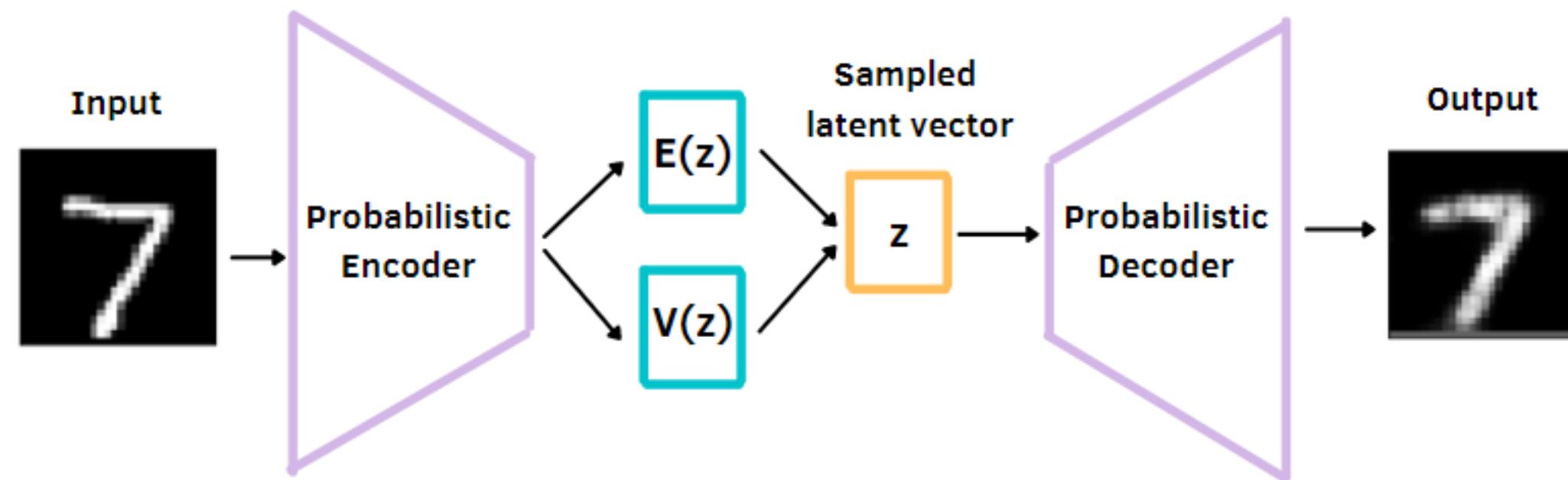


VARIATIONAL AUTOENCODER



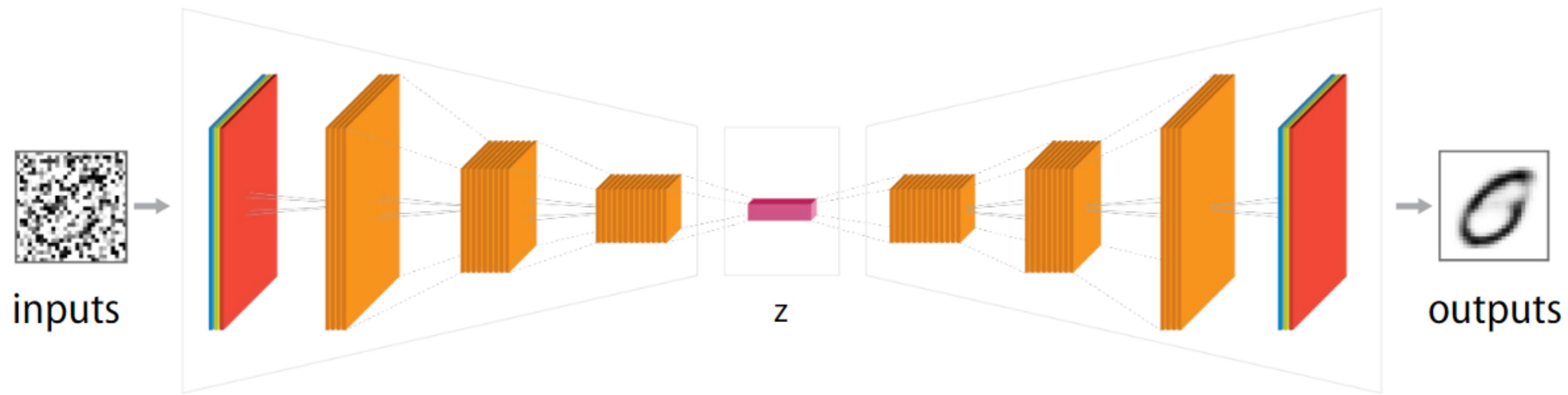
Diederik P. Kingma, Max Welling (2013), Universiteit Van Amsterdam,
“Auto-Encoding Variational Bayes” : <https://arxiv.org/pdf/1312.6114.pdf>

SOMMAIRE

- 1** Problématiques et motivations
- 2** Optimisation
- 3** Application au VAE
- 4** Application avec MNIST
- 5** Application avec FREYFACE
- 6** Conclusion
- 7** L'évolution depuis la publication du papier

1 : PROBLÉMATIQUES ET MOTIVATIONS

1.1 : RAPPELS SUR L'AUTOENCODEUR



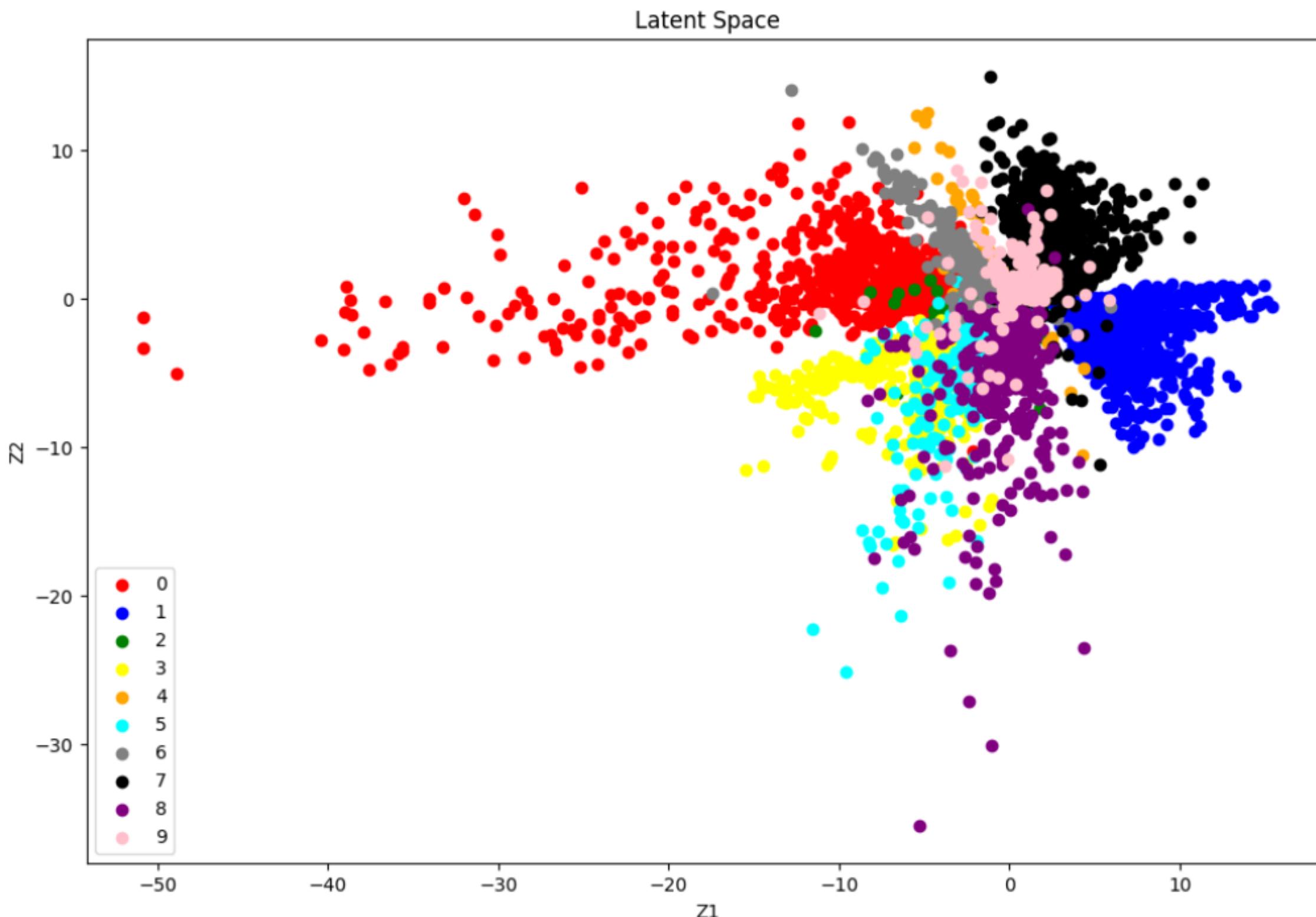
Réseau de neurones découpé en trois parties :

- L'encodeur
- La représentation latente
- Le décodeur

L'autoencodeur permet de capturer
des caractéristiques et structures
non linéaires des données

1 : PROBLÉMATIQUES ET MOTIVATIONS

1.1 : RAPPELS SUR L'AUTOENCODEUR



- Des clusters sont naturellement établis, l'autoencodeur à de bonnes capacités à regrouper les images avec des caractéristiques communes
- On distingue néanmoins quelques limites...

1 : PROBLÉMATIQUES ET MOTIVATIONS

1.2 : PRINCIPE GÉNÉRAL DE L'AUTOENCODEUR VARIATIONNEL

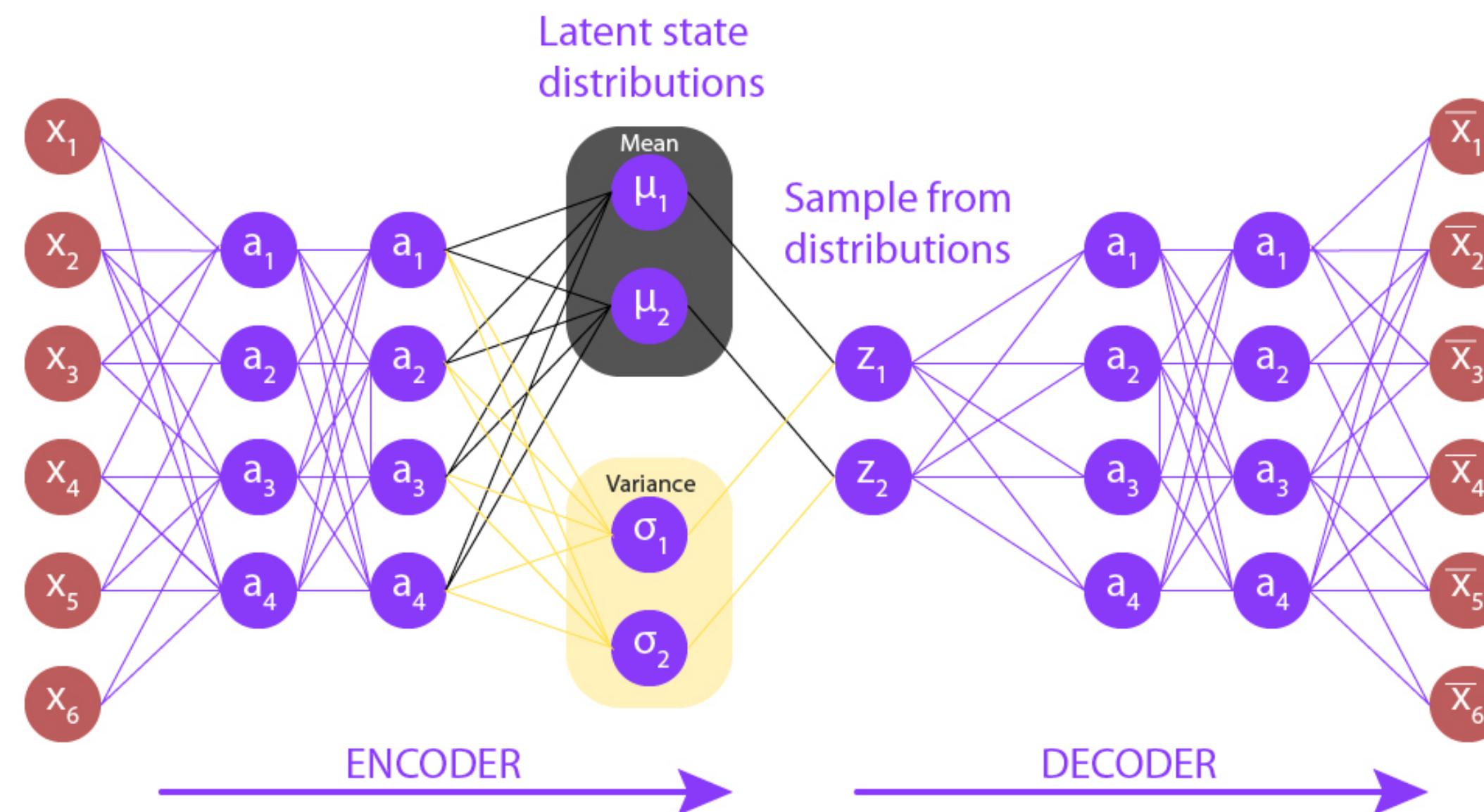
Comment remédier à ce problème ?

1 : PROBLÉMATIQUES ET MOTIVATIONS

1.2 : PRINCIPE GÉNÉRAL DE L'AUTOENCODEUR VARIATIONNEL

Comment remédier à ce problème ?

- Idée de l'autoencodeur variationnel : Contrôler cette dispersion



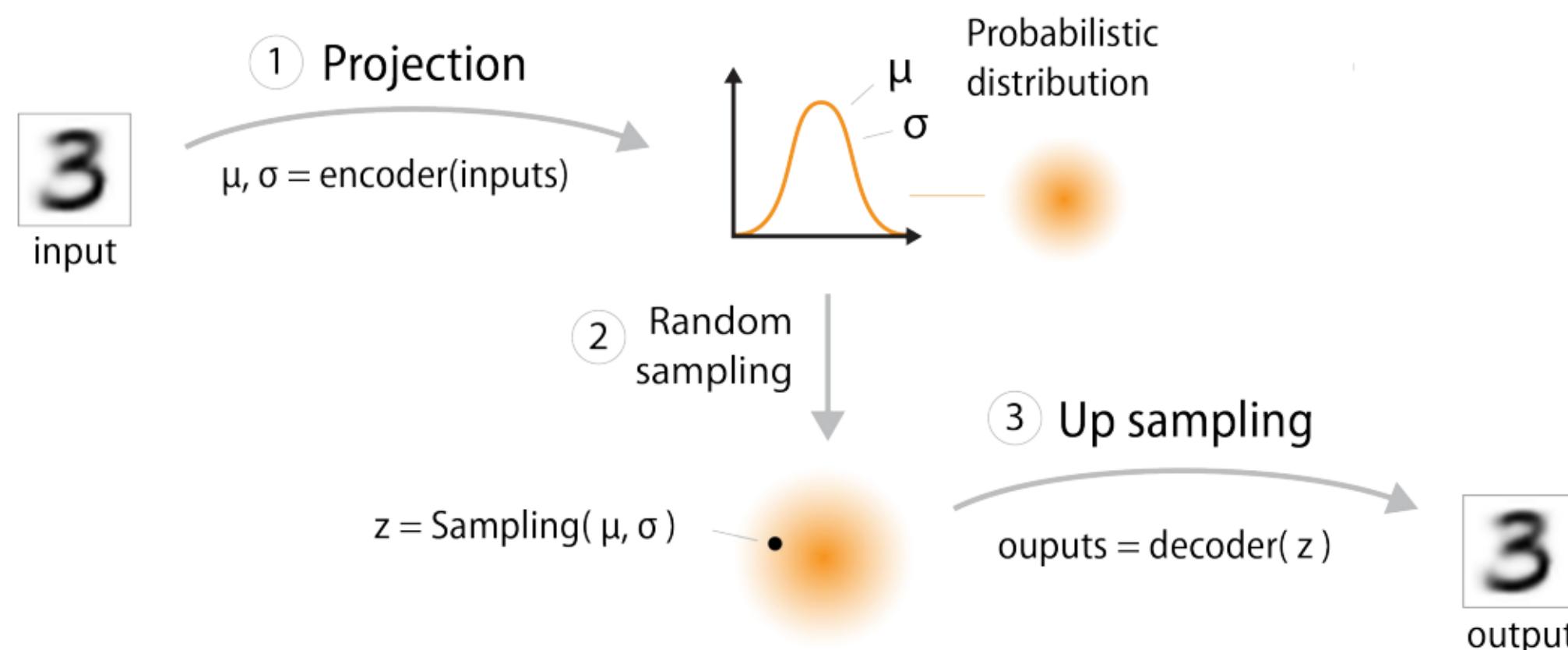
1 : PROBLÉMATIQUES ET MOTIVATIONS

1.2 : PRINCIPE GÉNÉRAL DE L'AUTOENCODEUR VARIATIONNEL

Comment remédier à ce problème ?

- Idée de l'autoencodeur variationnel : Contrôler cette dispersion

μ is a mean
 σ is a variance
 z vector in latent space



1 : PROBLÉMATIQUES ET MOTIVATIONS

1.2 : PRINCIPE GÉNÉRAL DE L'AUTOENCODEUR VARIATIONNEL

Comment remédier à ce problème ?

- Idée de l'autoencodeur variationnel : Contrôler cette dispersion

Il y a donc 2 aspects fondamentaux dans cette restructuration de l'autoencodeur :

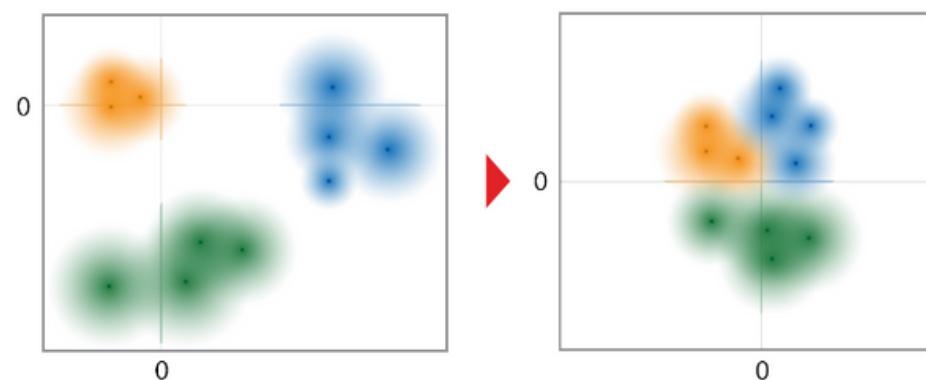
- On contrôle la distribution, en ne mettant dans notre espace latent que des points piochés dans une **distribution imposée**
- On échantillonne : donc une **variabilité stochastique** apparaît ce qui permet de générer de nouvelles données

1 : PROBLÉMATIQUES ET MOTIVATIONS

1.3 : PARLONS D'INFÉRENCE VARIATIONNELLE

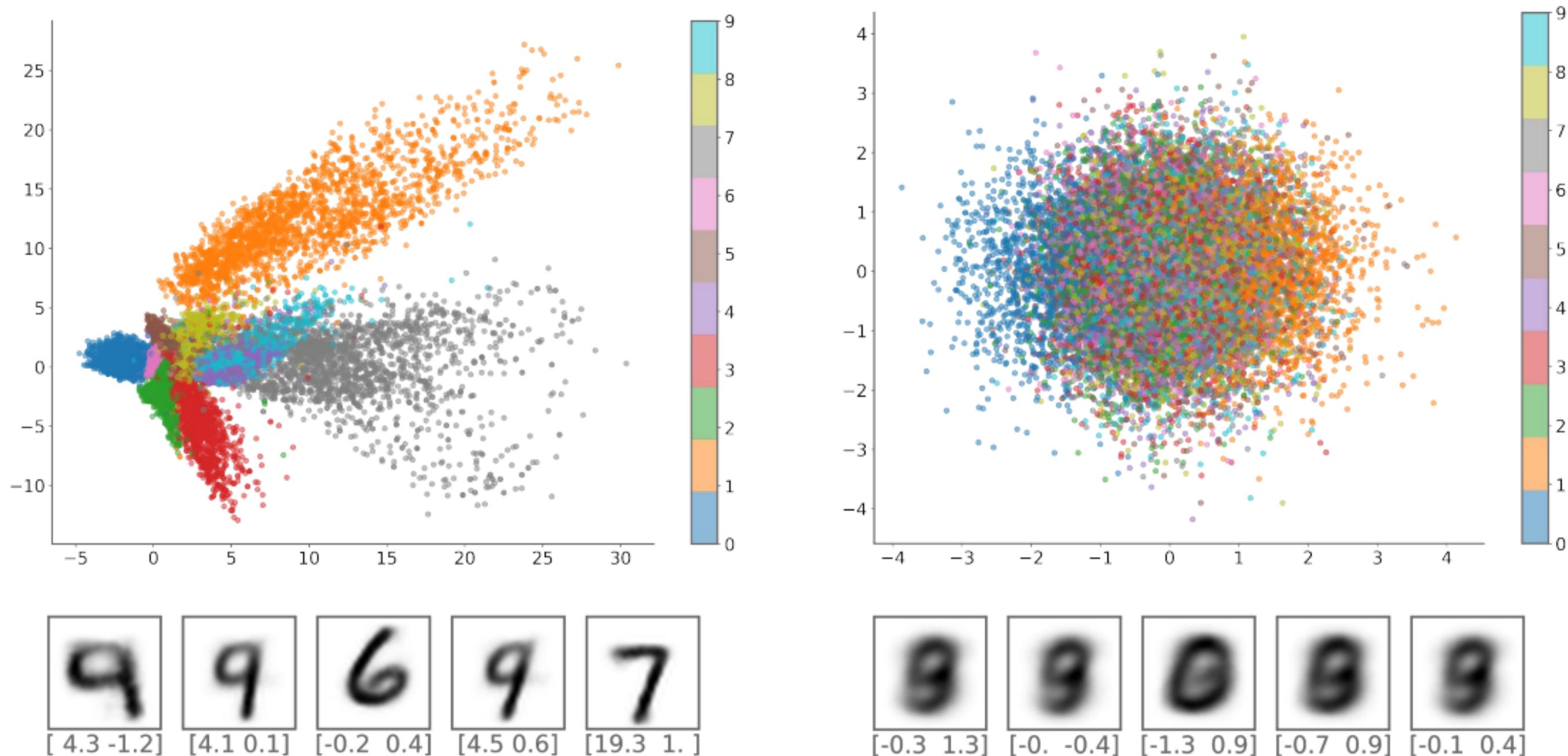
Les objectifs sont désormais différents :

- On ne veut plus simplement minimiser **la différence entre nos données de sortie et d'entrée** (reconstruction des données)
- Mais on veut aussi mesurer la capacité du réseau à générer un espace latent dans lequel **la distribution de probabilité a posteriori correspond à la distribution de probabilité imposée** (Minimisation de la divergence Kullback-Leibler)



1 : PROBLÉMATIQUES ET MOTIVATIONS

1.3 : PARLONS D'INFÉRENCE VARIATIONNELLE



2 : OPTIMISATION

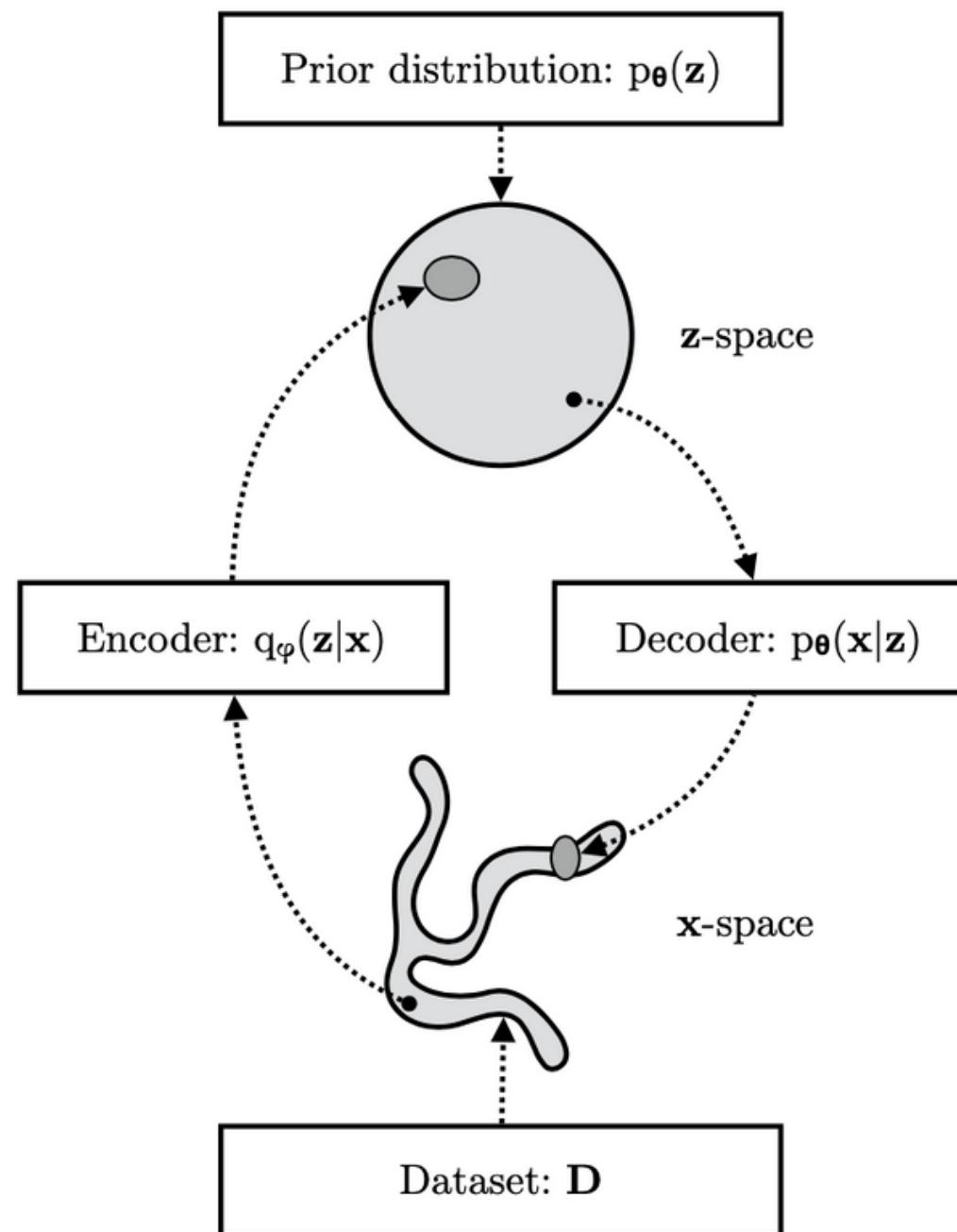
2.1 : BASE DE CALCUL

- On part d'une observation $(x^{(i)})_{i=1}^N$ d'une variable aléatoire x
- On suppose que ces points sont générés par un processus aléatoire invoquant une autre variable aléatoire z **inconnue**
- Ce processus est fait en 2 étapes :
 - une valeur $z^{(i)}$ est générée par une distribution $p_{\theta^*}(z)$
 - une valeur $x^{(i)}$ est générée par une distribution $p_{\theta^*}(x|z)$
- On va supposer que les distributions $p_{\theta^*}(z)$ et $p_{\theta^*}(x|z)$ viennent d'une famille de distribution paramétré par θ (et différentiable en tout point)

Objectif : les paramètres θ^* , on cherche à maximiser $\log(p_\theta(x)) = \sum_{i=1}^N \log(p_\theta(x^{(i)}))$

2 : OPTIMISATION

2.1 : BASE DE CALCUL



$$\text{On a } p_\theta(x) = \int p_\theta(x, z) dz = \int p_\theta(z) p_\theta(x|z) dz$$

Or le calcul est bien trop coûteux, voir irréalisable

Or d'après la formule de Bayes, on peut écrire :

$$p_\theta(x) = \frac{p_\theta(z) p_\theta(x|z)}{p_\theta(z|x)}$$

Si on a une approximation de $p_\theta(z|x)$ on peut

alors calculer $p_\theta(x)$

- On introduit alors la distribution $q_\phi(z|x)$, paramétré par ϕ et dont le but est d'approcher $p_\theta(z|x)$
- L'idée : faire comme si les variables $x^{(i)}$ avaient servies à encoder les variables $z^{(i)}$

2 : OPTIMISATION

2.1 : BASE DE CALCUL

$$\begin{aligned}\log(p_\theta(x)) &= \mathbb{E}_{q_\phi(z|x)}(\log(p_\theta(x))) = \int \log(p_\theta(x)) q_\phi(z|x) dz \\ &= \mathbb{E}_{q_\phi(z|x)}\left(\log \frac{p_\theta(x,z)}{p_\theta(z|x)}\right) && \text{règle de la chaîne} \\ &= \mathbb{E}_{q_\phi(z|x)}\left(\log \frac{p_\theta(x,z)}{q_\phi(z|x)} \frac{q_\phi(z|x)}{p_\theta(z|x)}\right) \\ &= \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)]}_{ELBO = \mathcal{L}_{\theta,\phi}(x)} + \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]}_{= D_{KL}(q_\phi(z|x) \| p_\theta(z|x))}\end{aligned}$$

On peut donc écrire $\mathcal{L}_{\theta,\phi}(x) = \log p_\theta(x) - D_{KL}[q_\phi(z|x) \| p_\theta(z|x)] \leq \log p_\theta(x)$

2 : OPTIMISATION

2.2 : CALCUL DES GRADIENTS

Calcul du gradient en θ :

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{\theta, \phi}(x) &= \nabla_{\theta} E_{q_{\phi}(z|x)}[\log p_{\theta}(x, z) - \log q_{\phi}(z|x)] \\ &= E_{q_{\phi}(z|x)}[\nabla_{\theta}(\log p_{\theta}(x, z) - \log q_{\phi}(z|x))] \\ &\approx \nabla_{\theta}(\log p_{\theta}(x, z) - \log q_{\phi}(z|x)) \quad <- \text{Estimateur non biaisé !} \\ &= \nabla_{\theta} \log p_{\theta}(x, z)\end{aligned}$$

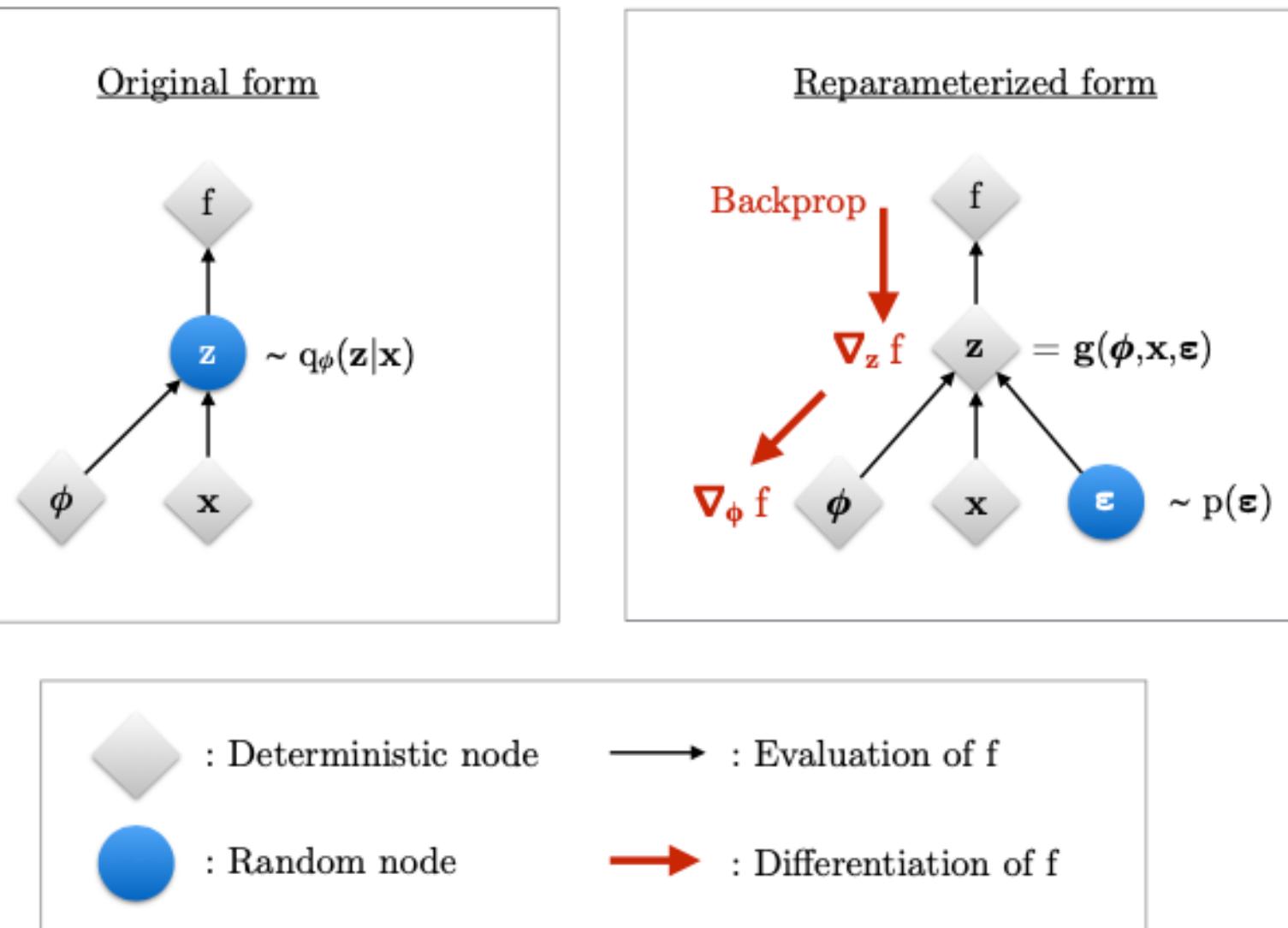
Par contre pour le gradient en ϕ on a

$$\nabla_{\phi} E_{q_{\phi}(z|x)}[\log p_{\theta}(x, z) - \log q_{\phi}(z|x)] \neq E_{q_{\phi}(z|x)}[\nabla_{\phi}(\log p_{\theta}(x, z) - \log q_{\phi}(z|x))]$$

Il faut trouver une autre méthode pour calculer ce gradient

2 : OPTIMISATION

2.3 : REPARAMETERIZATION TRICK



- On exprime la variable aléatoire z comme une transformation d'une variable aléatoire ϵ suivant $p(\epsilon)$ pour z et ϕ donné : $z = g(\epsilon, \phi, z)$
- On a alors $q_\phi(z|x)dz = p(\epsilon)d\epsilon$
- Et donc on peut écrire $E_{q_\phi(z|x)}[f(z)] = E_{p(\epsilon)}[f(z)]$

- Exemple :
- $$q_\phi(z|x) \sim N(z; \mu, \sigma)$$

$$p(\epsilon) \sim N(\epsilon; 1, 0)$$
$$z = \mu + \sigma \odot \epsilon$$

Diederik P. Kingma and Max Welling (2019), “An Introduction to Variational Autoencoders”, Foundations and Trends® in Machine Learning

2 : OPTIMISATION

2.3 : REPARAMETRIZATION TRICK

Finalement, en récrivant l'ELBO de la manière suivante :

$$\mathcal{L}_{\theta, \phi} = E_{p(\epsilon)}[\log p_\theta(x, z) - \log q_\phi(z|x)]$$

On obtient l'estimateur non biaisé $\tilde{\mathcal{L}}_{\theta, \phi}(x)$ pour un point de donné x qui est calculé par :

$$\epsilon \sim p(\epsilon)$$

$$z = g(\epsilon, \phi, x)$$

$$\tilde{\mathcal{L}}_{\theta, \phi}(x) = \log p_\theta(x, z) - \log q_\phi(z|x)$$

2 : OPTMISATION

2.4 : PSEUDO-ALGORITHME D'OPTMISATION

Data:

\mathcal{D} : Dataset

$q_\phi(\mathbf{z}|\mathbf{x})$: Inference model

$p_\theta(\mathbf{x}, \mathbf{z})$: Generative model

Result:

θ, ϕ : Learned parameters

$(\theta, \phi) \leftarrow$ Initialize parameters

while *SGD not converged* **do**

$\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

$\epsilon \sim p(\epsilon)$ (Random noise for every datapoint in \mathcal{M})

 Compute $\tilde{\mathcal{L}}_{\theta, \phi}(\mathcal{M}, \epsilon)$ and its gradients $\nabla_{\theta, \phi} \tilde{\mathcal{L}}_{\theta, \phi}(\mathcal{M}, \epsilon)$

 Update θ and ϕ using SGD optimizer

end

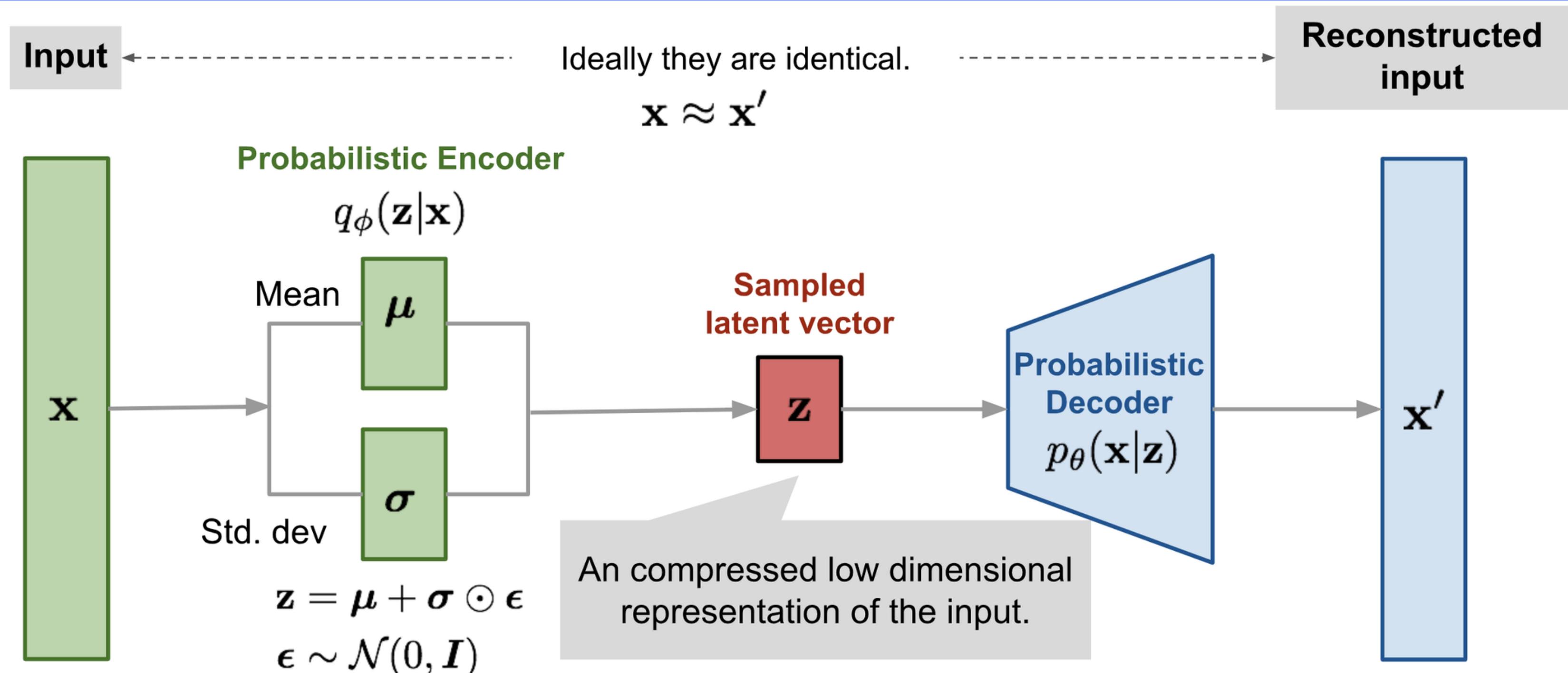
2 : OPTMISATION

2.5 : CONCLUSION

- L'introduction d'une distribution pour simuler l'encodage de l'espace latent par les variables oberservés
- L'utilisation de la fonction de coût ELBO et du DKL
- le trick de la reparamétrisation pour calculer le gradient

fin des maths ouf...

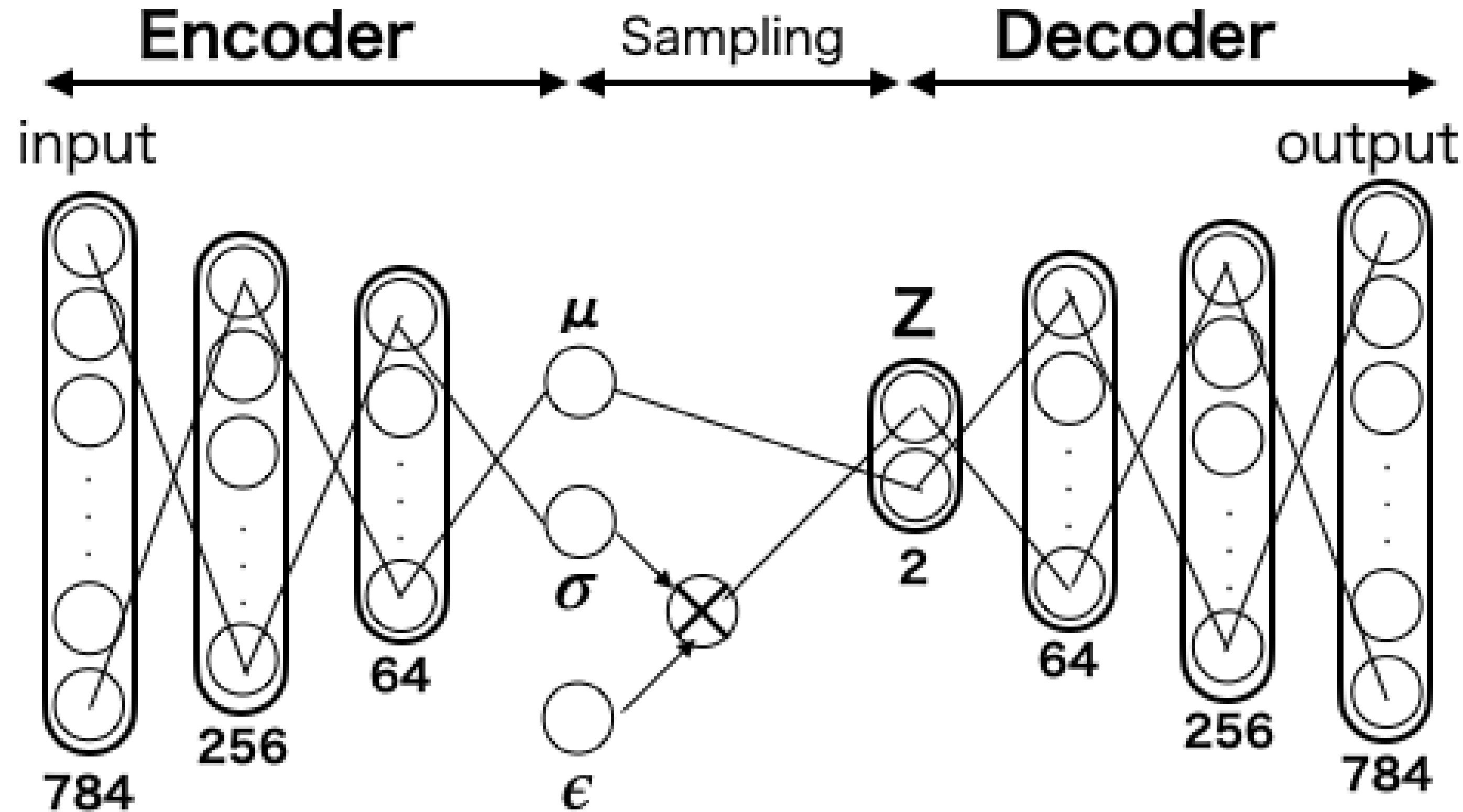
3. APPLICATION À UN VAE



All you need to know about Variational AutoEncoder

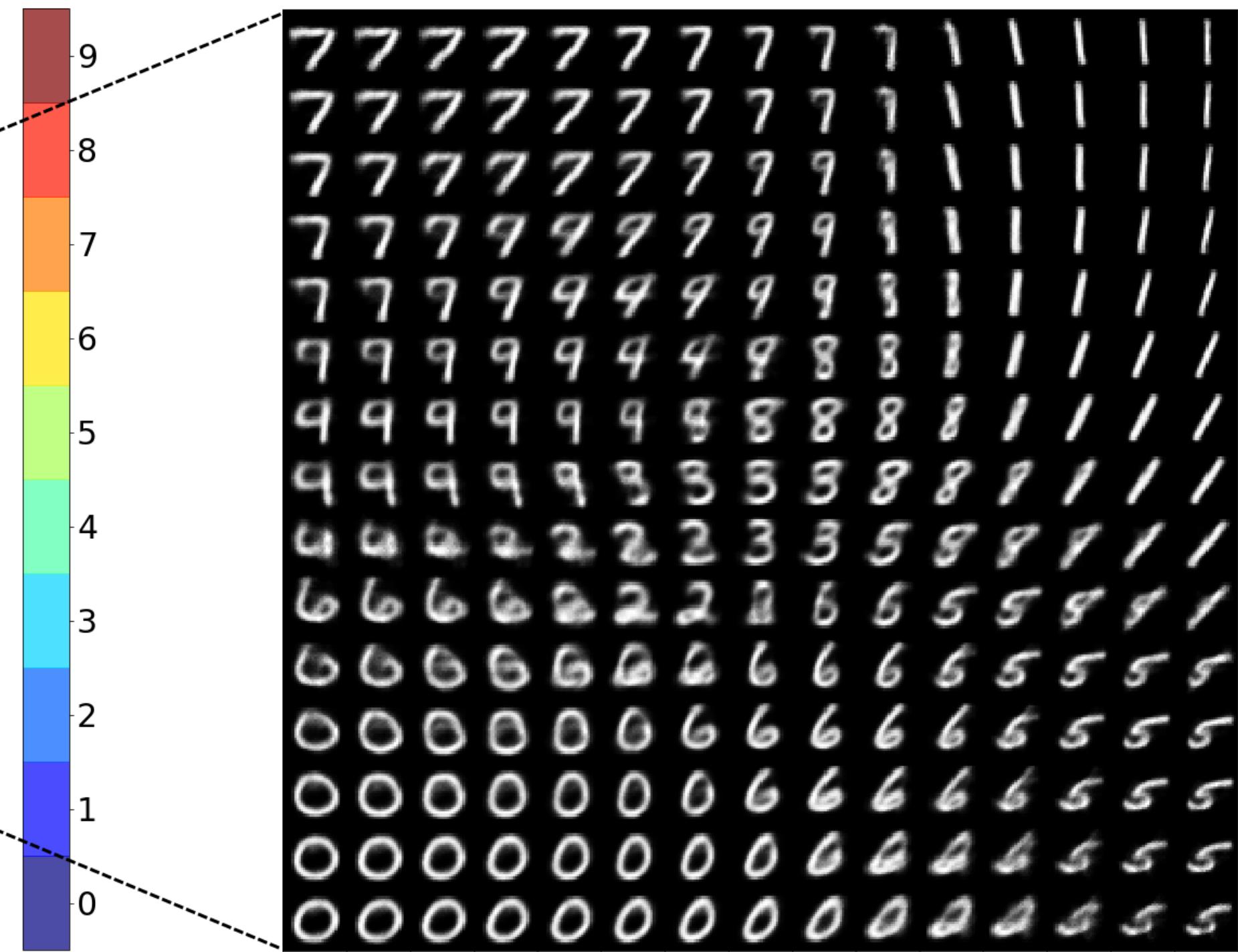
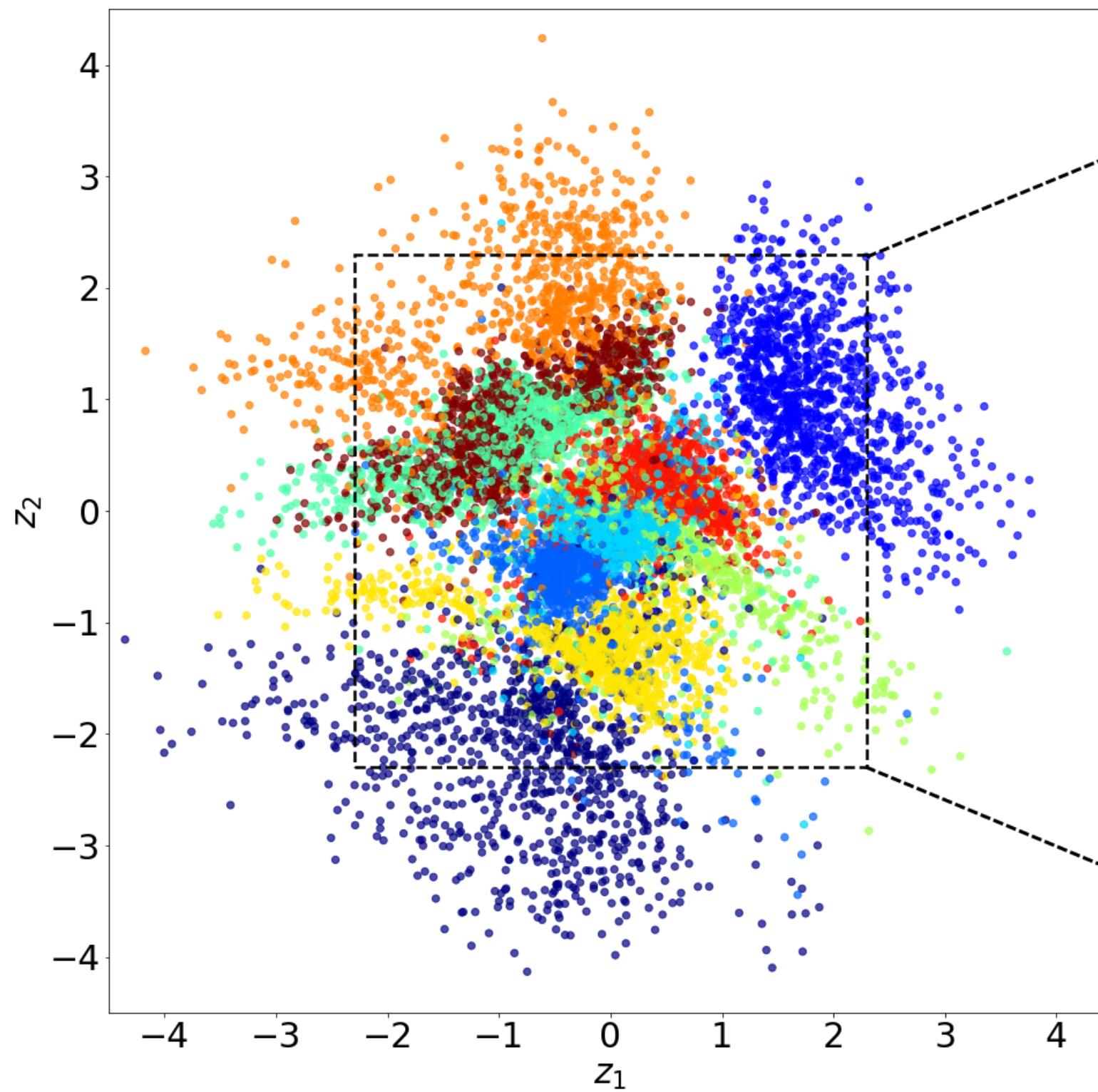
Variational AutoEncoders are being widely used in generative models since the day they came into existence. This blog is just a synopsis of understanding...

3. APPLICATION À UN VAE



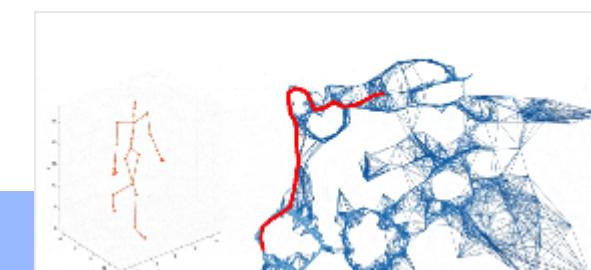
now about Variational

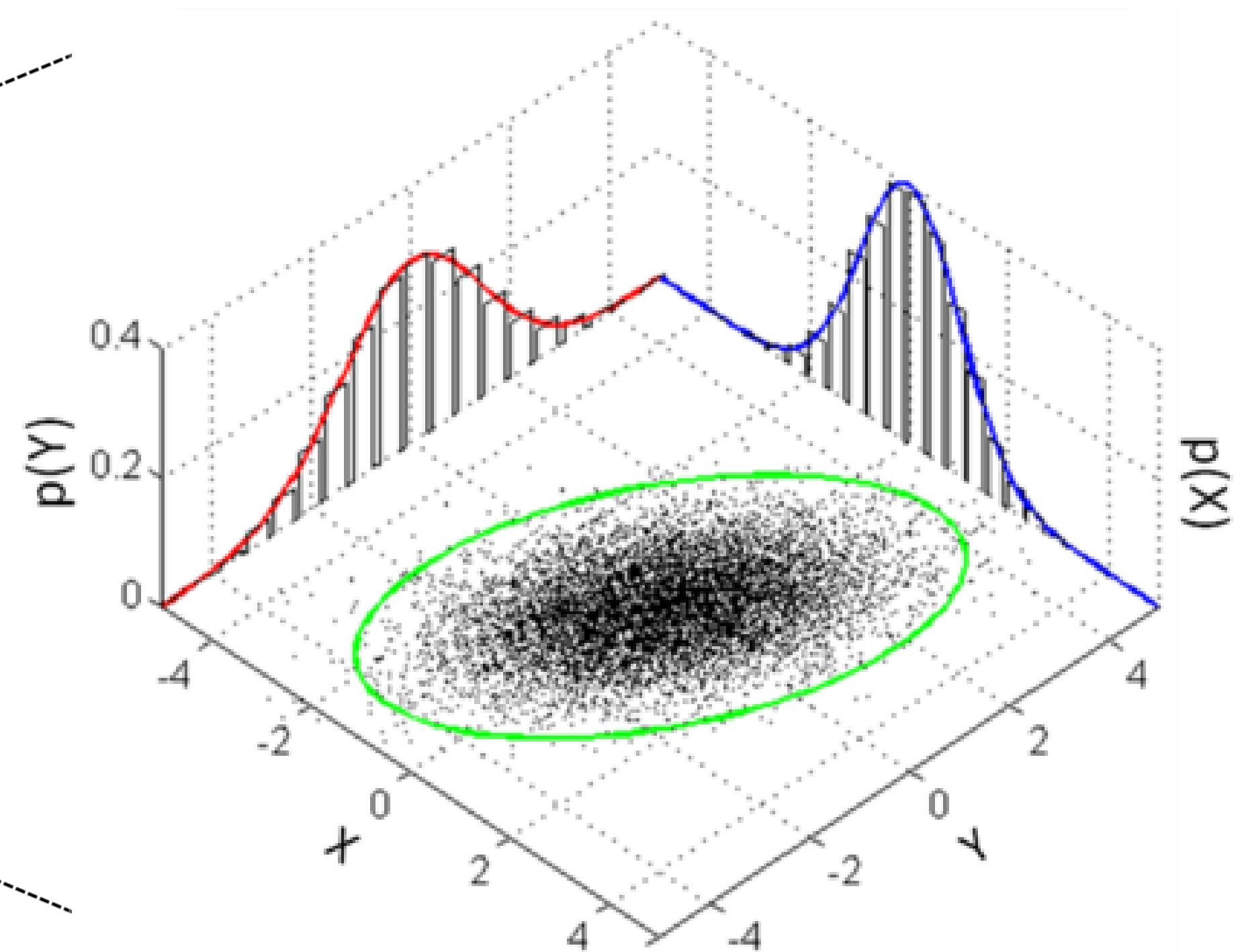
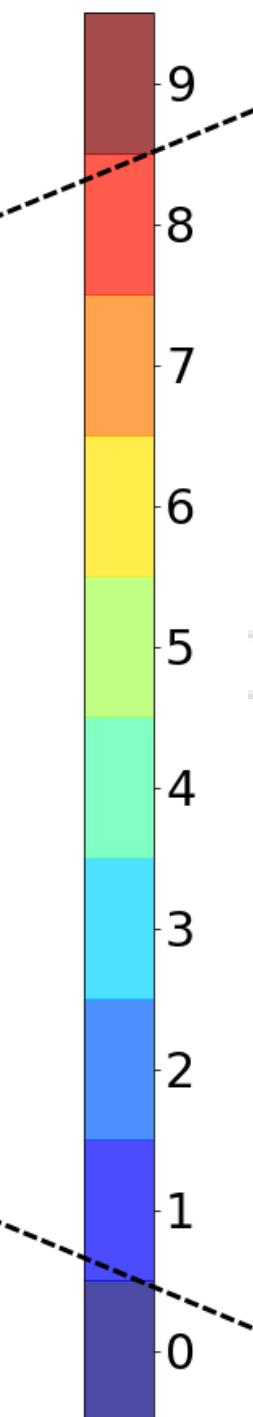
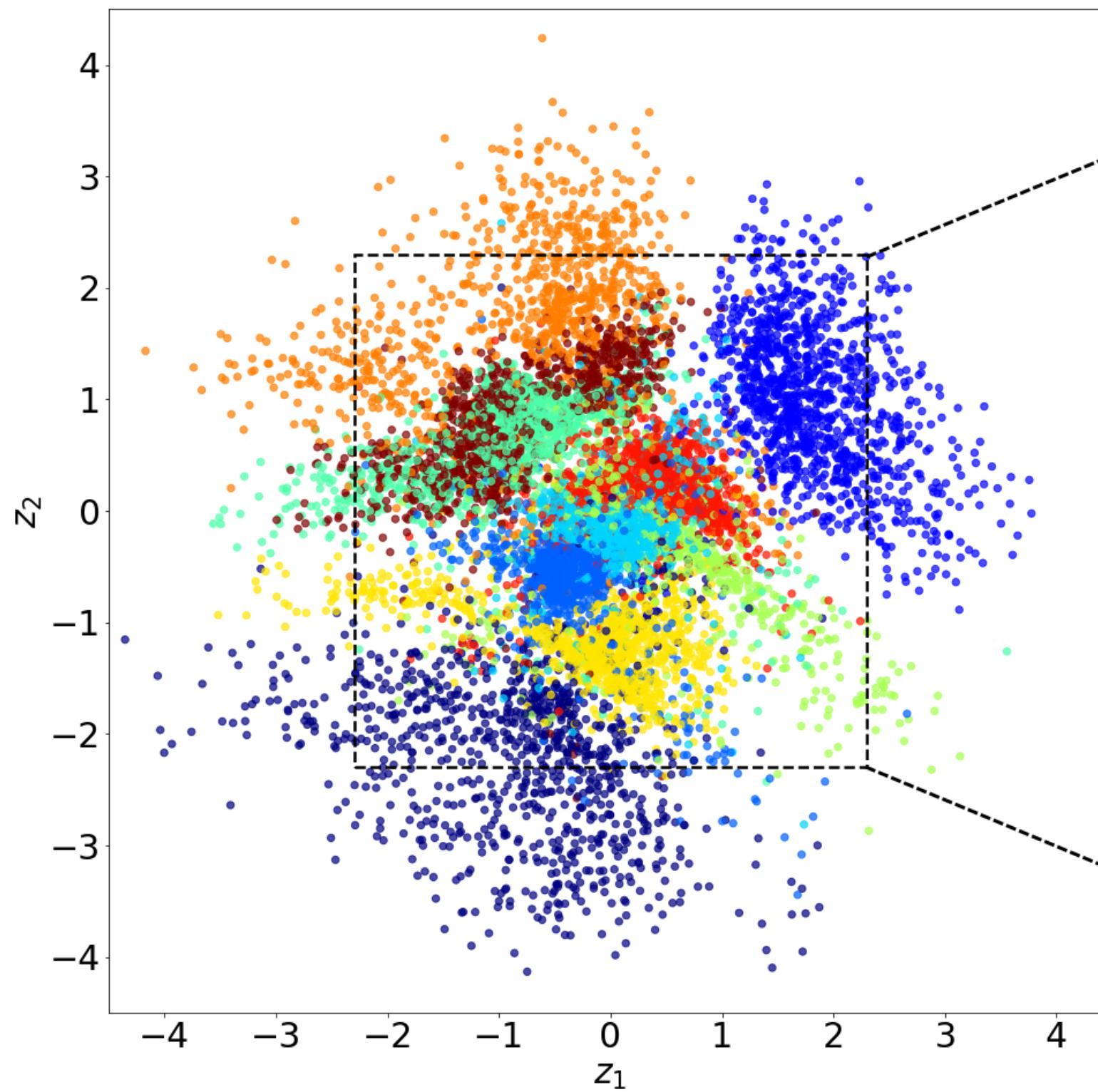
ders are being widely used in
generative models since the day they came into
existence. This blog is just a synopsis of understanding...



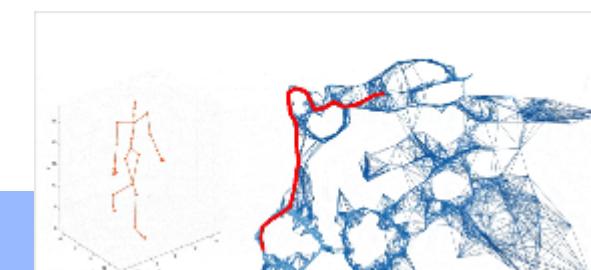
L'espace latent

ici un exemple en 2D

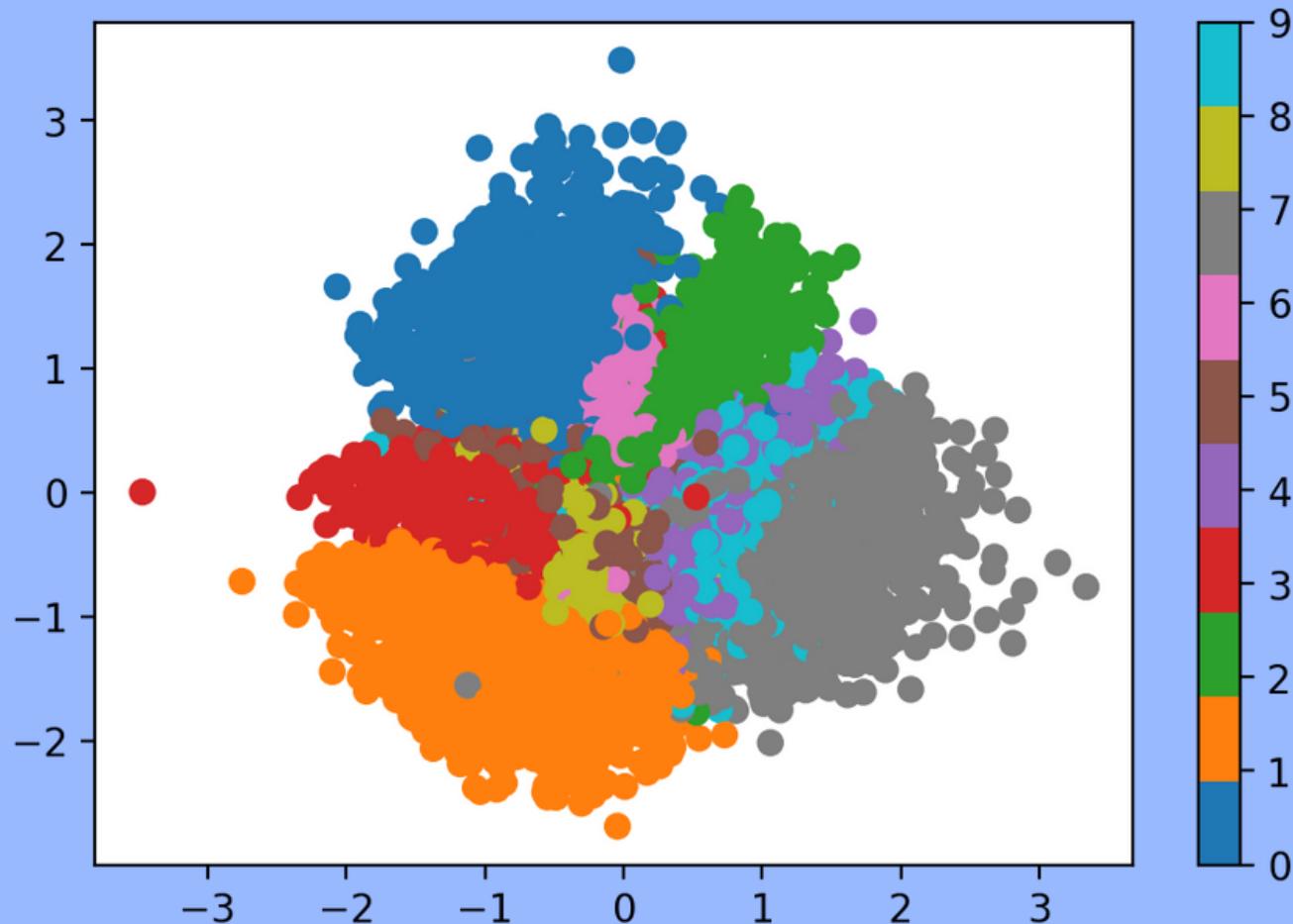
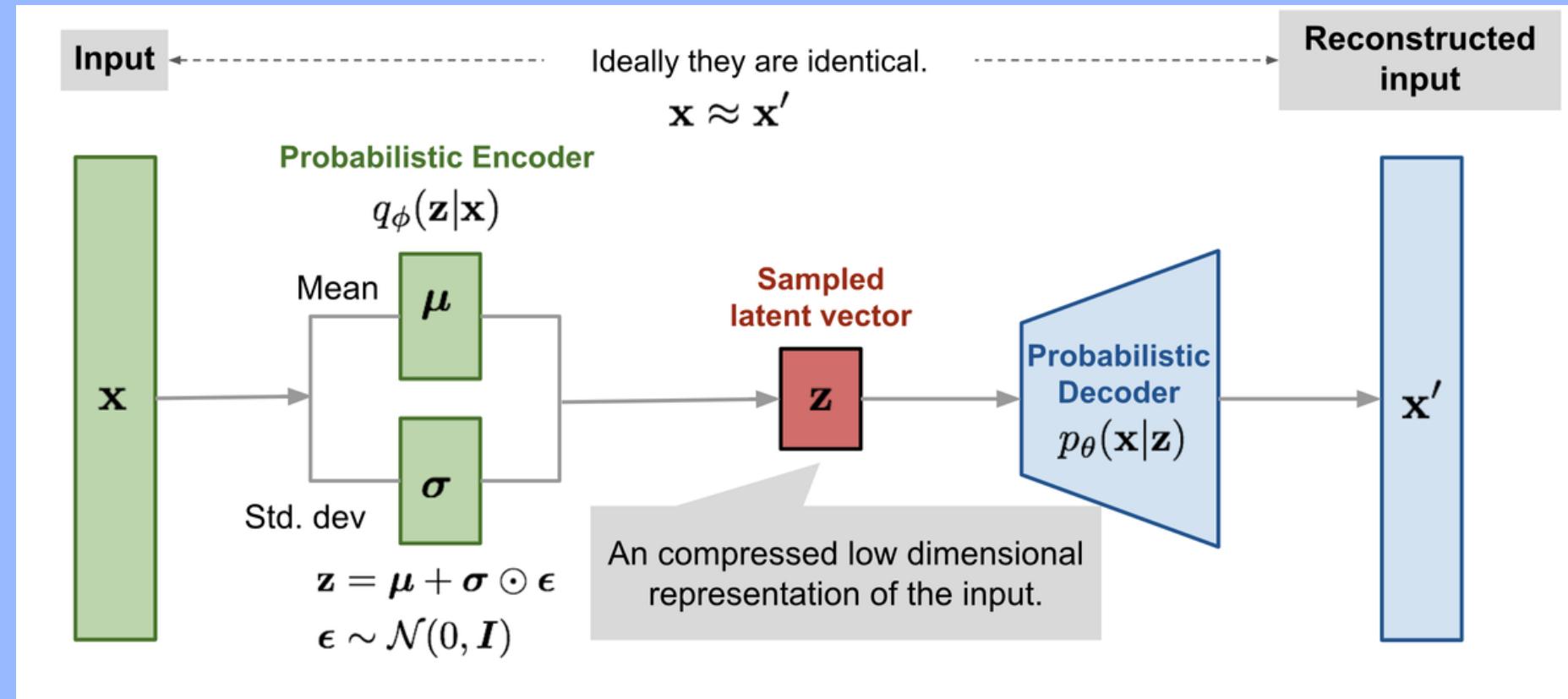


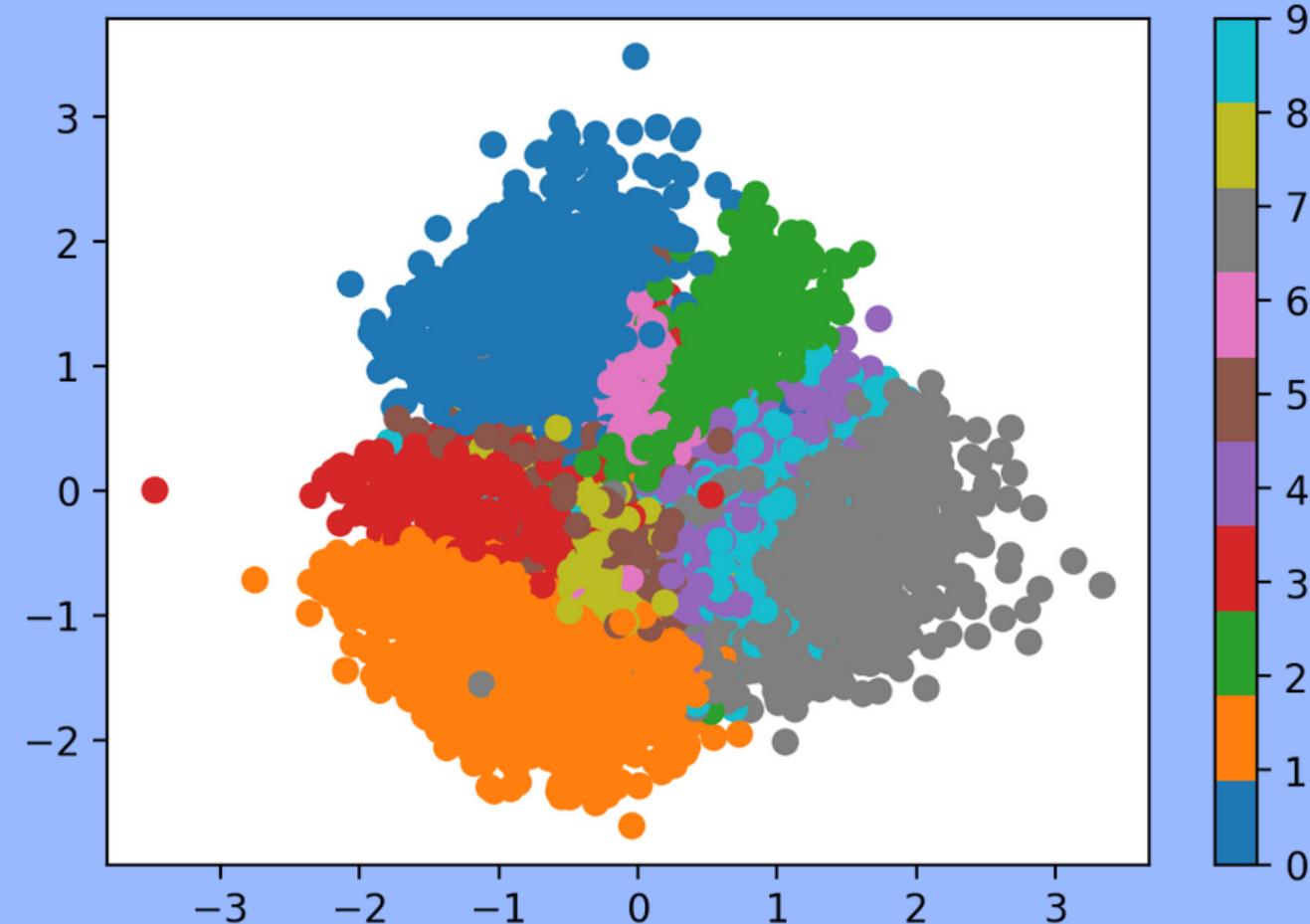
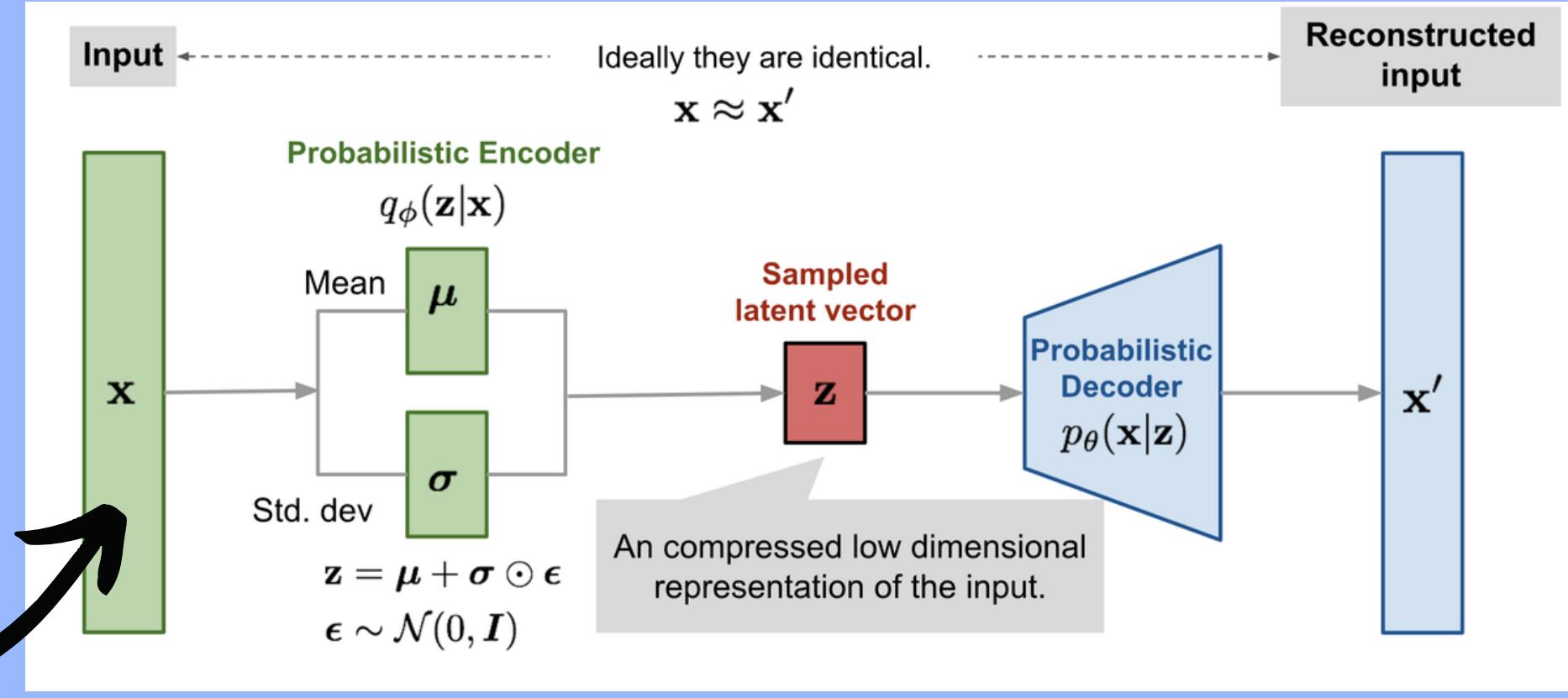
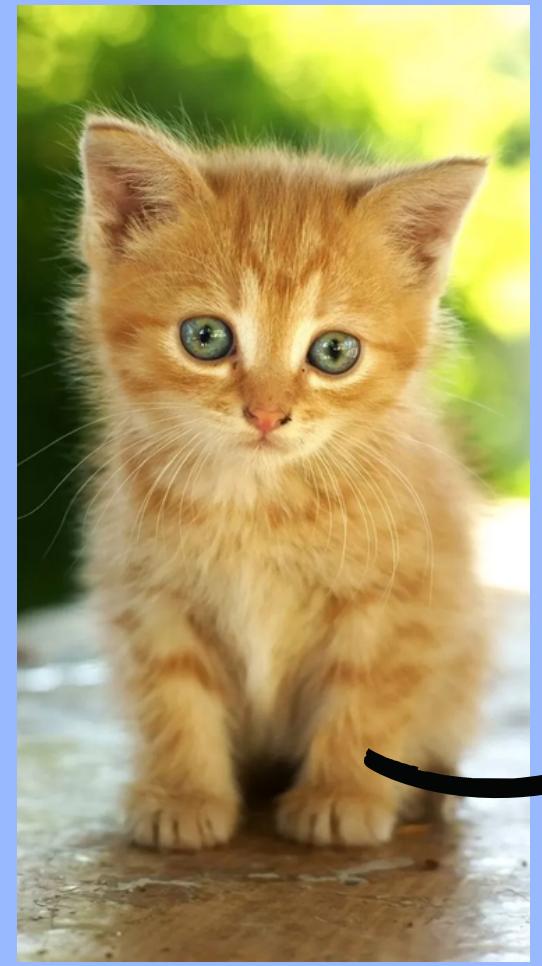


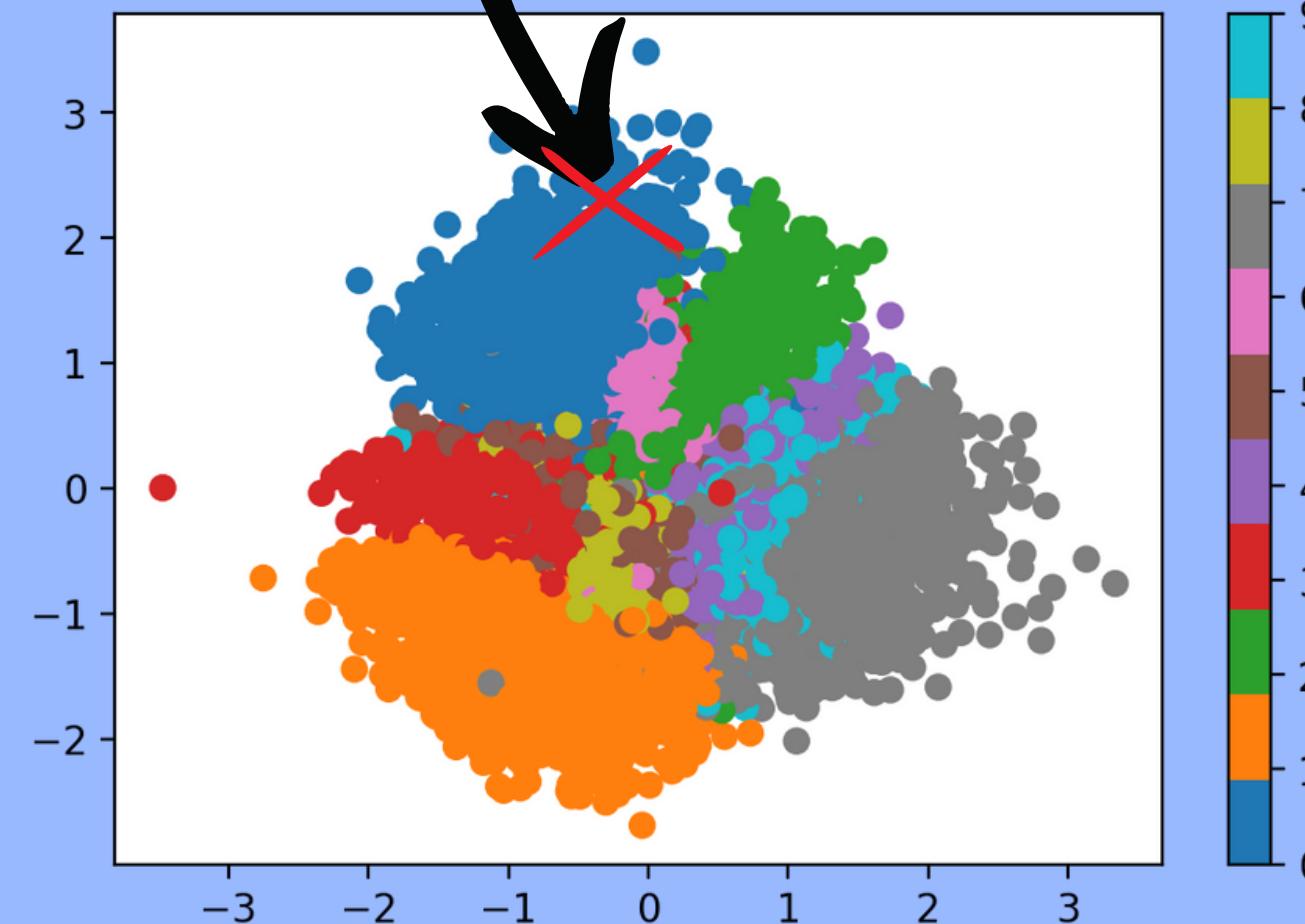
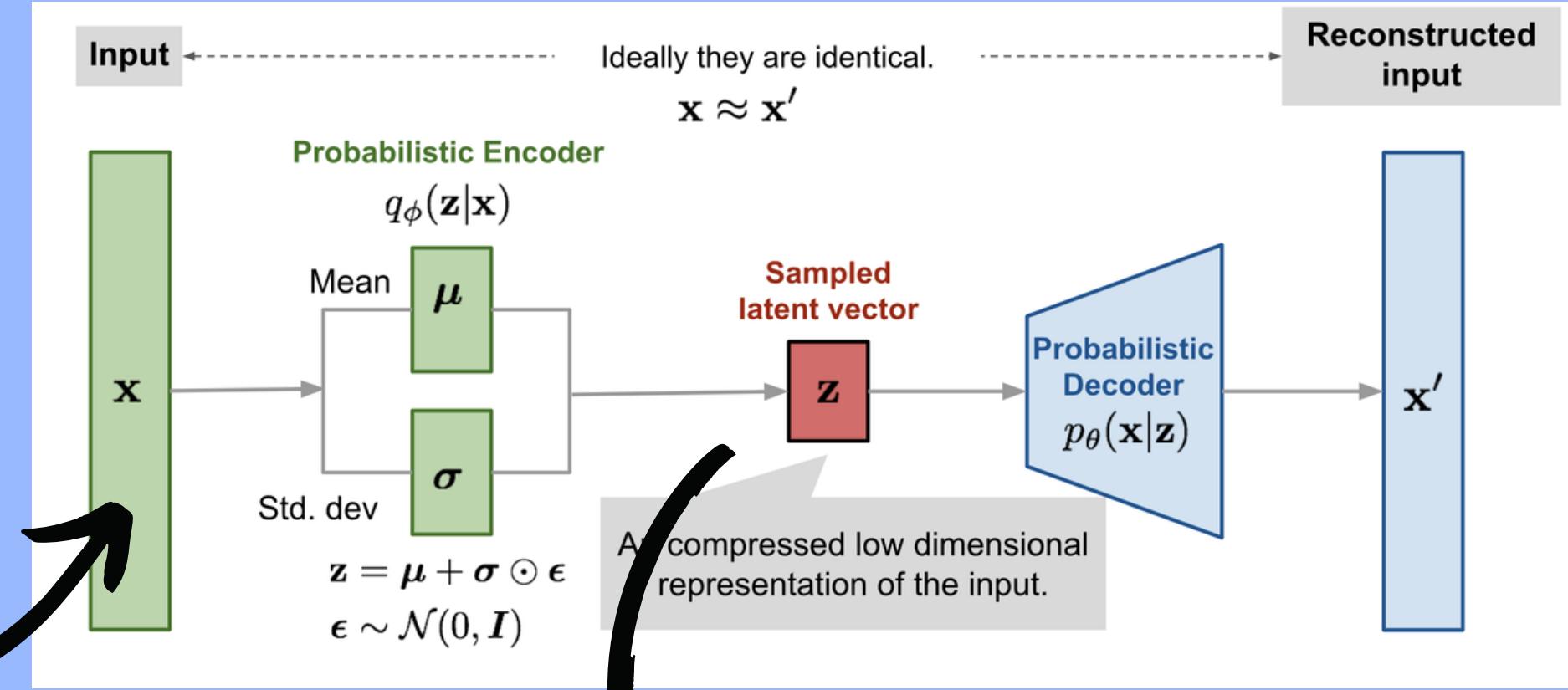
L'espace latent
ici un exemple en 2D

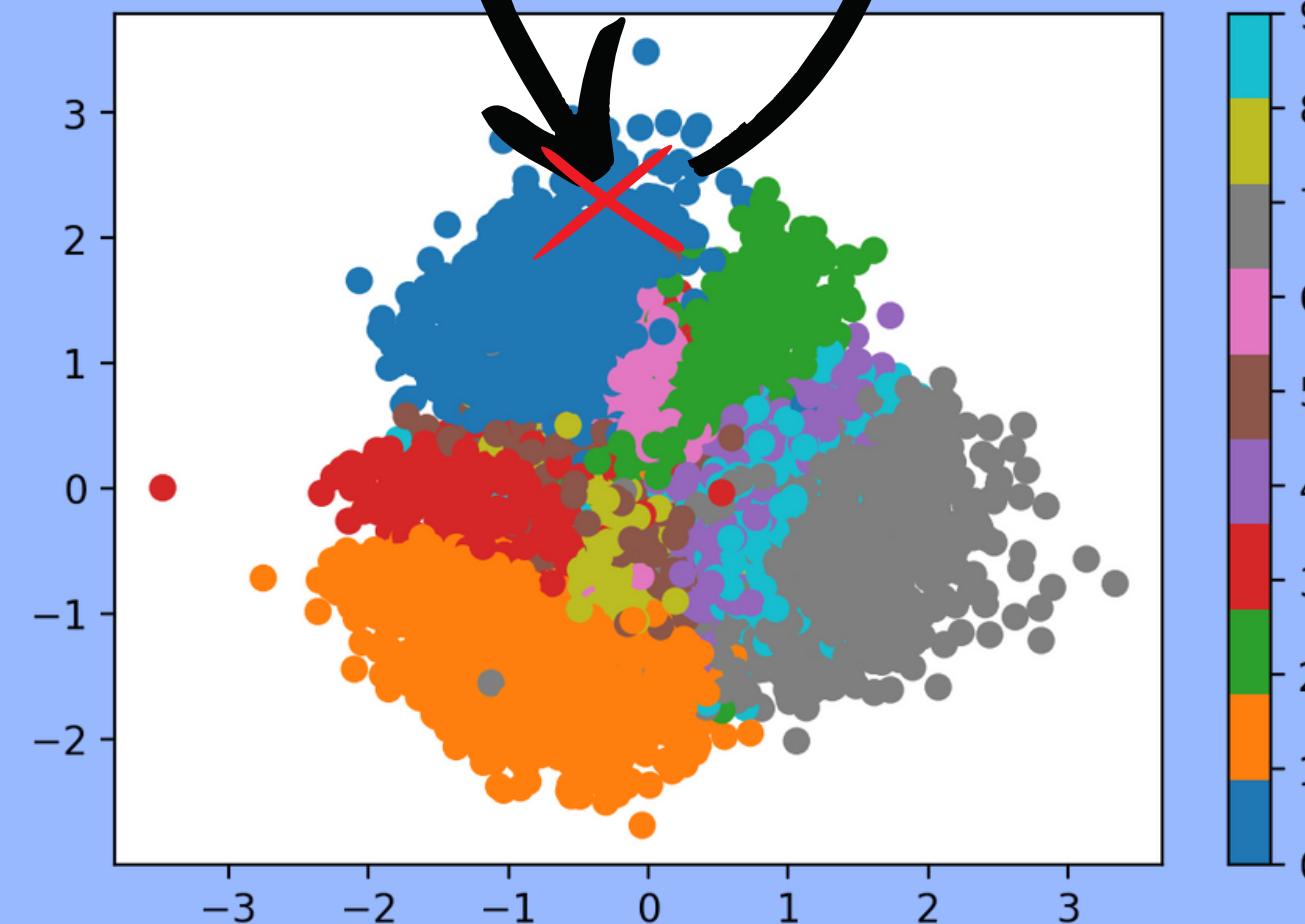
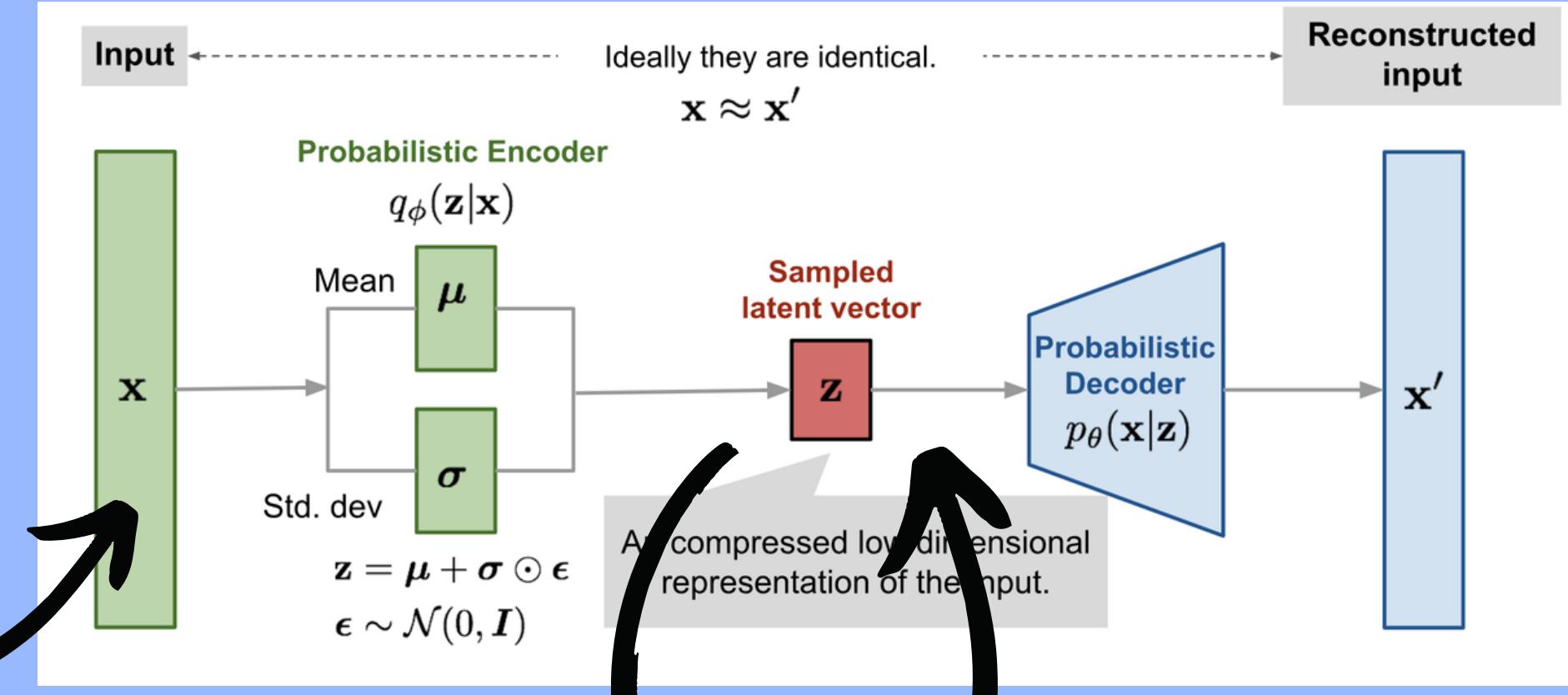


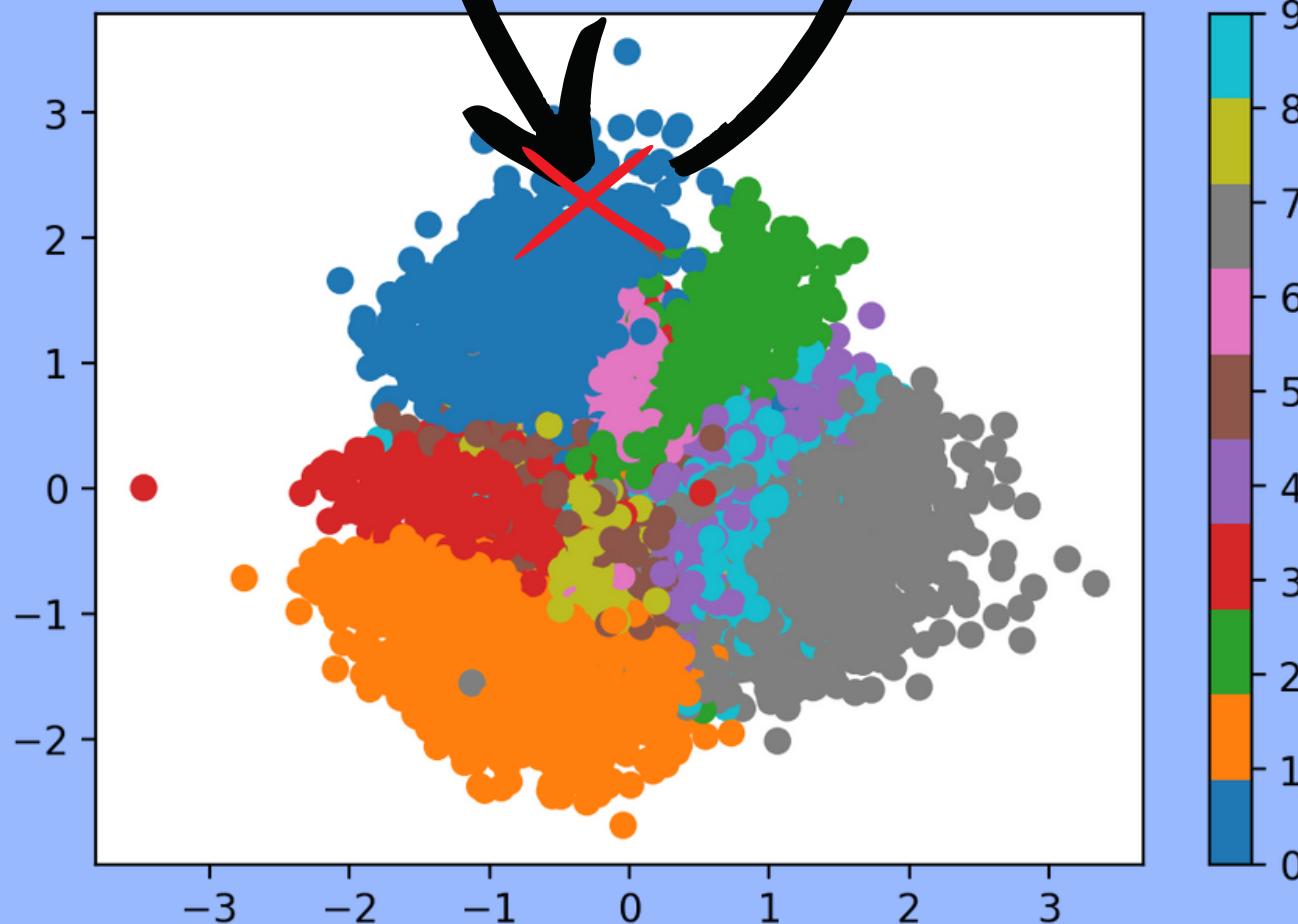
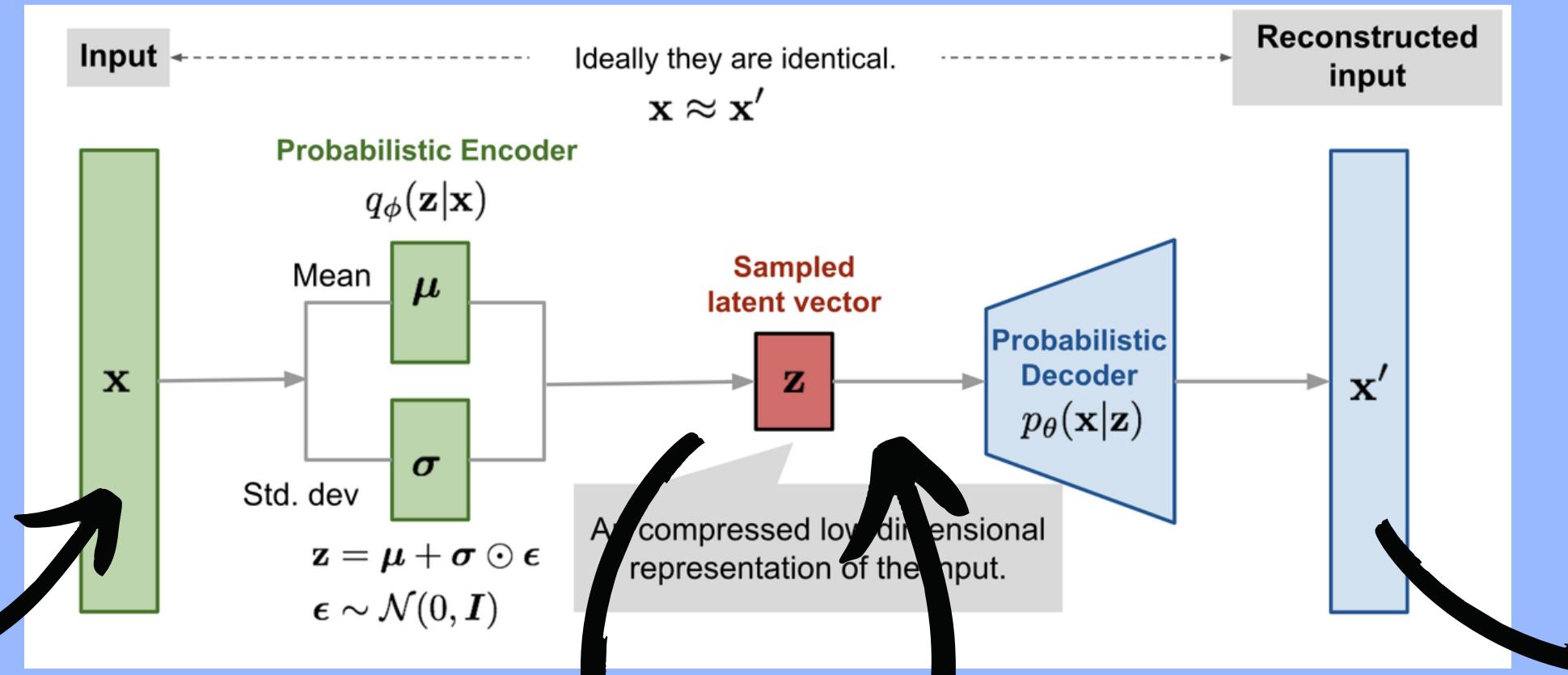
Learning Hierarchical Priors in VAEs
We address the issue of learning informative latent representations of data. In the normal VAE, the latent space prior is a standard normal distribution. This...











L'optimisation du VAE peut être réalisée par l'algorithme AEVB.

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

```
 $\theta, \phi \leftarrow$  Initialize parameters  
repeat  
     $\mathbf{X}^M \leftarrow$  Random minibatch of  $M$  datapoints (drawn from full dataset)  
     $\epsilon \leftarrow$  Random samples from noise distribution  $p(\epsilon)$   
     $\mathbf{g} \leftarrow \nabla_{\theta, \phi} \tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M, \epsilon)$  (Gradients of minibatch estimator (8))  
     $\theta, \phi \leftarrow$  Update parameters using gradients  $\mathbf{g}$  (e.g. SGD or Adagrad [DHS10])  
until convergence of parameters  $(\theta, \phi)$   
return  $\theta, \phi$ 
```

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^J \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2 \right) + \frac{1}{L} \sum_{l=1}^L \log p_{\theta}(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

Lower-bound dans notre cas

EXEMPLES D'APPLICATIONS

La génération de données

On génère un espace latent, et on prend un point dans cette espace latent pour générer une donnée.

La compression de données

En apprenant une représentation latente compressée via apprentissage automatique probabiliste.

L'apprentissage non supervisé

Pour apprendre les structures des données sans étiquettes.

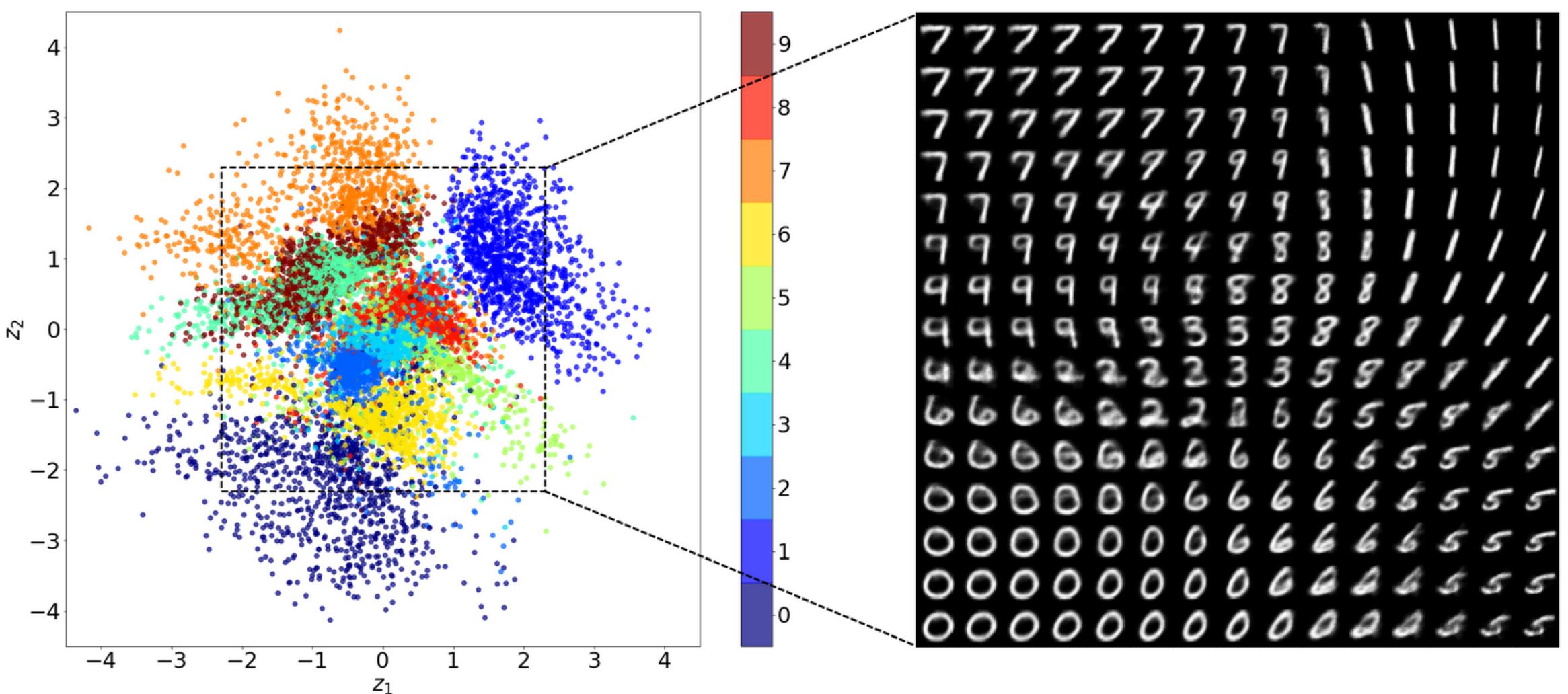
La détection de fraude

En cas de placement suspect dans l'espace latent.

EXEMPLES D'APPLICATIONS

La génération de données

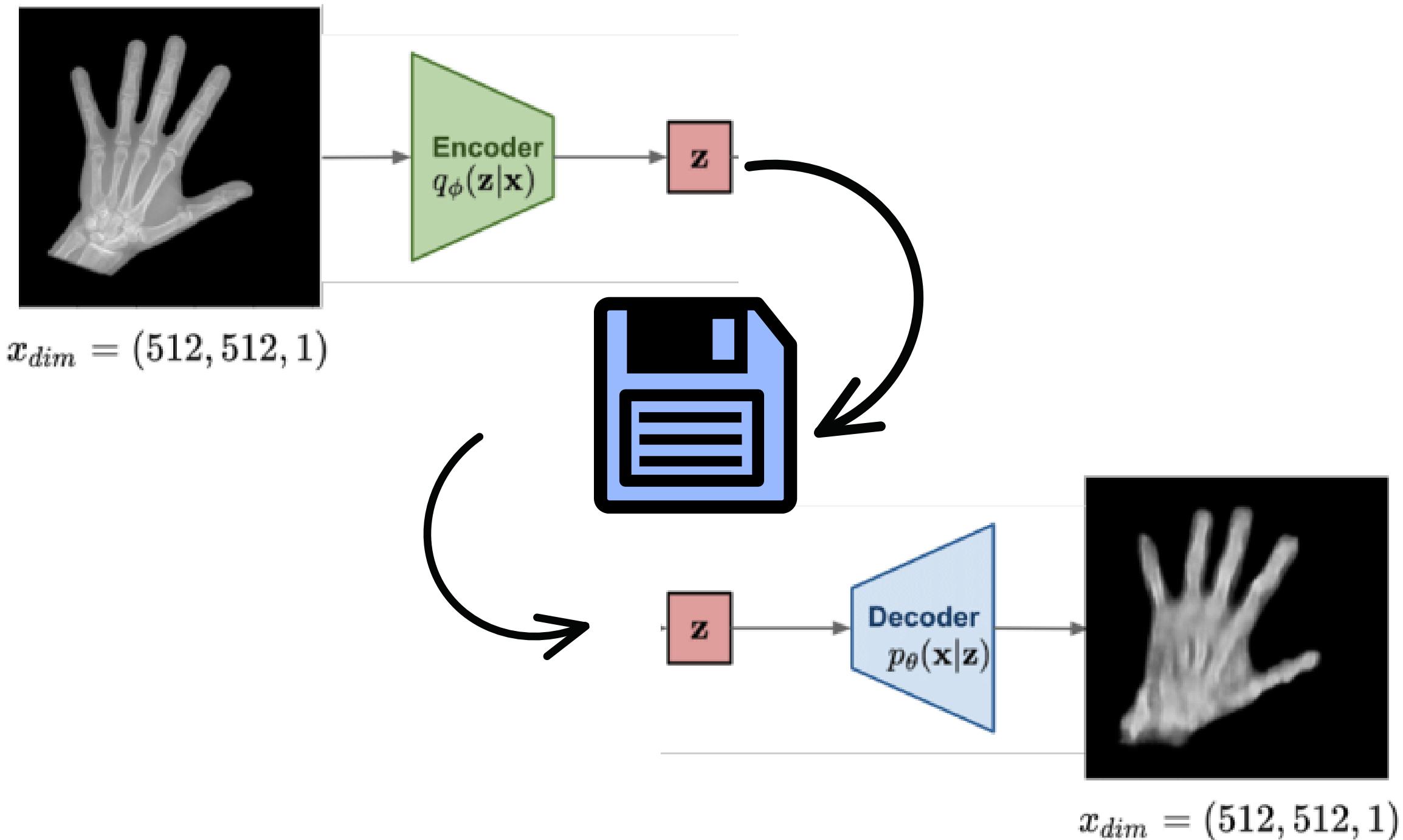
On génère un espace latent, et on prend un point dans cette espace latent pour générer une donnée.



EXEMPLES D'APPLICATIONS

La compression de données

En apprenant une représentation latente compressée via apprentissage automatique probabiliste.

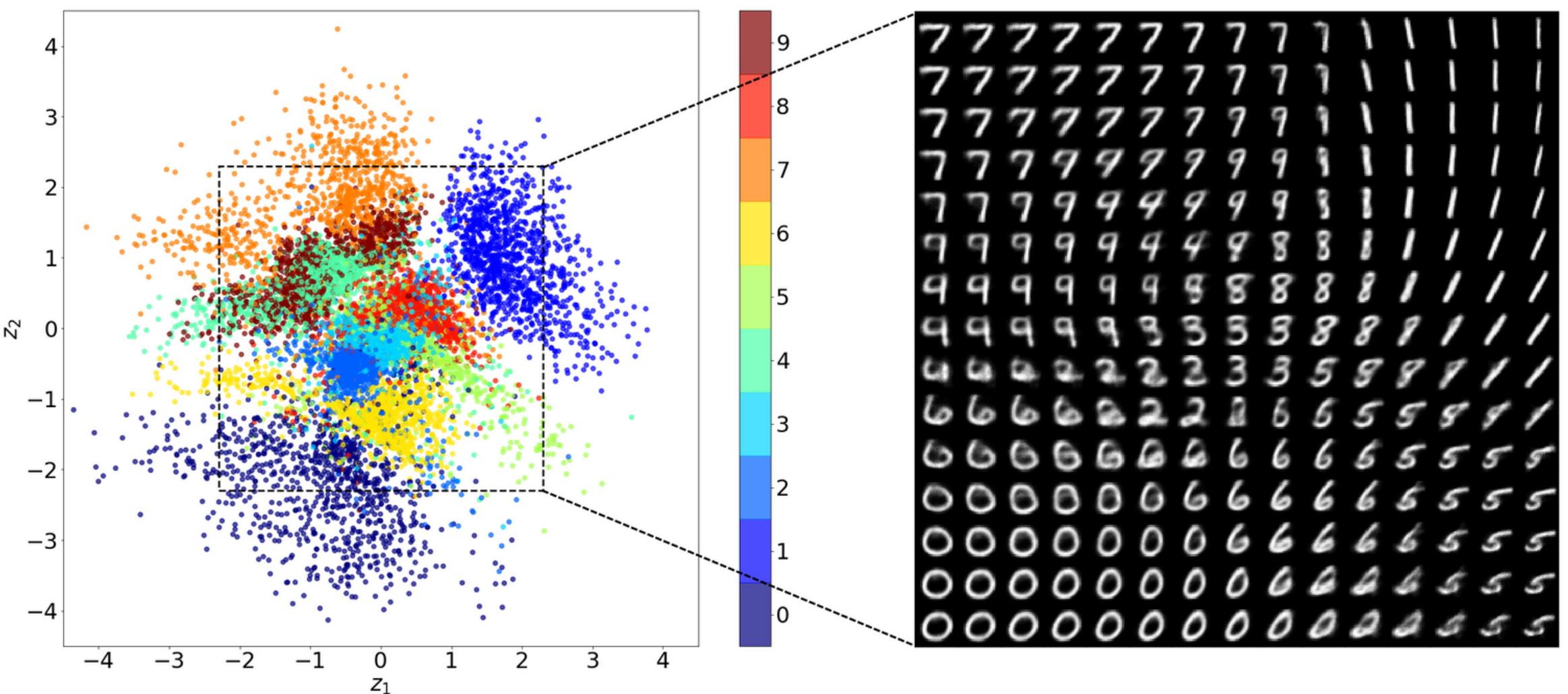


$x_{dim} = (512, 512, 1)$

EXEMPLES D'APPLICATIONS

L'apprentissage non supervisé

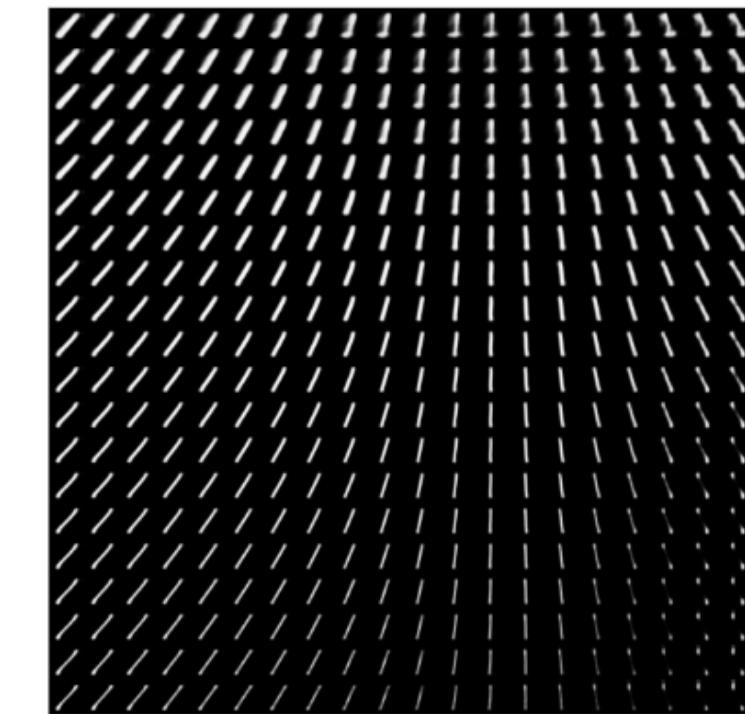
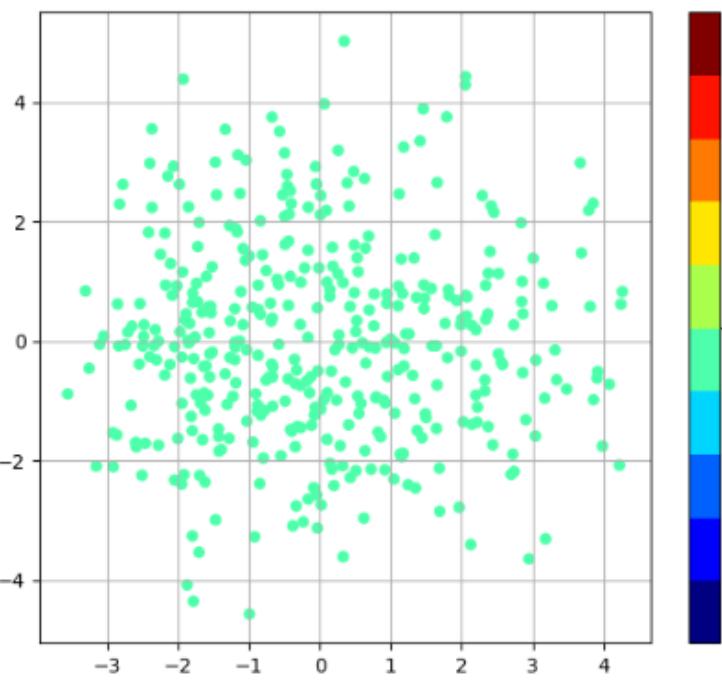
Pour apprendre les structures des données sans étiquettes.



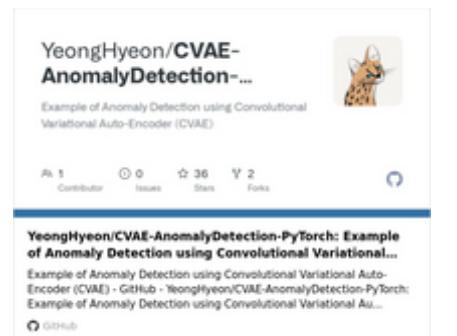
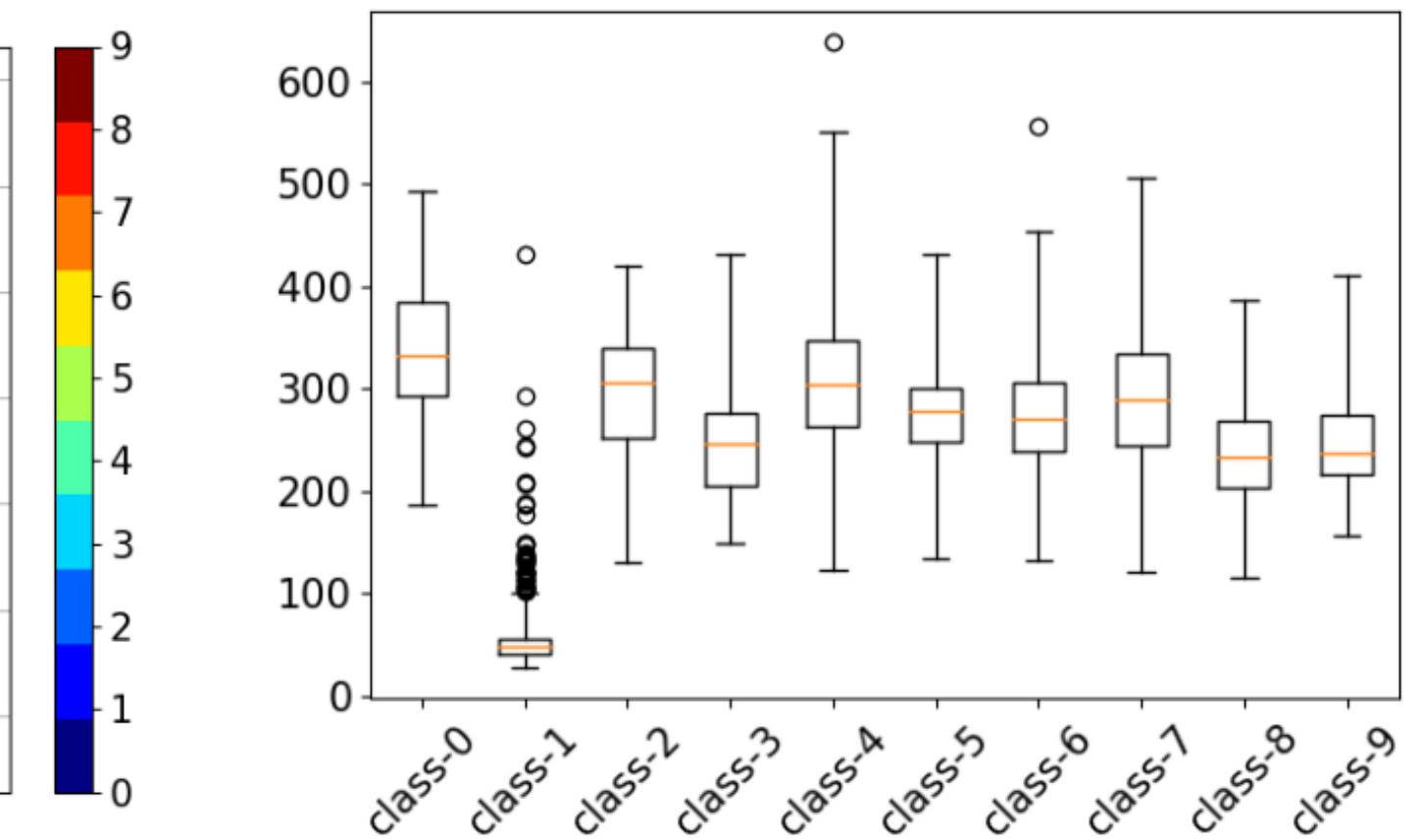
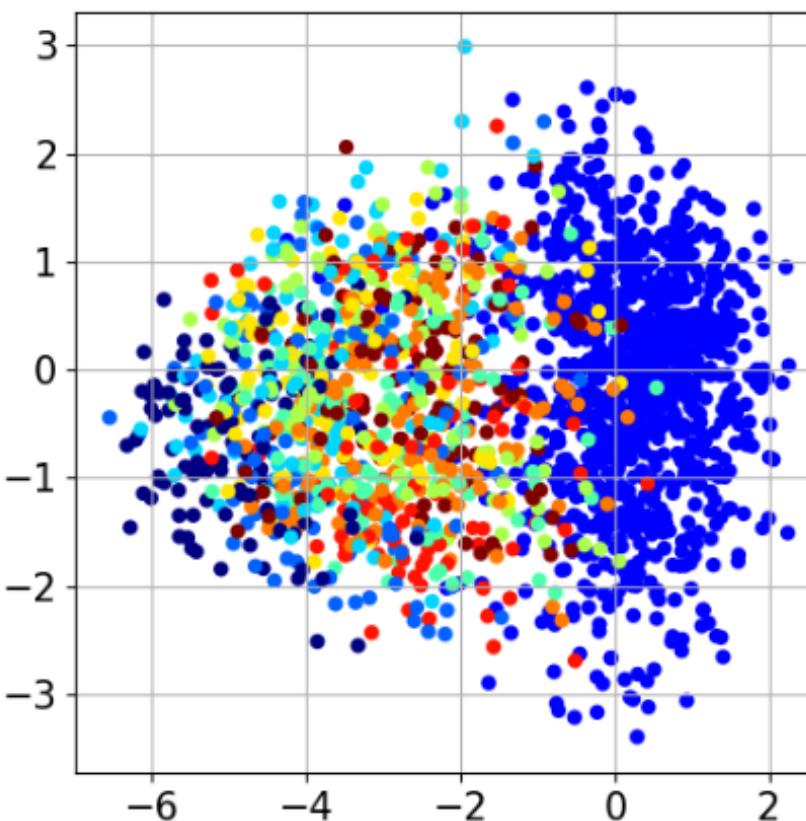
EXEMPLES D'APPLICATIONS

La détection de fraude

En cas de placement suspect dans l'espace latent.



Latent vector space of training set, and reconstruction result of latent space walking.



4. APPLICATION AVEC MNIST

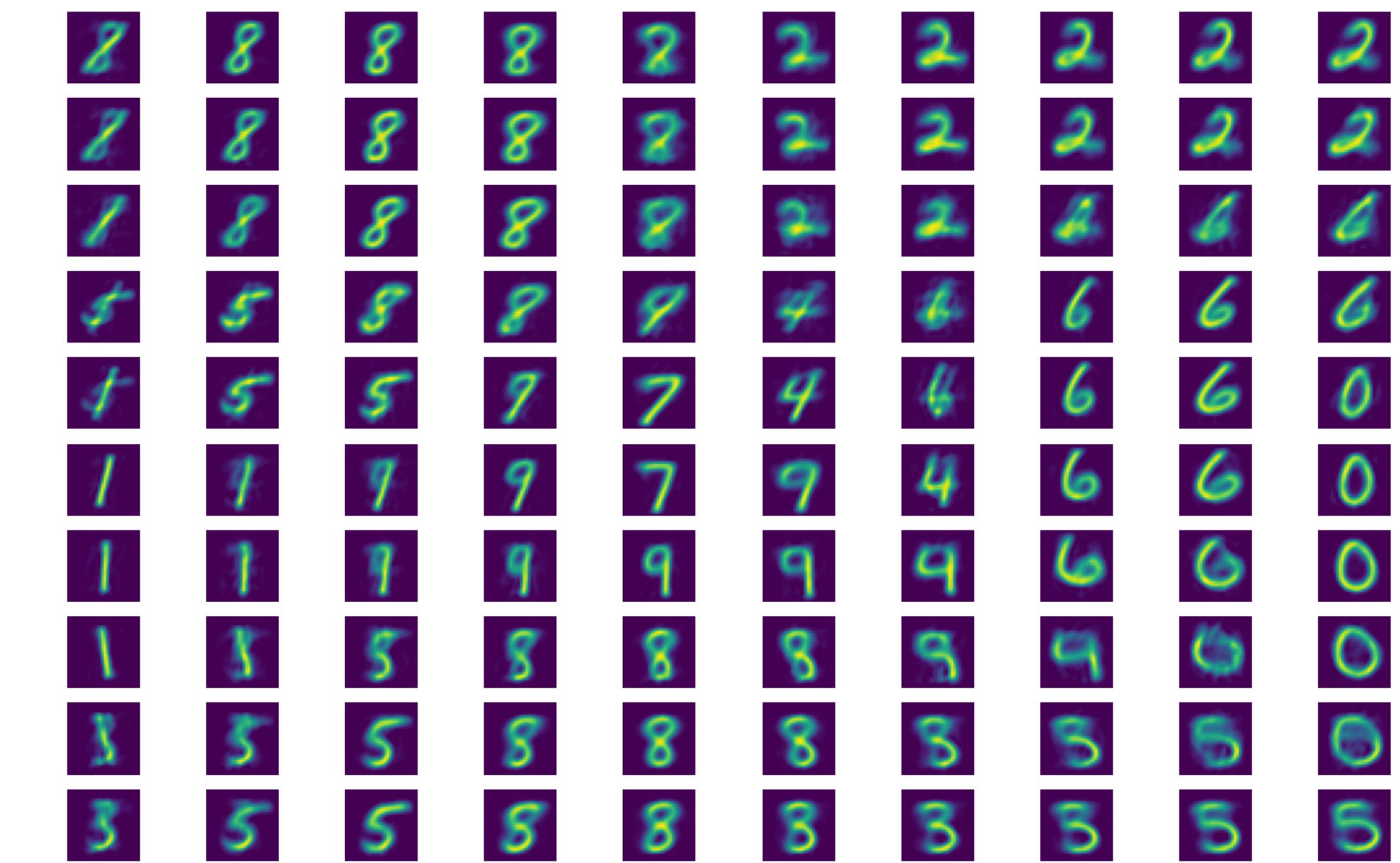
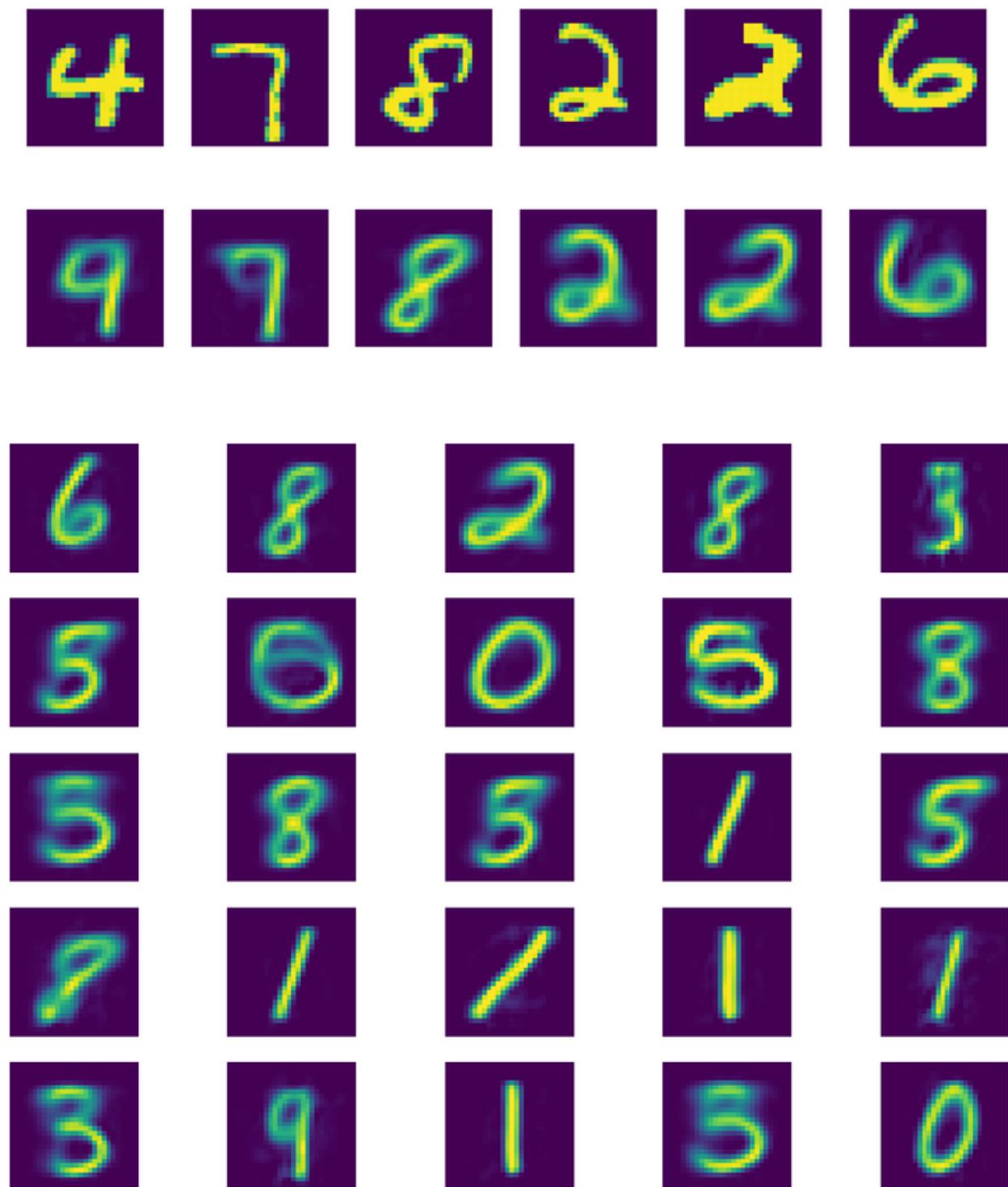
```
def forward(self, x: torch.Tensor, sample: bool = True) -> Dict[str, torch.Tensor]:  
    z_all = self._encoder(x)  
    z_mean, z_log_var = z_all[..., :self._latent_dim], z_all[..., self._latent_dim:]  
  
    if sample:  
        z_std = torch.sqrt(torch.exp(z_log_var))  
        e = torch.randn_like(z_mean)  
        z = z_mean + z_std * e  
    else:  
        z = z_mean  
  
    x_hat = self._decoder(z)  
    return {  
        'x': x_hat,  
        'z': z,  
        'z_mean': z_mean,  
        'z_log_var': z_log_var  
    }
```

```
class GaussianELBOLoss(nn.Module):  
    """  
    Computes the ELBO loss for a multivariate Gaussian distribution with a diagonal covariance matrix.  
    """  
  
    def __init__(self, noise: float):  
        """  
        Args:  
            noise: Value of the covariance matrix, a constant.  
                Increasing this value enhances the regularization effect.  
        """  
        super().__init__()  
        self._noise = noise  
  
    def forward(self, x_hat: torch.Tensor, x_target: torch.Tensor, z_mean: torch.Tensor, z_log_var: torch.Tensor) -> torch.Tensor:  
        likelihood_loss = torch.mean(1 / (2 * self._noise) * torch.sum(torch.square(x_hat - x_target), dim=-1))  
        z_var = torch.exp(z_log_var)  
        k1_loss = torch.mean(-1/2 * torch.sum(1 + z_log_var / 2 - z_var - torch.square(z_mean), dim=-1))  
        return likelihood_loss + k1_loss, likelihood_loss, k1_loss
```

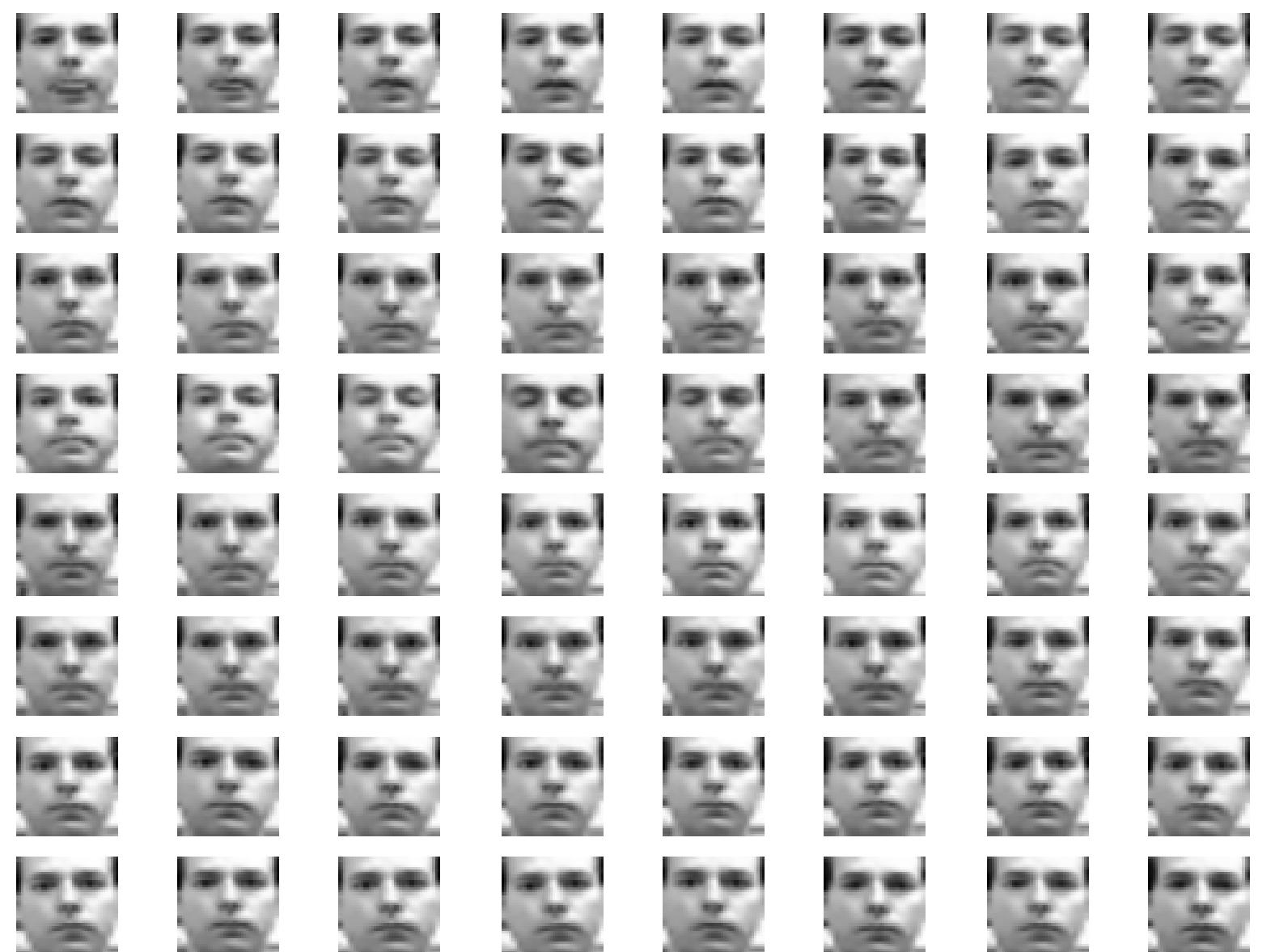
```
encoder = nn.Sequential(  
    nn.Flatten(start_dim=-3), # size: 28*28 = 784  
    nn.Linear(image_flat_size, 392),  
    nn.ReLU(),  
    nn.Linear(392, 196),  
    nn.ReLU(),  
    nn.Linear(196, 2 * latent_dim)  
)
```

```
decoder = nn.Sequential(  
    nn.Linear(latent_dim, 196),  
    nn.ReLU(),  
    nn.Linear(196, 392),  
    nn.ReLU(),  
    nn.Linear(392, image_flat_size),  
    nn.Unflatten(-1, image_shape)
```

4. APPLICATION AVEC MNIST

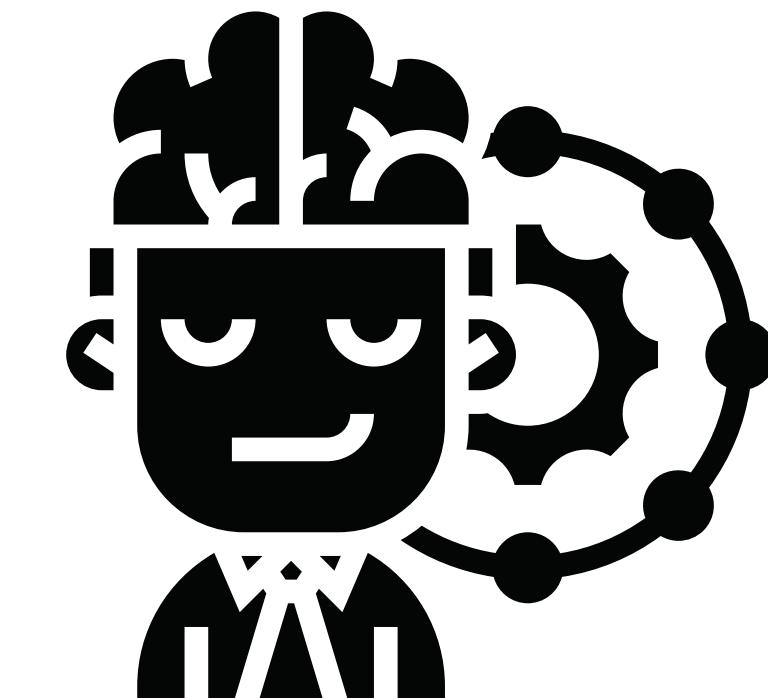
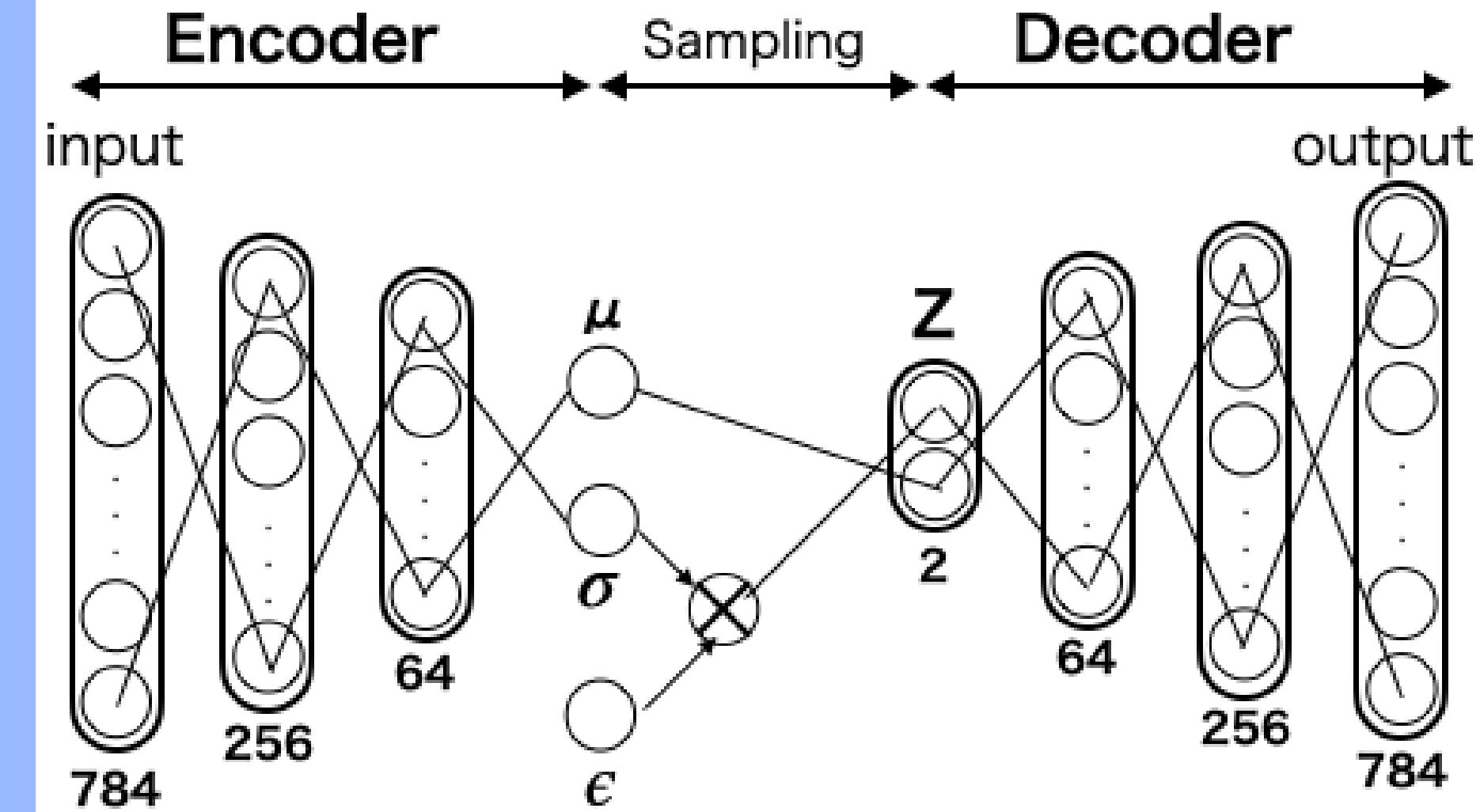


5. APPLICATION AVEC FREY FACE



6. CONCLUSION

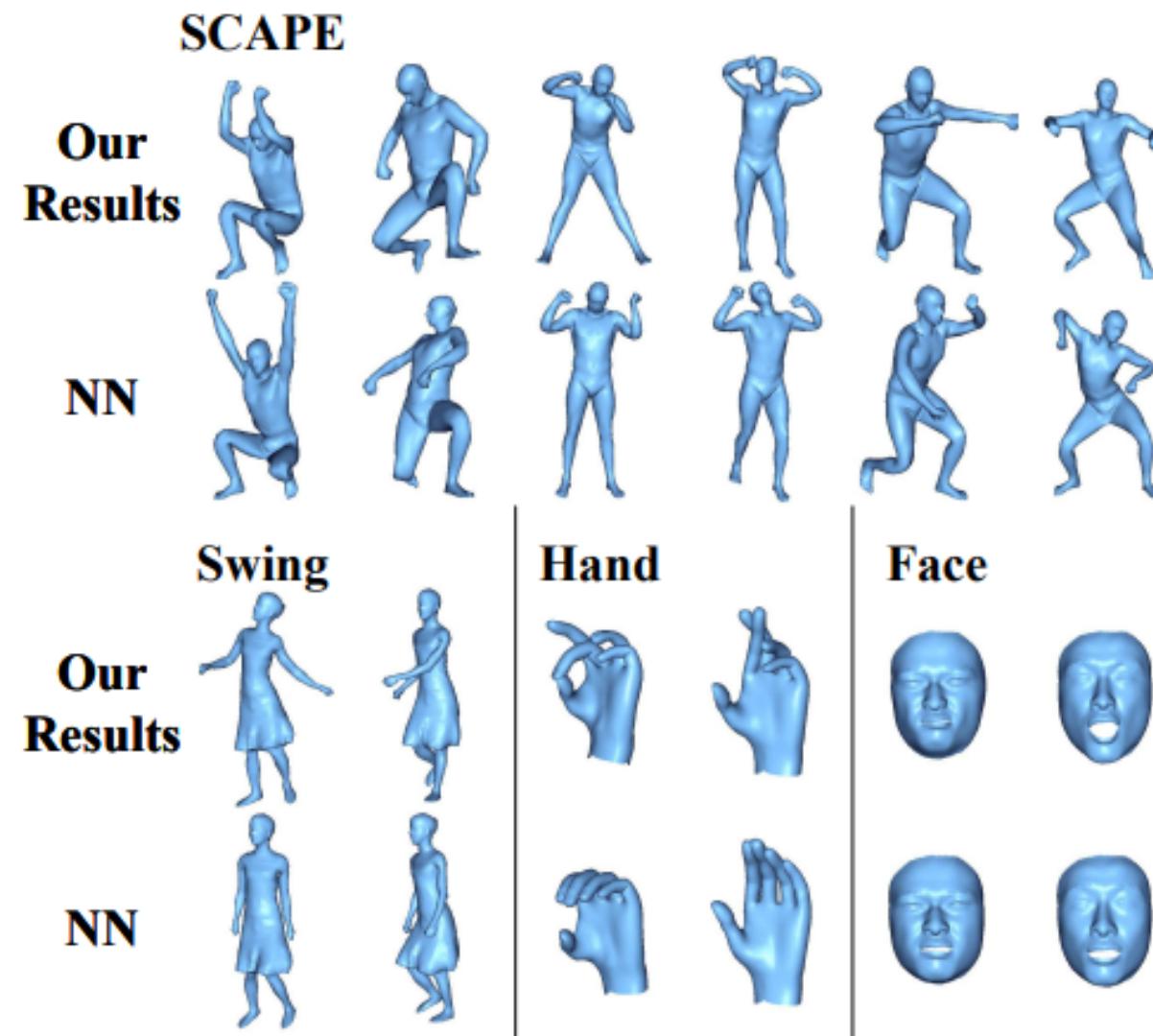
- Le mix entre un Auto-encoder et des méthodes variationnelles va générer un espace latent à partir d'une distribution normale.
- L'astuce de reparamétrisation est la clé d'un VAE.
- Capacité à générer des données nouvelles et à effectuer une interpolation continue dans l'espace latent.



7. L'EVOLUTION DEPUIS LA PUBLICATION DU PAPIER

Apprentissage d'architectures génératives hiérarchiques avec des réseaux de neurones profonds

e.g. Convolutional network for Deforming 3D Mesh Models



Produire des vecteurs raisonnables dans l'espace latent pour générer de nouveaux modèles en dehors de l'ensemble de données d'entraînement

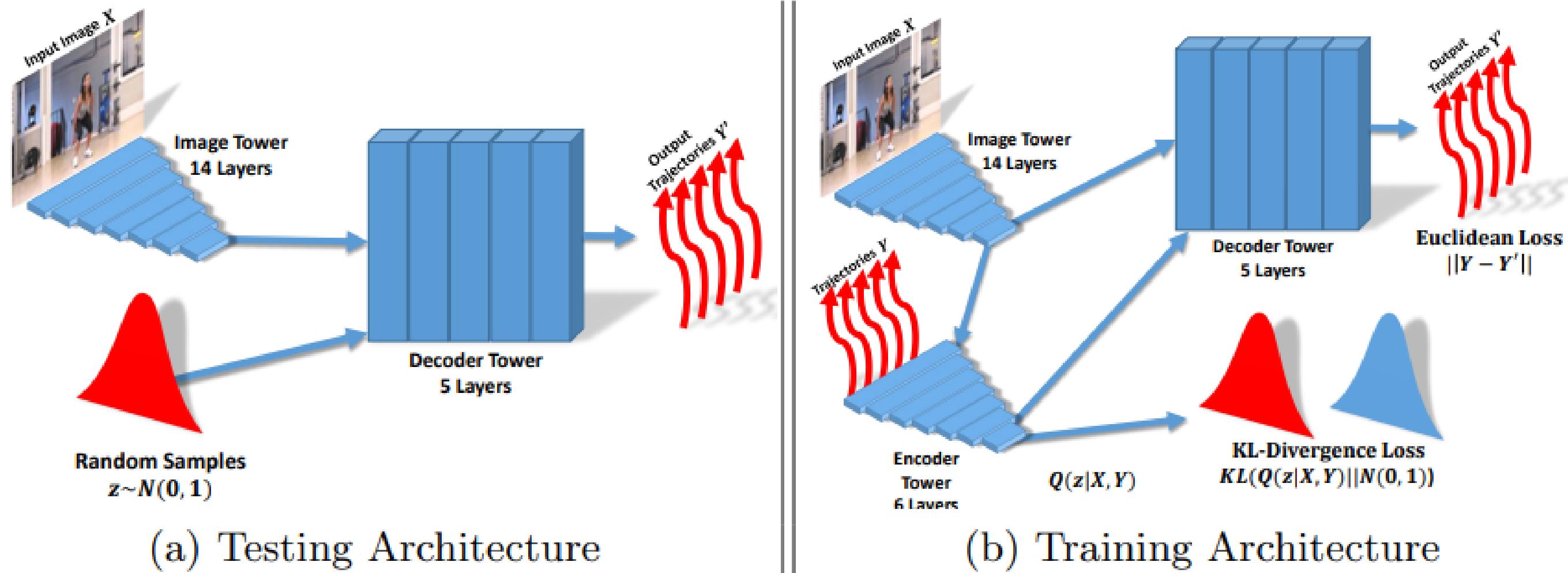
Insérez des vecteurs latents dans des formes nouvellement générées pour produire des séquences de déformation

Nouvelles formes générées aléatoirement à l'aide de VAE, avec leurs voisins les plus proches dans l'ensemble de données d'origine présenté ci-dessous.

7. L'EVOLUTION DEPUIS LA PUBLICATION DU PAPIER

Apprentissage de modèles génératifs pour les données temporelles avec structure dynamique

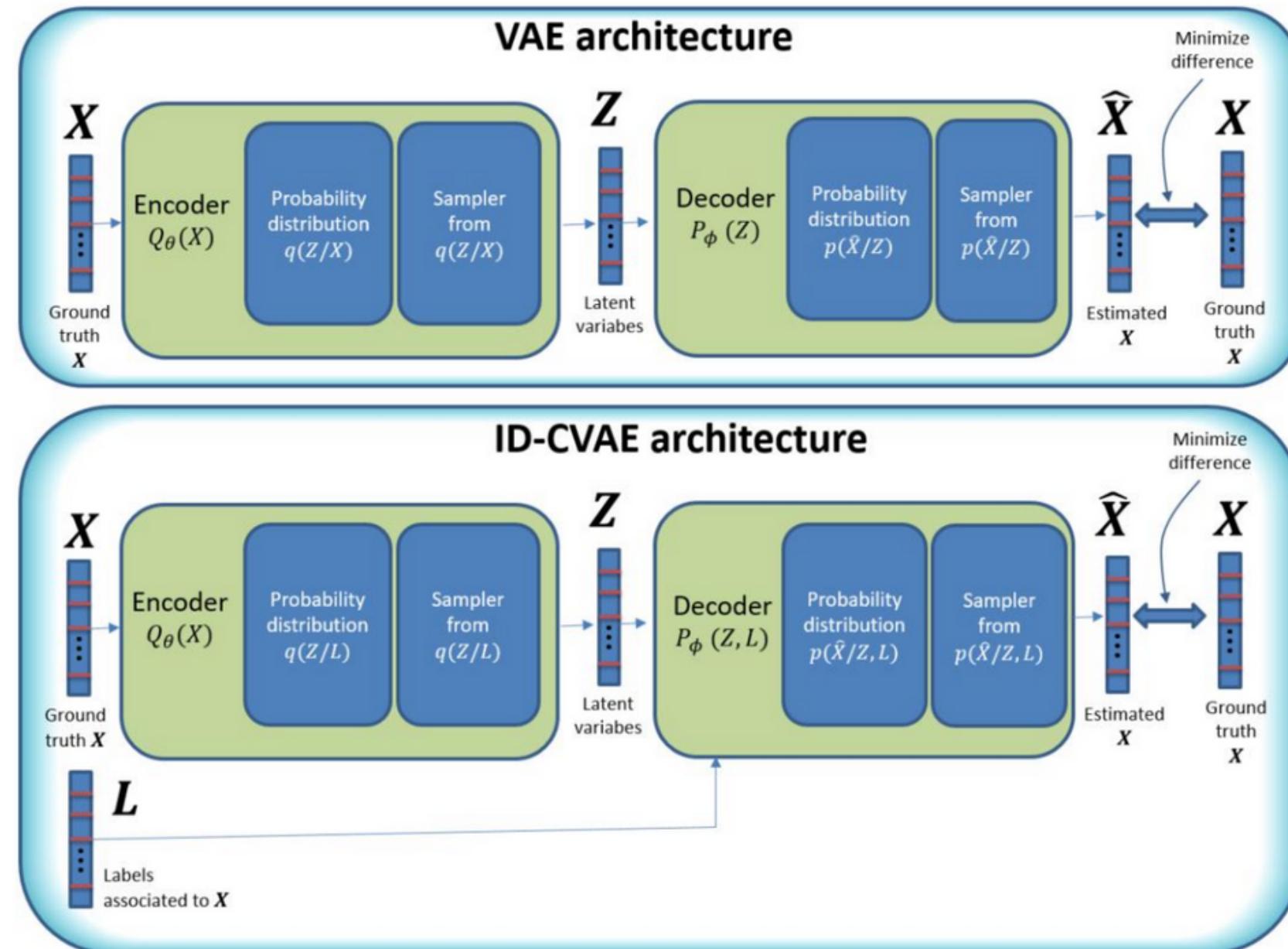
e.g. **An Uncertain Future: Forecasting from Static Images using Variational Autoencoders**



7. L'EVOLUTION DEPUIS LA PUBLICATION DU PAPIER

Modèles de variables latentes supervisées

e.g. **Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT**



La VAE conditionnelle est une VAE étendue qui introduit des informations conditionnelles pour permettre de modéliser des données dans des conditions données.

La récupération ultérieure des fonctionnalités peut être effectuée lorsque les fonctionnalités de l'échantillon d'entrée sont incomplètes.