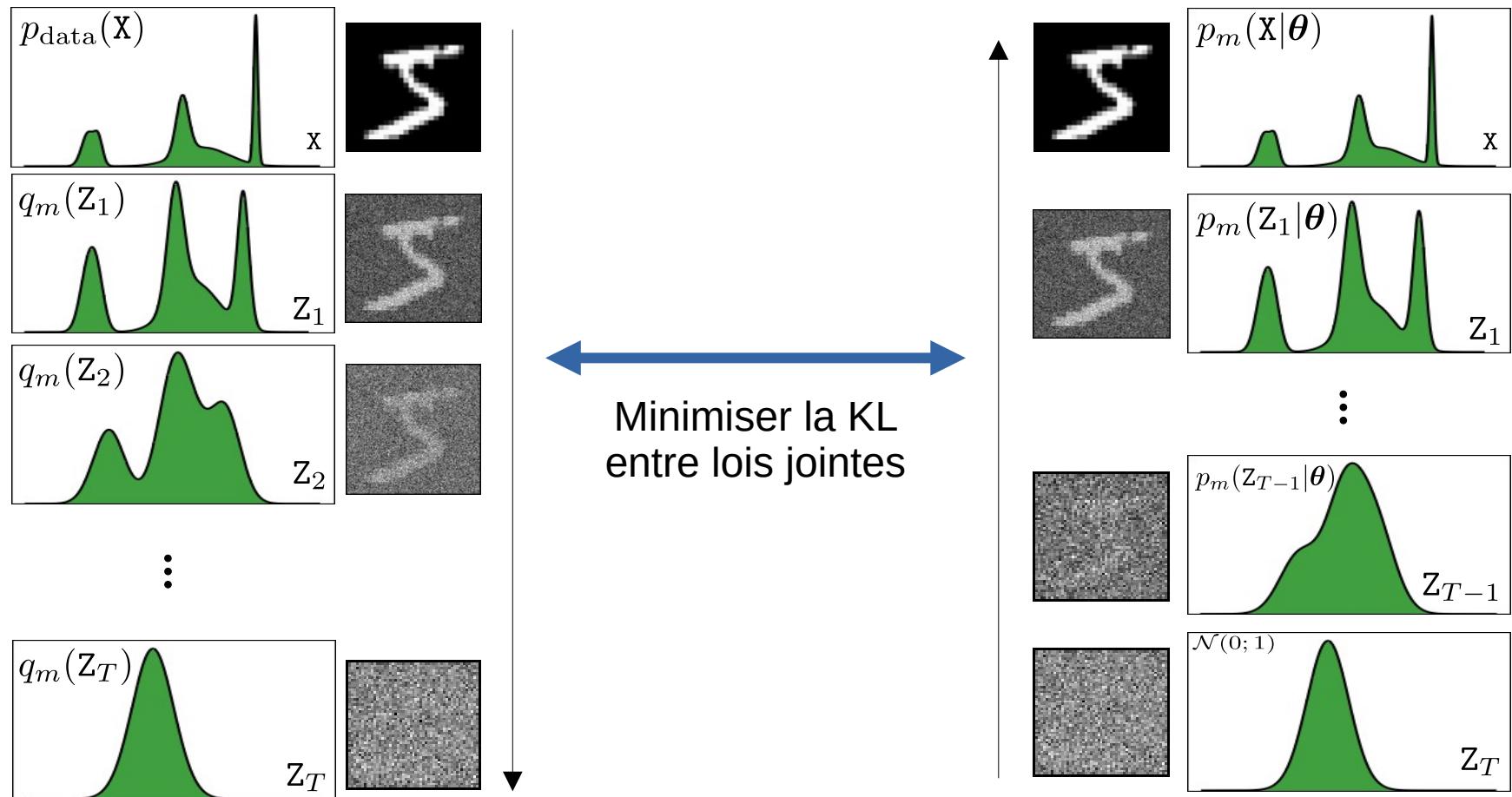

Modèle de diffusion : apprentissage



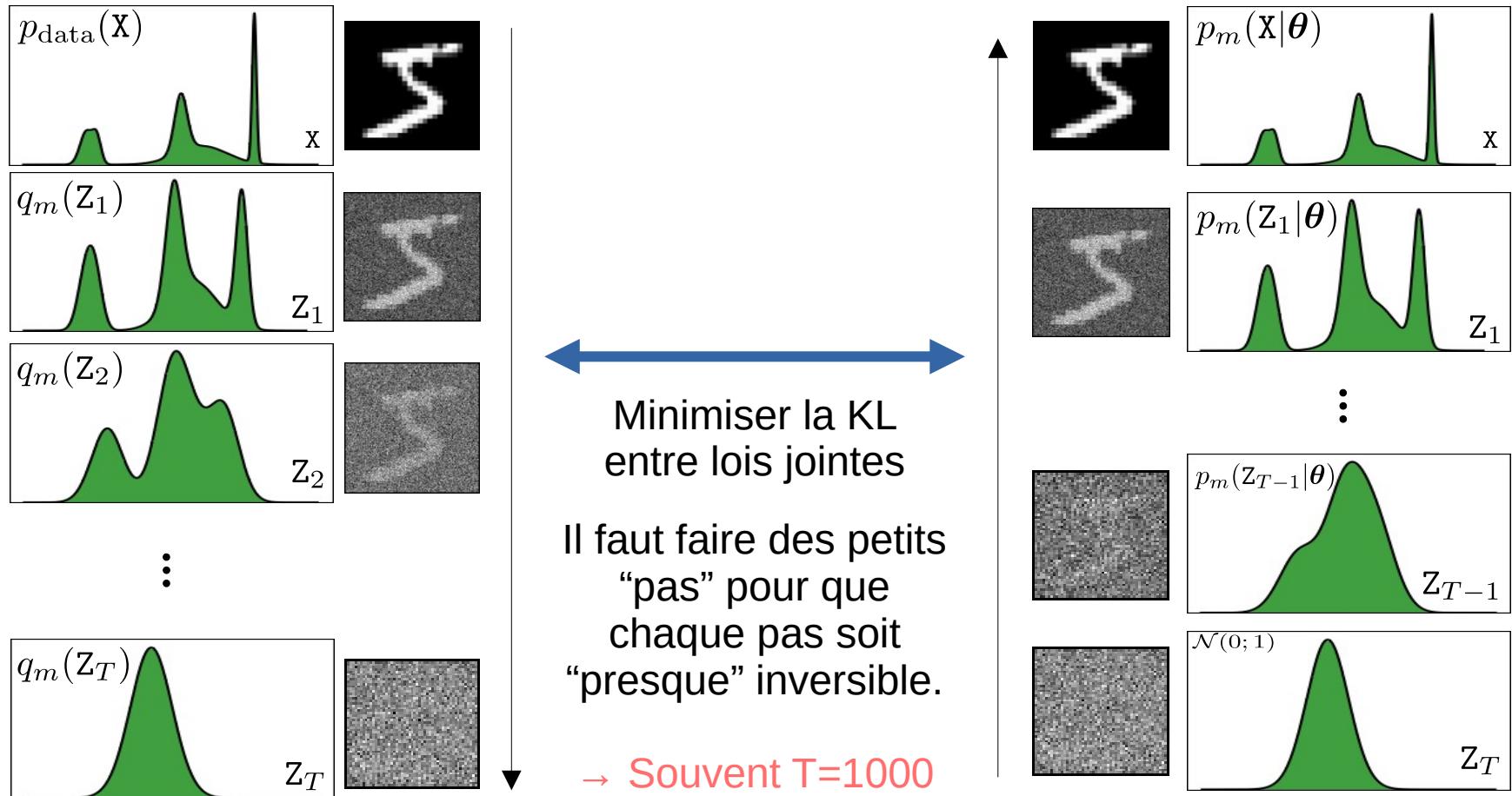
Modèle de diffusion : apprentissage



Modèle de diffusion : apprentissage



Modèle de diffusion : apprentissage



v)

Modèle de diffusion : apprentissage

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

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

↓

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

$$KL(q_m(\mathbf{X}, \mathbf{Z}_1, \dots, \mathbf{Z}_T) || p_m(\mathbf{X}, \mathbf{Z}_1, \dots, \mathbf{Z}_T | \boldsymbol{\theta}))$$

Modèle de diffusion : apprentissage (suite)

Jusqu'à présent on a parlé d'un réseau $\mu_{\theta_{t-1}}(Z_t)$ qui apprend à enlever "un petit peu" de bruit dans l'image Z_t

Modèle de diffusion : apprentissage (suite)

Jusqu'à présent on a parlé d'un réseau $\mu_{\theta_{t-1}}(Z_t)$ qui apprend à enlever "un petit peu" de bruit dans l'image Z_t

Il existe plusieurs variantes

- apprendre à prédire l'image parfaite X
- apprendre à prédire le bruit présent dans l'image (ce qu'on va faire).
- apprendre à prédire le "flow" ("flow matching" très utilisé aujourd'hui),

Modèle de diffusion : apprentissage (suite)

Jusqu'à présent on a parlé d'un réseau $\mu_{\theta_{t-1}}(Z_t)$ qui apprend à enlever "un petit peu" de bruit dans l'image Z_t

Il existe plusieurs variantes

- apprendre à prédire l'image parfaite X
- apprendre à prédire le bruit présent dans l'image (ce qu'on va faire). 
- apprendre à prédire le "flow" ("flow matching" très utilisé aujourd'hui),

Modèle de diffusion : apprentissage (suite)

Jusqu'à présent on a parlé d'un réseau $\mu_{\theta_{t-1}}(Z_t)$ qui apprend à enlever "un petit peu" de bruit dans l'image Z_t

Il existe plusieurs variantes

- apprendre à prédire l'image parfaite X
- apprendre à prédire le bruit présent dans l'image (ce qu'on va faire).
- apprendre à prédire le "flow" ("flow matching" très utilisé aujourd'hui),



Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(Z_t) = \frac{1}{\sqrt{1 - \beta_t}} \left(Z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(Z_t) \right)$$

où

$$\alpha_t = \prod_{k=1}^t (1 - \beta_k)$$

Réseau qui prédit le bruit présent dans l'image

Modèle de diffusion : apprentissage (suite)

Jusqu'à présent on a parlé d'un réseau $\mu_{\theta_{t-1}}(Z_t)$ qui apprend à enlever "un petit peu" de bruit dans l'image Z_t

Il existe plusieurs variantes

- apprendre à prédire l'image parfaite X
 - apprendre à prédire le bruit présent dans l'image (ce qu'on va faire).
 - apprendre à prédire le "flow" ("flow matching" très utilisé aujourd'hui),
- 

Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(Z_t) = \frac{1}{\sqrt{1 - \beta_t}} \left(Z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(Z_t) \right)$$

où

$$\alpha_t = \prod_{k=1}^t (1 - \beta_k)$$

On enlève un peu de ce bruit de Z_t , pour prédire Z_{t-1}^{162}

Modèle de diffusion : apprentissage (suite)

Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(z_t) = \frac{1}{\sqrt{1 - \beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(z_t) \right)$$

Fonction de coût :

$$L_i(\theta) = \sum_{t=1}^T c_t \| \epsilon_{\text{ech},i,t} - \hat{\epsilon}_{\theta_{t-1}}(\sqrt{\alpha_t} \mathbf{x}_i + \sqrt{1 - \alpha_t} \epsilon_{\text{ech},i,t}) \|_2^2$$

$$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; \mathbf{I})$$

Modèle de diffusion : apprentissage (suite)

Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(z_t) = \frac{1}{\sqrt{1 - \beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(z_t) \right)$$

Fonction de coût :

$$L_i(\theta) = \sum_{t=1}^T c_t \| \epsilon_{\text{ech},i,t} - \hat{\epsilon}_{\theta_{t-1}}(\sqrt{\alpha_t} \mathbf{x}_i + \sqrt{1 - \alpha_t} \epsilon_{\text{ech},i,t}) \|_2^2$$

$$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; \mathbb{I})$$

Vocabulaire : “Denoising Score Matching”

v)

Modèle de diffusion : apprentissage (suite)

Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(z_t) = \frac{1}{\sqrt{1 - \beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(z_t) \right)$$

Fonction de coût :

$$L_i(\theta) = \sum_{t=1}^T c_t \| \epsilon_{\text{ech},i,t} - \hat{\epsilon}_{\theta_{t-1}}(\sqrt{\alpha_t} \mathbf{x}_i + \sqrt{1 - \alpha_t} \epsilon_{\text{ech},i,t}) \|_2^2$$

$$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; \mathbf{I})$$

Réseau de neurones qui apprend à prédire le bruit $\epsilon_{\text{ech},i,t}$

Modèle de diffusion : apprentissage (suite)

Paramétrisation de μ_{θ_t} : $\mu_{\theta_{t-1}}(z_t) = \frac{1}{\sqrt{1-\beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(z_t) \right)$

Fonction de coût :

$$L_i(\theta) = \sum_{t=1}^T c_t \| \epsilon_{\text{ech},i,t} - \hat{\epsilon}_{\theta_{t-1}}(\sqrt{\alpha_t} X_i + \sqrt{1-\alpha_t} \epsilon_{\text{ech},i,t}) \|_2^2$$

Un réseau de neurones par pas de temps !

→ T réseaux de neurones à apprendre ...

$$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; I)$$

v)

Modèle de diffusion : apprentissage (suite)

Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(z_t) = \frac{1}{\sqrt{1-\beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(z_t) \right)$$

Fonction de coût :

$$L_i(\theta) = \sum_{t=1}^T c_t \| \epsilon_{\text{ech},i,t} - \hat{\epsilon}_{\theta_{t-1}}(\sqrt{\alpha_t} x_i + \sqrt{1-\alpha_t} \epsilon_{\text{ech},i,t}) \|_2^2$$

$$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; I)$$

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



v)

Modèle de diffusion : apprentissage (suite)

Paramétrisation de μ_{θ_t} :

$$\mu_{\theta_{t-1}}(z_t) = \frac{1}{\sqrt{1-\beta_t}} \left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t}} \hat{\epsilon}_{\theta_{t-1}}(z_t) \right)$$

Fonction de coût :

$$f(\alpha_t, \beta_t)$$

$$L_i(\theta) = \sum_{t=1}^T c_t \| \epsilon_{\text{ech},i,t} - \hat{\epsilon}_{\theta_{t-1}}(\sqrt{\alpha_t} \mathbf{x}_i + \sqrt{1-\alpha_t} \epsilon_{\text{ech},i,t}) \|_2^2$$

$$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; \mathbf{I})$$

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

Modèle de diffusion : apprentissage (suite)

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

$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; \mathbf{I})$



Modèle de diffusion : apprentissage (suite)

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

$\epsilon_{\text{ech},i,t} \sim \mathcal{N}(0; \mathbb{I})$


Apprentissage par descente de gradient stochastique

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

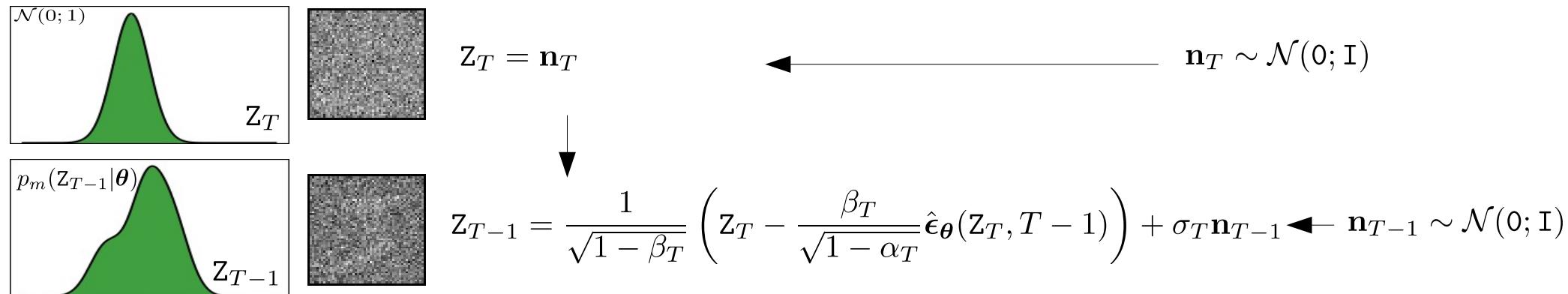
Minibatch sur les données et les pas de temps → méthode “simulation free” ! 171

v)

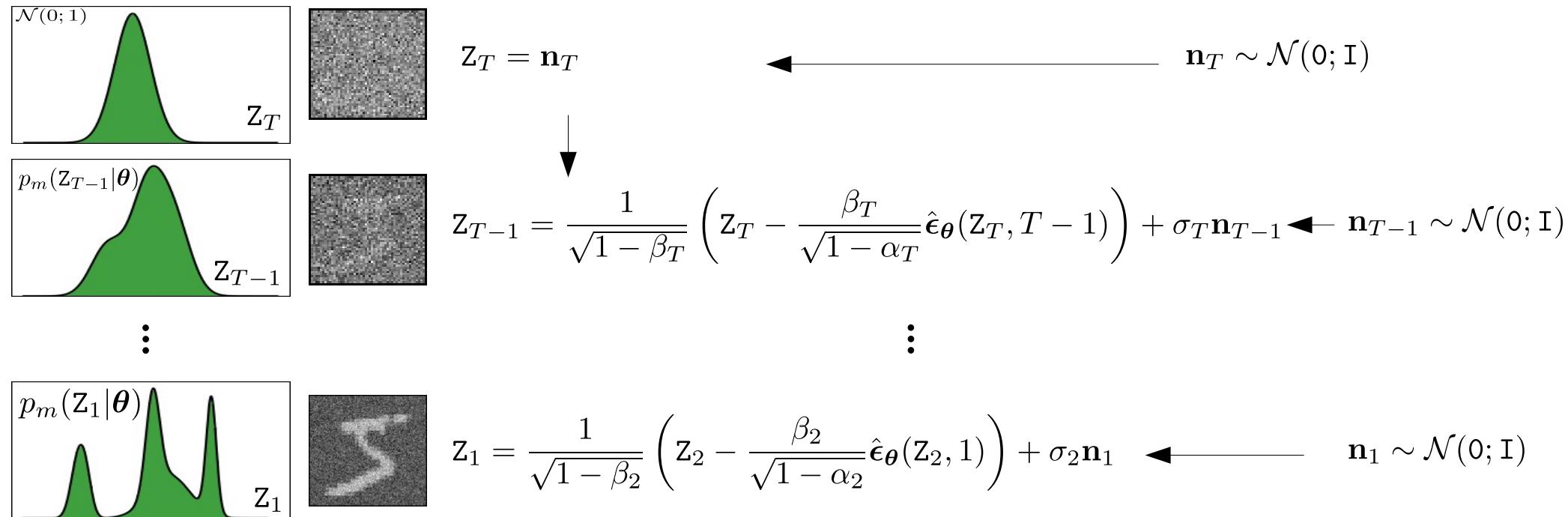
Résumé de l'étape de génération



Résumé de l'étape de génération

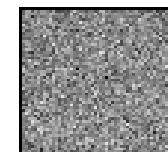
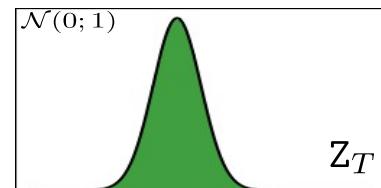


Résumé de l'étape de génération



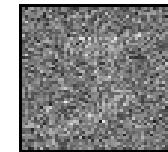
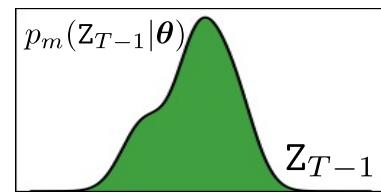
v)

Résumé de l'étape de génération



$$Z_T = \mathbf{n}_T$$

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

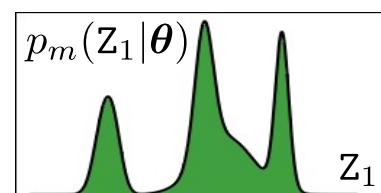


$$Z_{T-1} = \frac{1}{\sqrt{1 - \beta_T}} \left(Z_T - \frac{\beta_T}{\sqrt{1 - \alpha_T}} \hat{\epsilon}_{\theta}(Z_T, T-1) \right) + \sigma_T \mathbf{n}_{T-1}$$

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

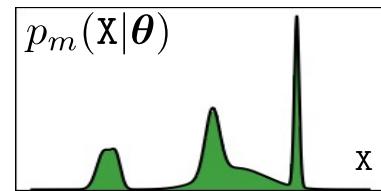
⋮

⋮



$$Z_1 = \frac{1}{\sqrt{1 - \beta_2}} \left(Z_2 - \frac{\beta_2}{\sqrt{1 - \alpha_2}} \hat{\epsilon}_{\theta}(Z_2, 1) \right) + \sigma_2 \mathbf{n}_1$$

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



$$x = \frac{1}{\sqrt{1 - \beta_1}} \left(Z_1 - \frac{\beta_1}{\sqrt{1 - \alpha_1}} \hat{\epsilon}_{\theta}(Z_1, 0) \right) + \sigma_1 \mathbf{n}_0$$

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

Modèle de diffusion : 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)

Modèle de diffusion : 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)

Modèle de diffusion : améliorations

Axe 1 : Réduire le nombre de pas de temps
→ accélérer la génération.

Modèle de diffusion : améliorations

Axe 1 : Réduire le nombre de pas de temps
→ accélérer la génération.

Axe 2 : Diffuser dans un espace latent de plus petite taille
→ réduire l'empreinte mémoire et accélérer la génération

Modèle de diffusion : améliorations

Axe 1 : Réduire le nombre de pas de temps
→ accélérer la génération.

Axe 2 : Diffuser dans un espace latent de plus petite taille
→ réduire l'empreinte mémoire et accélérer la génération

Différents formalismes :

- diffusion à temps discret (DDPM)
- diffusion à temps continu, équation différentielle stochastique (SDE)
- “Flow matching”

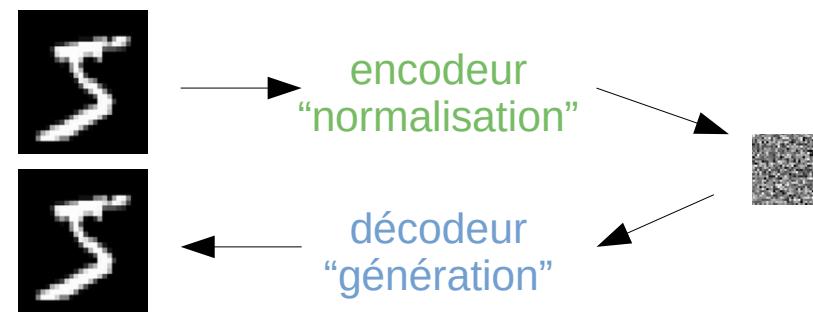
“Latent diffusion” : Diffusion dans un espace latent

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

“Latent diffusion” : Diffusion dans un espace latent

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

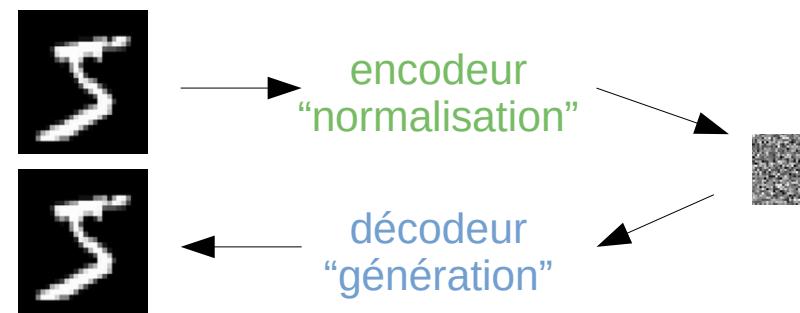
- 1) entraîner un VAE pour encoder les images dans un espace latent de plus faible dimension



“Latent diffusion” : Diffusion dans un espace latent

“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

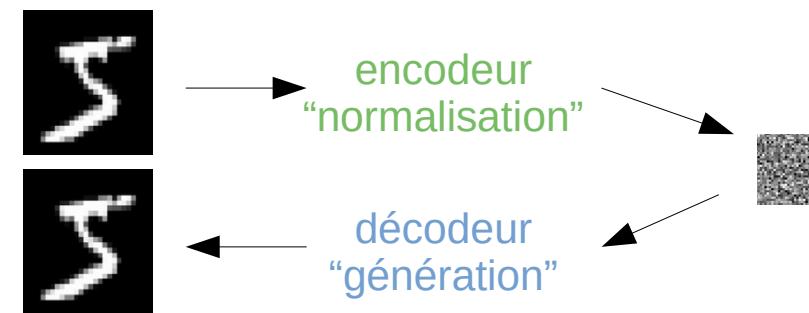


En pratique, on veut surtout que la génération se passe bien, pas que l'espace latent soit gaussien, donc on ne fait pas un vrai VAE.

“Latent diffusion” : Diffusion dans un espace latent

“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 modèle de diffusion sur cet espace latent



“Latent diffusion” : Diffusion dans un espace latent

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



“Latent diffusion” : Diffusion dans un espace latent

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



Autre perspective : diffusion sur un espace discret

“Text diffusion”

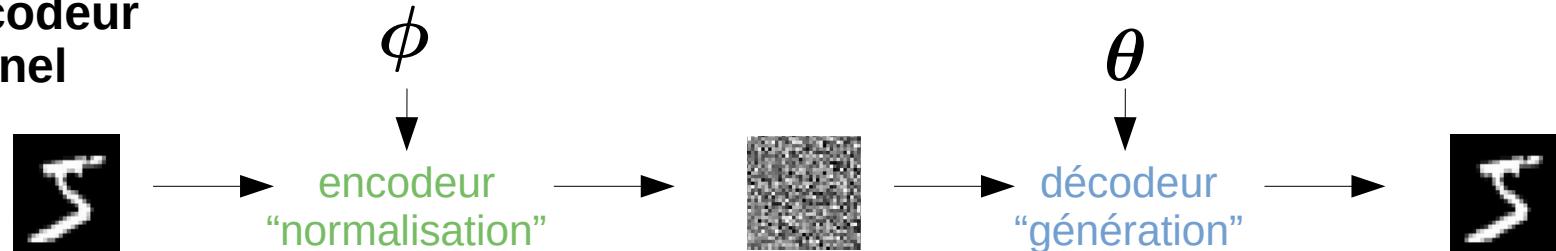
Following their victory in the French and Indian War, Britain began to assert greater _____



VI) Réseau antagoniste (GAN)

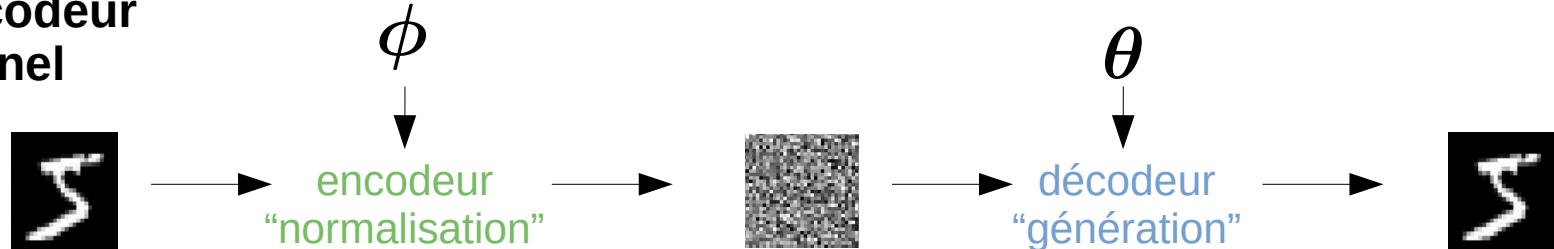
Réseau antagoniste : idée générale

**Auto-encodeur
variationnel**

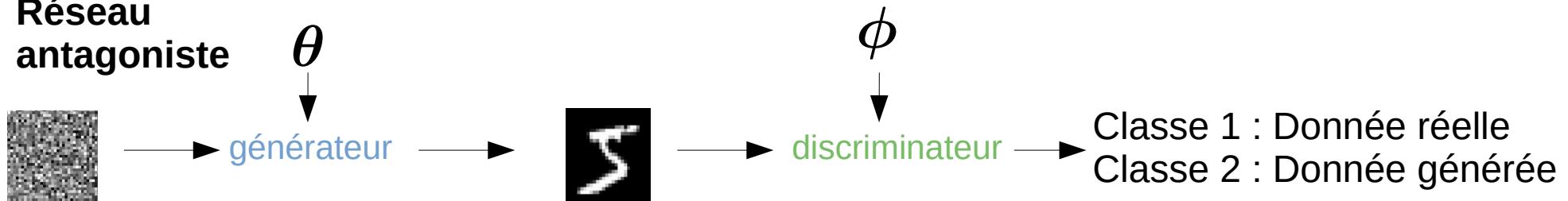


Réseau antagoniste : idée générale

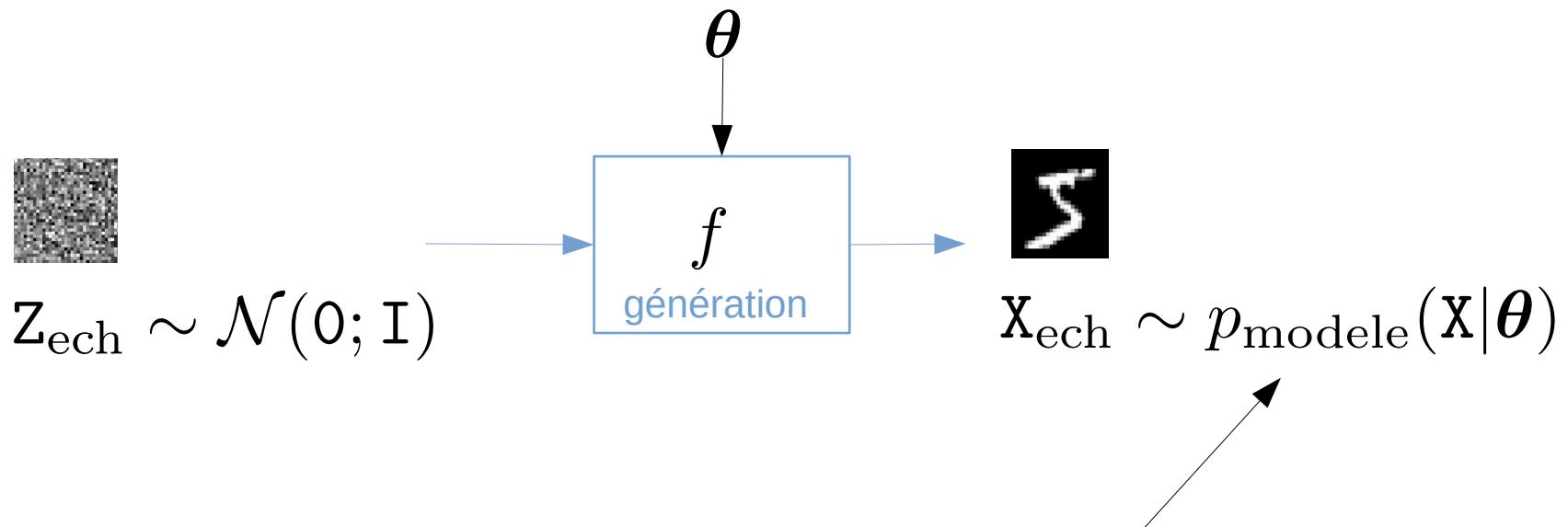
Auto-encodeur variationnel



Réseau antagoniste

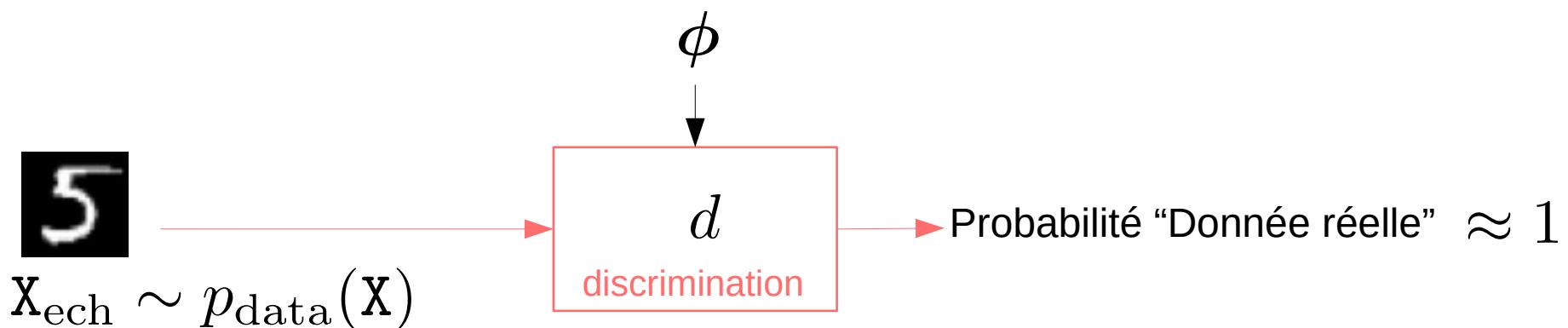
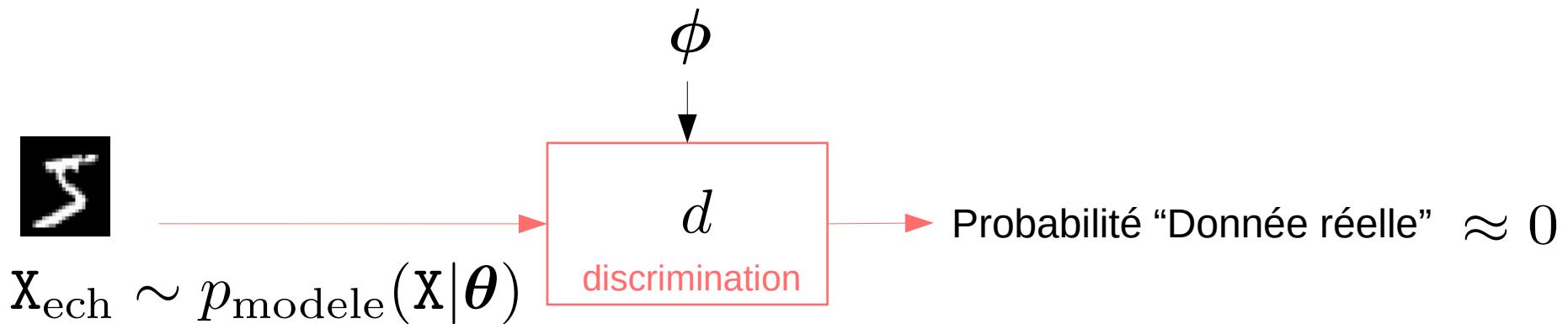


Réseau antagoniste : générateur

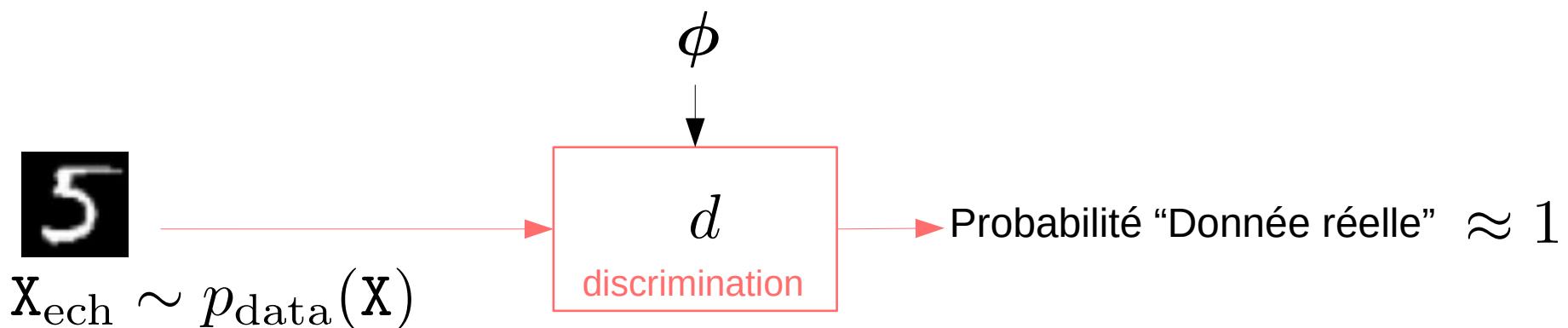
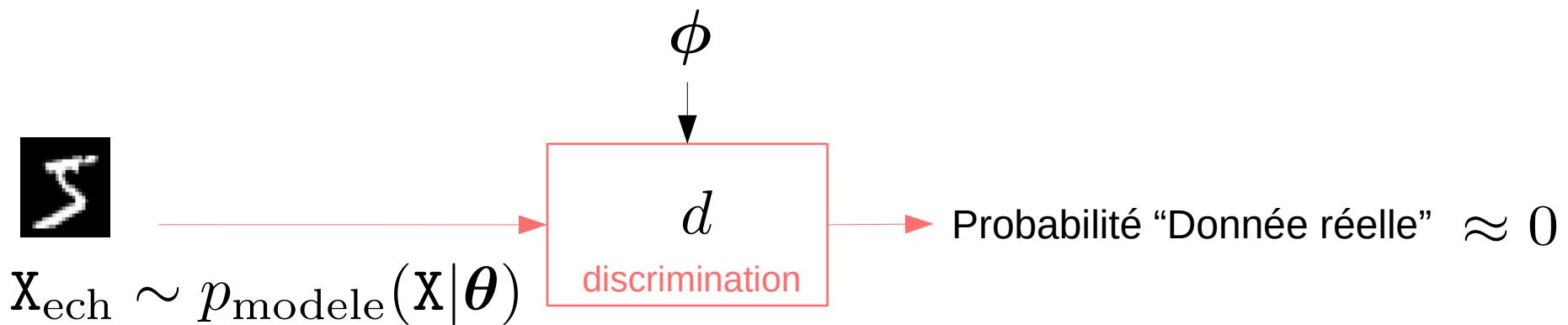


Définition implicite, le générateur n'est pas inversible

Réseau antagoniste : discriminateur

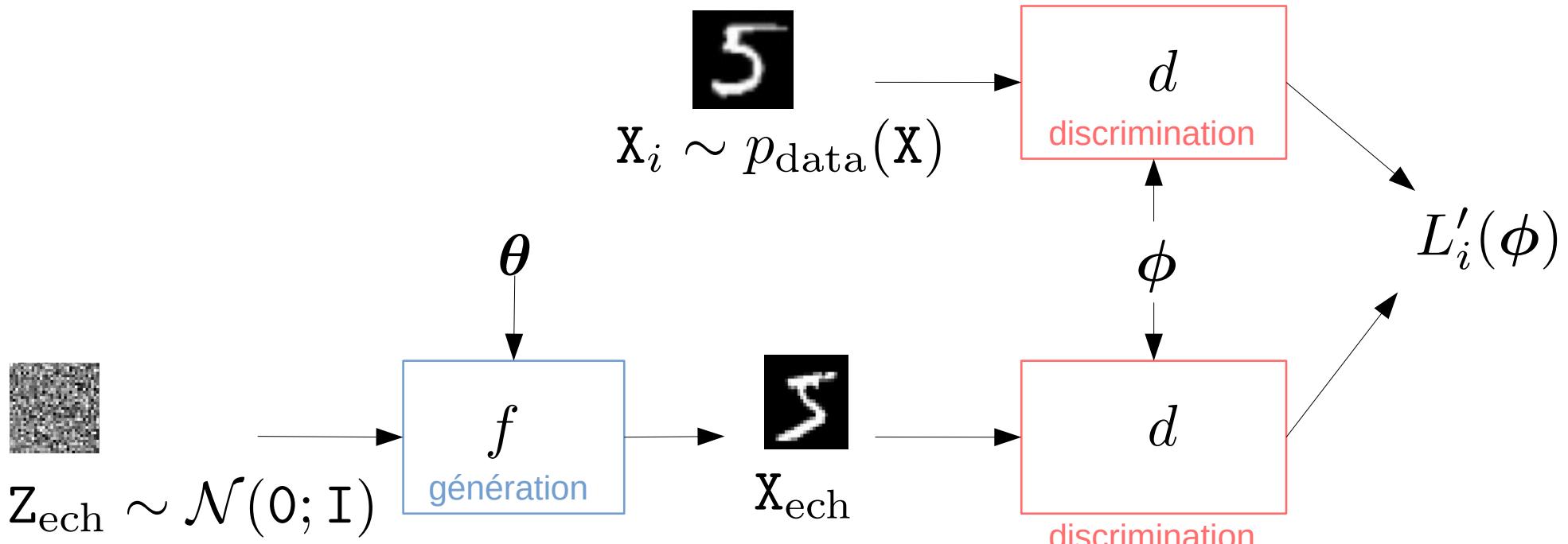


Réseau antagoniste : discriminateur



Réseau antagoniste : apprentissage

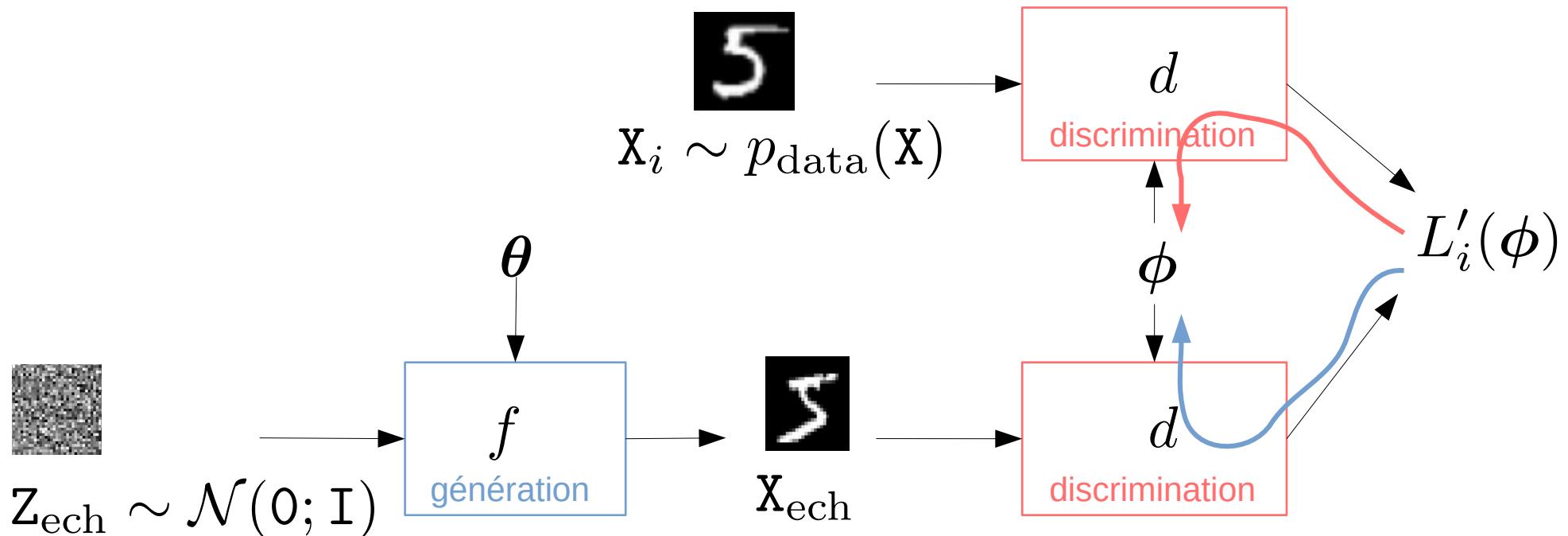
Pas de gradient sur les paramètres du discriminateur



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

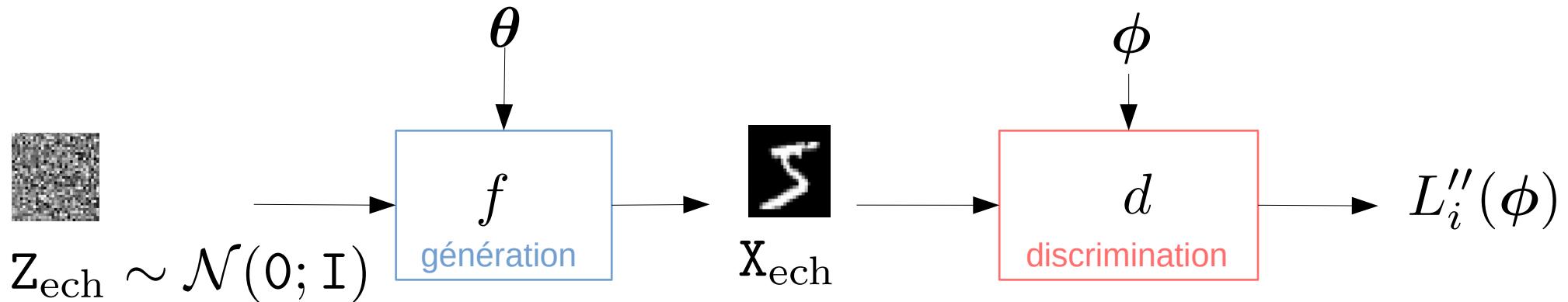
Réseau antagoniste : apprentissage

Pas de gradient sur les paramètres du discriminateur



Réseau antagoniste : apprentissage (suite)

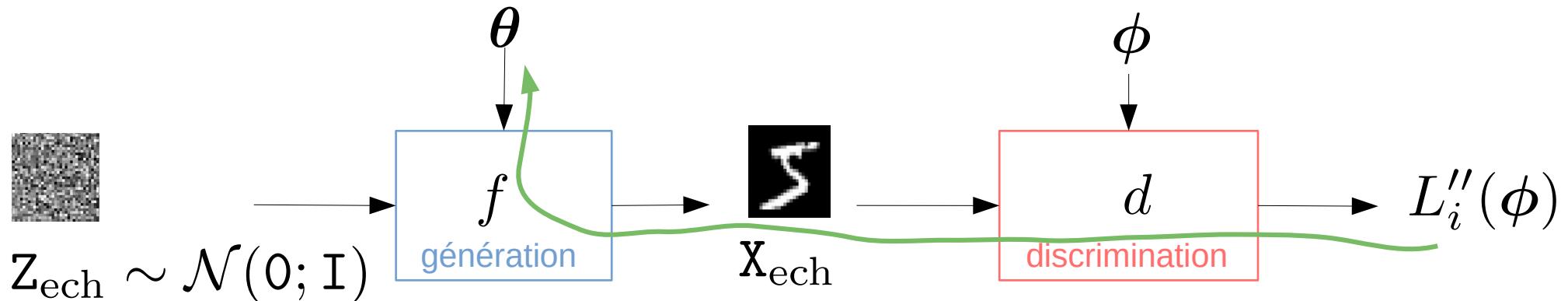
Pas de gradient sur les paramètres du générateur



Le générateur s'améliore pour mieux tromper le discriminateur.

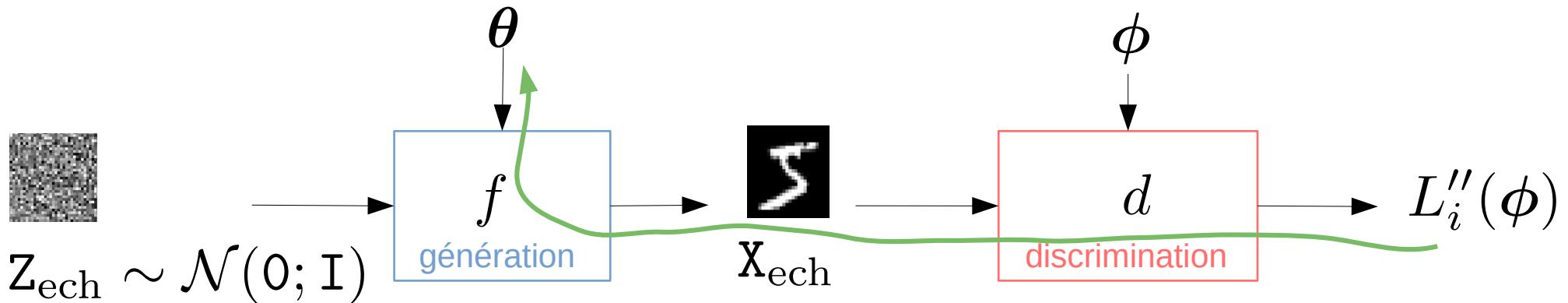
Réseau antagoniste : apprentissage (suite)

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 θ pour que $p_{\text{modele}}(\mathbf{x}|\theta) \approx p_{\text{data}}(\mathbf{x})$

Réseau antagoniste : apprentissage (formalisé)

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

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

$$\begin{aligned} D(p_m(\mathbf{x}|\boldsymbol{\theta}) || p_{\text{data}}(\mathbf{x})) \\ \geq \max_{d(.)} \mathbb{E}_{p_m(\mathbf{x}|\boldsymbol{\theta})}(\ln(1 - d(\mathbf{x})) + \mathbb{E}_{p_{\text{data}}(\mathbf{x})}(\ln(d(\mathbf{x}))) \end{aligned}$$


discriminateur

Réseau antagoniste : apprentissage (formalisé)

Objectif : optimiser θ pour que $p_{\text{modele}}(\mathbf{x}|\theta) \approx p_{\text{data}}(\mathbf{x})$

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

$$D(p_m(\mathbf{x}|\theta) || p_{\text{data}}(\mathbf{x}))$$

$$\geq \max_{d(\cdot)} \mathbb{E}_{p_m(\mathbf{x}|\theta)}(\ln(1 - d(\mathbf{x})) + \mathbb{E}_{p_{\text{data}}(\mathbf{x})}(\ln(d(\mathbf{x})))$$

discriminateur

Uniquement besoin de savoir échantillonner selon $p_m(\mathbf{x}|\theta)$

Réseau antagoniste : apprentissage (formalisé)

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

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

$$\begin{aligned} D(p_m(\mathbf{x}|\boldsymbol{\theta}) || p_{\text{data}}(\mathbf{x})) \\ \geq \max_{d(.)} \mathbb{E}_{p_m(\mathbf{x}|\boldsymbol{\theta})}(\ln(1 - d(\mathbf{x})) + \mathbb{E}_{p_{\text{data}}(\mathbf{x})}(\ln(d(\mathbf{x}))) \end{aligned}$$

Astuce de reparamétrisation (“Reparameterization trick”)

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

Réseau antagoniste : apprentissage (formalisé)

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

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

$$\begin{aligned} D(p_m(\mathbf{x}|\boldsymbol{\theta}) || p_{\text{data}}(\mathbf{x})) \\ \geq \max_{d(.)} \mathbb{E}_{p_m(\mathbf{x}|\boldsymbol{\theta})}(\ln(1 - d(\mathbf{x})) + \mathbb{E}_{p_{\text{data}}(\mathbf{x})}(\ln(d(\mathbf{x}))) \end{aligned}$$

Astuce de reparamétrisation (“Reparameterization trick”)

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

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

Réseau antagoniste : problème d'apprentissage

$$\min_{\theta} \max_{\phi} \mathbb{E}_{\mathcal{N}(0, I)}(\ln(1 - d(f(z, \theta), \phi))) + \mathbb{E}_{p_{\text{data}}(x)}(\ln(d(x, \phi)))$$

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 ϕ

$$L'_i(\phi) = -\ln(1 - d(f(z_{\text{ech}}, \theta), \phi)) - \ln(d(x_i, \phi))$$

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 ϕ

$$L'_i(\phi) = -\ln(1 - d(f(z_{\text{ech}}, \theta), \phi)) - \ln(d(x_i, \phi))$$

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 θ

$$L''_i(\theta) = \ln(1 - d(f(z_{\text{ech}}, \theta), \phi)) \rightarrow$$

En pratique remplacé par :

$$-\ln(d(f(z_{\text{ech}}, \theta), \phi))$$

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")

