


# Auto-Encoding Variational Bayes (Auto-Encodeur Variationnel)

A l'attention de Bertrand MICHEL & Matthieu RIBATET

---

24 mars 2025

DOBELI Pierre  
DAVOINE Camille  
FAESSEL Mathieu



# Sommaire

**01** Autoencodeurs classiques: concept et limites

**02** Introductions aux Auto-encodeurs Variationnels (VAE)

**03** Formulation variationnelle et ELBO

**04** Entraînement pratique d'un VAE

**05** Applications pratiques, limite et perceptives

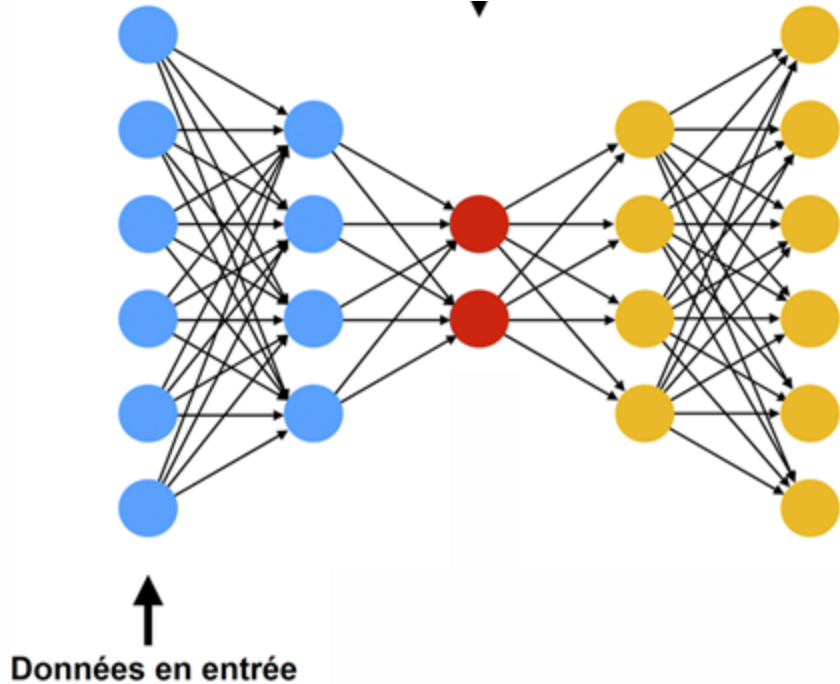
**06** Conclusion

# 01

## **Autoencodeurs classiques: concept et limites**



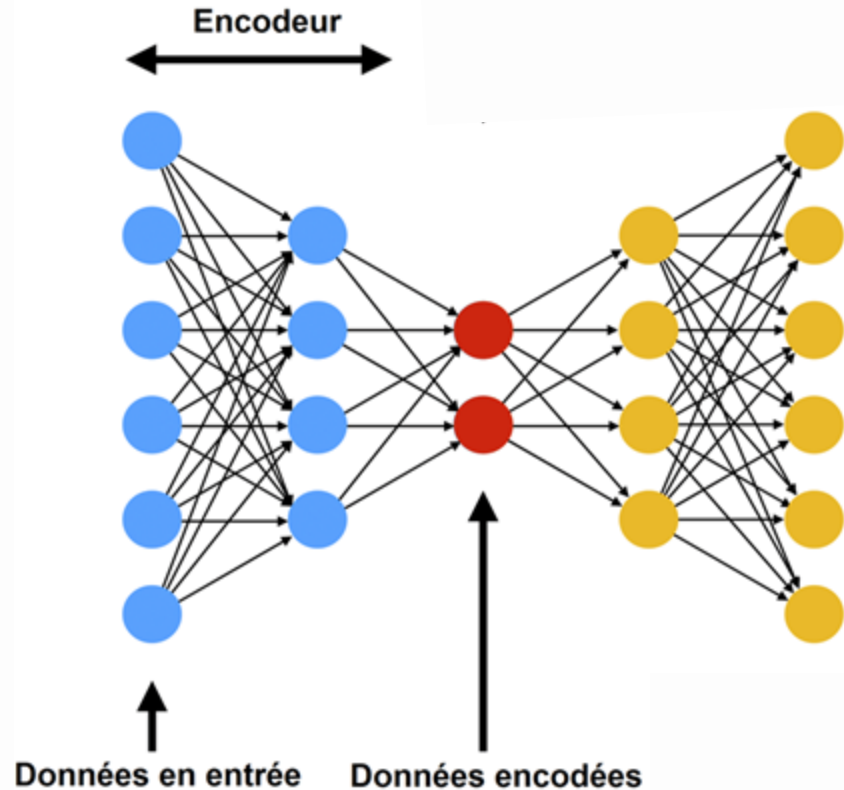
# Qu'est-ce qu'un Autoencodeur ?



## Architecture typique :

- **Entrée :**  $x$  (ex: image, signal audio, texte encodé...)

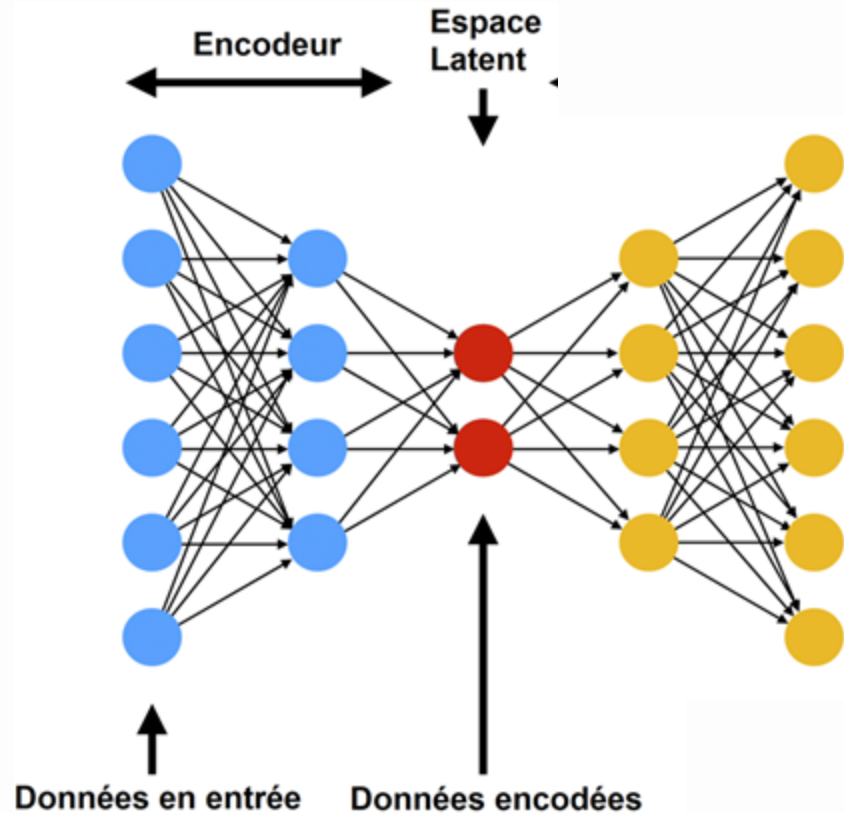
# Qu'est-ce qu'un Autoencodeur ?



## Architecture typique :

- **Entrée** :  $x$  (ex: image, signal audio, texte encodé...)
- **Couche d'encodeur** : Transforme  $x$  en une **représentation latente**  $z$ .

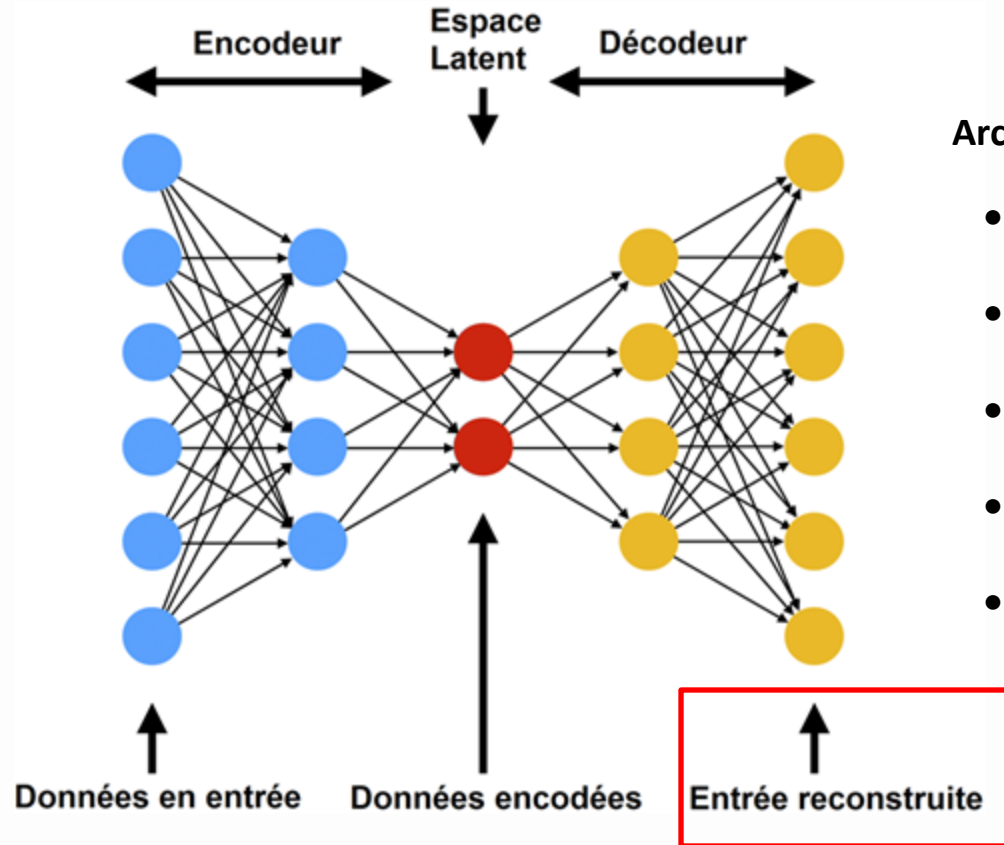
# Qu'est-ce qu'un Autoencodeur ?



## Architecture typique :

- **Entrée** :  $x$  (ex: image, signal audio, texte encodé...)
- **Couche d'encodeur** : Transforme  $x$  en une **représentation latente**  $z$ .
- **Latent Space** : Contient les **données compressées** de  $x$ .

# Qu'est-ce qu'un Autoencodeur ?



## Architecture typique :

- **Entrée** :  $x$  (ex: image, signal audio, texte encodé...)
- **Couche d'encodeur** : Transforme  $x$  en une **représentation latente**  $z$ .
- **Latent Space** : Contient les **données compressées** de  $x$ .
- **Couche de décodeur** : Essaie de reconstruire  $x'$  à partir de  $z$ .
- **Sortie** :  $x'$ , qui doit être aussi proche que possible de  $x$

## Et sa représentation mathématique?

$x$

$z = f_{\phi}(x) : \text{encodeur}$

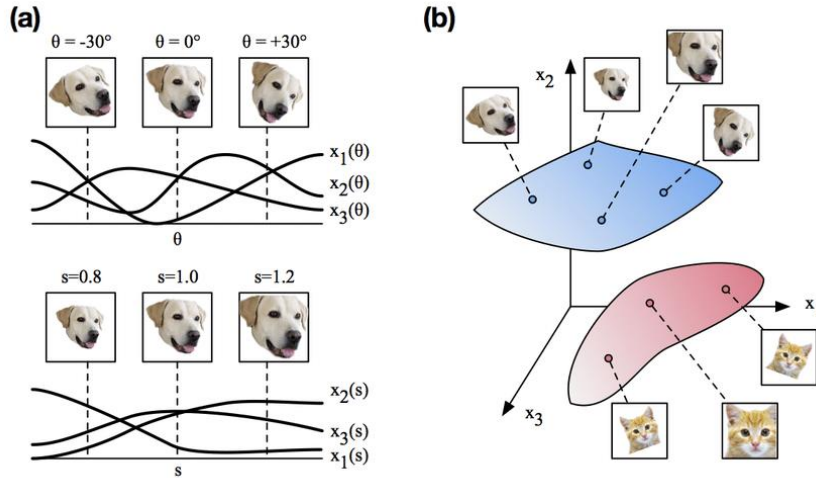
$x' = g_{\theta}(z) : \text{décodeur}$

$$\mathcal{L}_{rec} = \arg \min_{\theta, \phi} ||x - x'||^2$$

**Objectif:** Minimiser l'erreur de reconstruction



# Comprendre l'Espace Latent



**Espace latent** : Représentation compacte des données initiale  
 $z$

**Classification** : Une nouvelle image est positionnée dans l'espace latent.  
 $x \rightarrow z$

**Génération** : Un point latent permet de créer une nouvelle image  
 $z \rightarrow x'$

# Limitations de l'autoencodeur

## Espace latent non structuré

- Données similaires possiblement éloignées dans l'espace latent

## Difficulté de générer des données réalistes

- Prise aléatoire dans l'espace latent -> bruit incohérent

## Risque de sur-apprentissage

# 02

## Introductions aux Auto-encodeurs Variationnels (VAE)

A decorative graphic on the right side of the slide consisting of numerous thin, dark blue lines that curve and flow from the top right towards the bottom right, creating a sense of movement and depth. The lines are more densely packed in some areas, creating a gradient of blue tones.

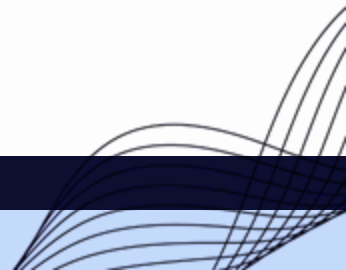
# Introduction des VAE : une approche probabiliste

Au lieu d'apprendre un point  $z$ , on veut que l'**encodeur** apprenne une distribution:

$$z \sim p_{\theta}(z|x)$$

⇒ Impose une structure à l'espace latent

⇒ Régularisation: on force  $p_{\theta}(z|x)$  à être proche d'une **distribution prior**  
 $p(z) = \mathcal{N}(0, I)$



# Inférence des variables latentes (Bayes)

Distribution a posteriori

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

où :

- $p_{\theta}(x|z)$  correspond au modèle génératif.
- $p(z)$  est le prior imposé sur l'espace latent.
- $p(x)$  est la **vraisemblance marginale**, obtenue en intégrant sur toutes les valeurs possibles de  $z$  :

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



# Problème d'inférence exacte

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

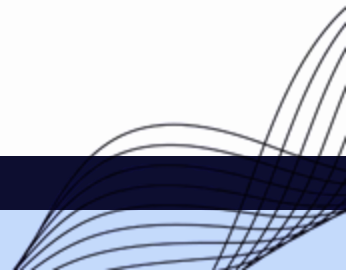
**Problème:** Impossible à calculer:

- Dimensionnalité élevée de  $z \rightarrow$  numériquement intractable
- Forme complexe des distributions
- Absence de solution analytique

**Solution:** [Approximation Variationnelle](#)

Pas de calcul direct de  $p_{\theta}(x)$

Approximation:  $p_{\theta}(z|x) \rightsquigarrow q_{\phi}(z|x)$  plus manipulable



# 03

## Formulation Variationnelle et ELBO

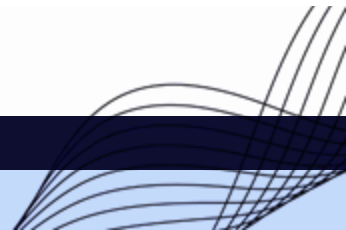
A decorative graphic consisting of multiple thin, dark blue wavy lines that originate from the bottom right corner and curve upwards and to the left, extending across the right side of the slide.

# Approche variationnelle

$$\log p_{\theta}(x) = D_{\text{KL}}(q_{\phi}(z|x)||p_{\theta}(z|x)) + \mathcal{L}(x)$$

où :

- $D_{\text{KL}}(q_{\phi}(z|x)||p_{\theta}(z|x))$  est la **divergence de Kullback-Leibler**, qui mesure l'écart entre notre approximation  $q_{\phi}(z|x)$  et la vraie postérieure  $p_{\theta}(z|x)$ . Cette quantité est toujours positive.
- $\mathcal{L}(x)$  est l'**ELBO**, une borne inférieure de  $\log p_{\theta}(x)$ .





# Preuve mathématique de la borne inf

$$\log p_{\theta}(x) = \log \int p_{\theta}(x|z)p(z)dz \quad (1)$$

$$= \log \int q_{\phi}(z|x) \frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)} dz \quad (2)$$

$$= \log E_{q_{\phi}(z|x)} \left[ \frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)} \right] \quad (3)$$

$$\geq E_{q_{\phi}(z|x)} \left[ \log \frac{p_{\theta}(x|z)p(z)}{q_{\phi}(z|x)} \right] \quad (4)$$

$$= E_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - D_{\text{KL}}(q_{\phi}(z|x) \| p(z)) \quad (5)$$

$$= \mathcal{L}(x) \quad (6)$$

# Objectif d'apprentissage: Maximiser l'ELBO

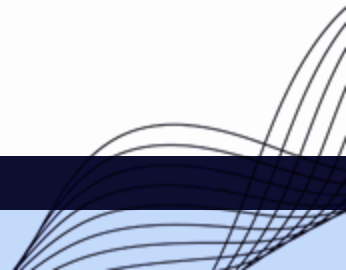
Définition de l'**ELBO** (Evidence Lower Bound):

$$\log p_{\theta}(x) \geq \mathcal{L}(x)$$

Où:

$$\mathcal{L}(x) = E_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] - D_{\text{KL}}(q_{\phi}(z|x) || p(z))$$

**Objectif d'apprentissage:** Maximiser  $\mathcal{L}(x)$  au lieu d'un vraisemblance intractable



# Interprétation intuitive de l'ELBO

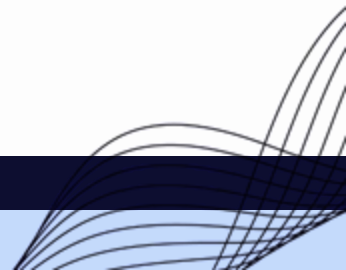
$$\mathcal{L}(x) = E_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] - D_{\text{KL}}(q_{\phi}(z|x) || p(z))$$

## Terme de reconstruction (1):

- Mesure à quel point le décodeur peut reconstruire  $x$  à partir de  $z$
- Encourage le décodeur à reconstruire des données réalistes

## Terme de régularisation (2):

- Force la distribution latente à rester proche du prior
- Evite un espace chaotique



# 04

## Entrainement pratique d'un VAE



# Rendre l'apprentissage possible



$$z \sim q_{\phi}(z|x) = \mathcal{N}(\mu_{\phi}(x), \sigma_{\phi}^2(x)I)$$

**Solution:** Trick de reparamétrisation

# Trick de reparamétrisation

$$z = \mu_{\phi}(x) + \sigma_{\phi}(x) \cdot \epsilon$$

où  $\epsilon \sim \mathcal{N}(0, I)$  est une variable aléatoire indépendante du réseau.

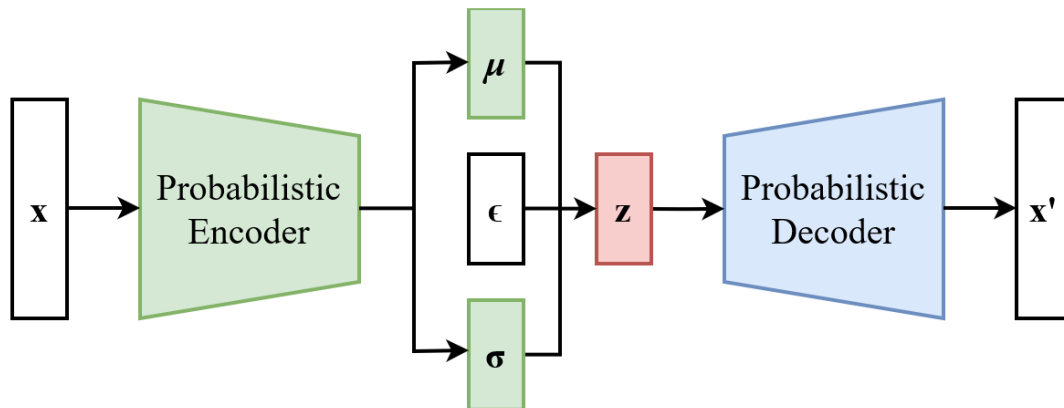
Grâce à cette reformulation :

- $\mu_{\phi}(x)$  et  $\sigma_{\phi}(x)$  restent entièrement différentiables.
- L'aléa est déplacé sur  $\epsilon$ , qui ne dépend pas des paramètres du modèle.
- On peut maintenant **rétropropager le gradient** à travers  $\mu_{\phi}(x)$  et  $\sigma_{\phi}(x)$  sans problème.



# Trick de reparamétrisation

$$z = \mu_{\phi}(x) + \sigma_{\phi}(x) \cdot \epsilon$$



# Entraînement pratique d'un VAE

- On choisit un mini-batch de données  $x$ .
- L'encodeur génère les paramètres  $\mu_\phi(x)$  et  $\sigma_\phi(x)$ .
- On échantillonne  $z$  via le **trick de reparamétrisation** :

$$z = \mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

- Le décodeur génère  $\hat{x}$  à partir de  $z$ , selon  $p_\theta(x|z)$ .
- On calcule la **perte du VAE**, qui est l'opposée de l'ELBO :

$$\mathcal{L}_{\text{VAE}} = -E_{q_\phi(z|x)}[\log p_\theta(x|z)] + D_{\text{KL}}(q_\phi(z|x) || p(z))$$

- On met à jour les paramètres  $\phi$  et  $\theta$  via la descente de gradient.



---

**Algorithm 1** Entraînement d'un Autoencodeur Variationnel (VAE)

---

```
1: Entrée: Données  $\{x_i\}$ , dimension latente  $d$ , taux d'apprentissage  $\eta$ , nombre  
   d'époques  $T$ , taille de batch  $B$   
2: Initialisation: Paramètres des réseaux  $\theta$  (encodeur) et  $\phi$  (décodeur)  
3: for  $t = 1$  to  $T$  do  
4:   for chaque mini-batch  $x_{batch}$  de taille  $B$  do  
5:     Encodage:  $(\mu, \log \sigma^2) \leftarrow \text{Encoder}(x_{batch})$   
6:     Reparamétrisation:  $z \leftarrow \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$   
7:     Décodage:  $\hat{x} \leftarrow \text{Decoder}(z)$   
8:     Calcul de la perte:  
9:        $\mathcal{L}_{rec} \leftarrow \|x_{batch} - \hat{x}\|^2$  (ou entropie croisée)  
10:       $\mathcal{L}_{KL} \leftarrow 0.5 \sum (\sigma^2 + \mu^2 - 1 - \log \sigma^2)$   
11:       $\mathcal{L} \leftarrow \mathcal{L}_{rec} + \mathcal{L}_{KL}$   
12:     Mise à jour des poids:  $\theta, \phi \leftarrow \theta, \phi - \eta \nabla_{\theta, \phi} \mathcal{L}$   
13:   end for  
14: end for  
15: Génération de nouvelles données:  
16: Échantillonner  $z \sim \mathcal{N}(0, I)$   
17: Générer  $\hat{x} \leftarrow \text{Decoder}(z)$ 
```

---

# 05

## Applications pratiques, limite et perceptives



# Application Pratiques

## Compression/Décompression et Génération d'Images

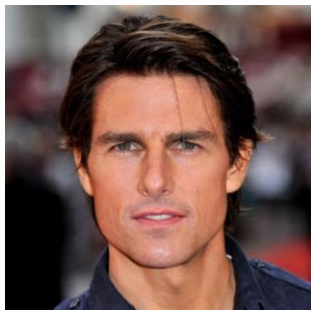
Will Smith



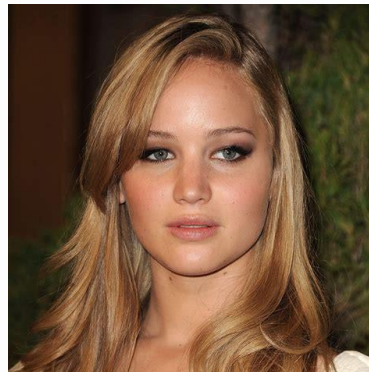
Tom Hanks



Tom Cruise



Jennifer Lawrence

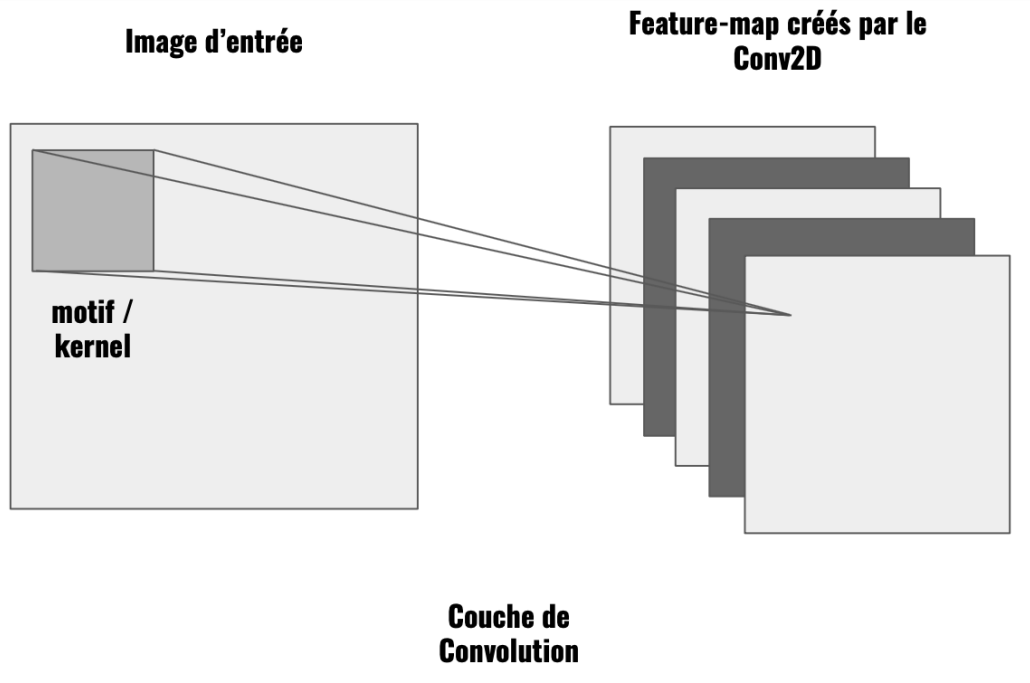


Angelina Jolie



# Application Pratiques

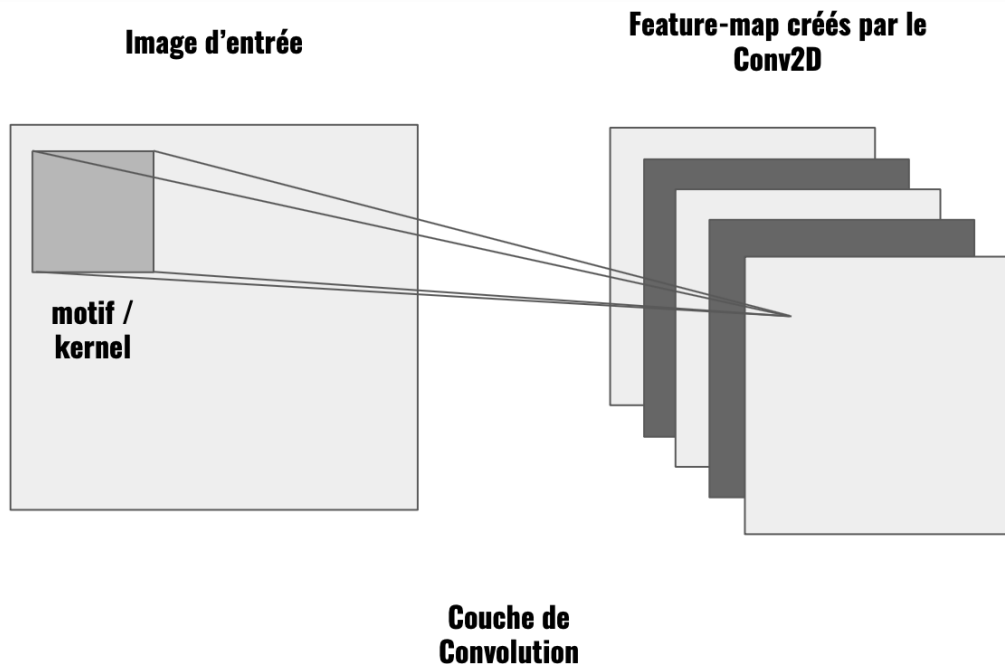
## Structure de Réseaux de neurones : Réseaux CNN



- Elle réduit la **dimension de l'image**
- Elle augmente la **profondeur de l'image** en sortie (nombre de canaux)
- Elle préserve la **structure spatiale** des objets

# Application Pratiques

## Structure de Réseaux de neurones : Réseaux CNN



Pourquoi utiliser un réseaux CNN pour un VAE ?

- **Extraction efficace des motifs visuels** : les convolutions détectent bords, textures et formes locales indispensables pour encoder une image de manière pertinente
- **Compression hiérarchique avec moins de paramètres** : elles réduisent progressivement la taille de l'image tout en conservant l'essentiel
- **Préserver la structure spatiale** : contrairement à un **flatten** direct, les Conv2D respectent l'organisation 2D de l'image, ce qui améliore la qualité du latent et des reconstructions

# Application Pratiques

## Structure Encodeur

Layer (type)	Output Shape	Param #	Connected to
input_layer_32 (InputLayer)	(None, 128, 128, 3)	0	-
conv2d_55 (Conv2D)	(None, 64, 64, 32)	896	input_layer_32[0]...
conv2d_56 (Conv2D)	(None, 32, 32, 64)	18,496	conv2d_55[0][0]
conv2d_57 (Conv2D)	(None, 16, 16, 128)	73,856	conv2d_56[0][0]
flatten_16 (Flatten)	(None, 32768)	0	conv2d_57[0][0]
dense_43 (Dense)	(None, 128)	4,194,432	flatten_16[0][0]
dense_44 (Dense)	(None, 64)	8,256	dense_43[0][0]
z_mean (Dense)	(None, 32)	2,080	dense_44[0][0]
z_log_var (Dense)	(None, 32)	2,080	dense_44[0][0]
sampling_16 (Sampling)	(None, 32)	0	z_mean[0][0], z_log_var[0][0]

## Structure Décodeur

Layer (type)	Output Shape	Param #
input_layer_33 (InputLayer)	(None, 32)	0
dense_45 (Dense)	(None, 32768)	1,081,344
reshape_16 (Reshape)	(None, 16, 16, 128)	0
conv2d_transpose_49 (Conv2DTranspose)	(None, 32, 32, 64)	73,792
conv2d_58 (Conv2D)	(None, 32, 32, 64)	36,928
conv2d_transpose_50 (Conv2DTranspose)	(None, 64, 64, 32)	18,464
conv2d_59 (Conv2D)	(None, 64, 64, 32)	9,248
conv2d_transpose_51 (Conv2DTranspose)	(None, 128, 128, 3)	867

# Application Pratiques

## Structure de Réseaux de neurones (Extrait du code)

```
# Paramètres globaux
input_shape = (128, 128, 3) # Dimensions des images
latent_dim = 32 # Taille de l'espace latent

# Définition de la couche d'échantillonnage
class Sampling(layers.Layer):
    """Utilise (z_mean, z_log_var) pour échantillonner un vecteur latent 'z'."""
    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.random.normal(shape=(batch, dim)) # Bruit aléatoire
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon # Réparamétrisation

# Construction de l'encodeur
encoder_inputs = keras.Input(shape=input_shape)
x = layers.Conv2D(32, 3, activation="relu", strides=2, padding="same")(encoder_inputs)
x = layers.Conv2D(64, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Conv2D(128, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Flatten()(x)
x = layers.Dense(128, activation="relu")(x)
x = layers.Dense(64, activation="relu")(x)
z_mean = layers.Dense(latent_dim, name="z_mean")(x)
z_log_var = layers.Dense(latent_dim, name="z_log_var")(x)
z = Sampling()([z_mean, z_log_var])
encoder = keras.Model(encoder_inputs, [z_mean, z_log_var, z], name="encoder")
encoder.summary()

# Construction du décodeur
latent_inputs = keras.Input(shape=(latent_dim,))
x = layers.Dense(16 * 16 * 128, activation="relu")(latent_inputs)
x = layers.Reshape((16, 16, 128))(x)

x = layers.Conv2DTranspose(64, 3, strides=2, padding="same", activation="relu")(x)
x = layers.Conv2D(64, 3, padding="same", activation="relu")(x)

x = layers.Conv2DTranspose(32, 3, strides=2, padding="same", activation="relu")(x)
x = layers.Conv2D(32, 3, padding="same", activation="relu")(x)

x = layers.Conv2DTranspose(3, 3, strides=2, padding="same", activation="sigmoid")(x)

# Décodeur final
decoder = keras.Model(latent_inputs, x, name="decoder")
decoder.summary()
```

```
class VAE(keras.Model): # VAE hérite donc de keras.Model (pemet d'utiliser des fonctionnalités comme .fit ou .compile)
    def __init__(self, encoder, decoder, **kwargs):
        super().__init__(**kwargs) # Appelle le constructeur de keras.Model
        self.encoder = encoder # l'encodeur
        self.decoder = decoder # le décodeur
        self.total_loss_tracker = keras.metrics.Mean(name="total_loss") #perte total
        self.reconstruction_loss_tracker = keras.metrics.Mean(
            name="reconstruction_loss"
        ) # perte de reconstruction
        self.kl_loss_tracker = keras.metrics.Mean(name="kl_loss") # Divergence KL

@property # Permet d'accéder aux métriques comme une variable (vae.metrics)
def metrics(self):
    return [
        self.total_loss_tracker,
        self.reconstruction_loss_tracker,
        self.kl_loss_tracker,
    ]

def train_step(self, data): # Remplace le .fit() de Keras
    with tf.GradientTape() as tape:
        z_mean, z_log_var, z = self.encoder(data) #encodeur transforme l'image en un espace latent (loi normale)
        reconstruction = self.decoder(z) #décodeur l'image
        reconstruction_loss = ops.mean(ops.sum(ops.abs(data - reconstruction),axis=(1, 2, 3)))
        #reconstruction_loss = ops.mean(ops.sum(keras.losses.binary_crossentropy(data, reconstruction),axis=(1, 2,)))

        # compare les pixels entre l'image d'origine et reconstruite
        kl_loss = -0.5 * (1 + z_log_var - ops.square(z_mean) - ops.exp(z_log_var)) # Divergence KL
        kl_loss = ops.mean(ops.sum(kl_loss, axis=1))
        total_loss = reconstruction_loss + kl_loss

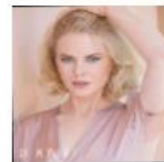
    grads = tape.gradient(total_loss, self.trainable_weights) # calcul les gradients
    self.optimizer.apply_gradients(zip(grads, self.trainable_weights)) # met à jour les poids
    self.total_loss_tracker.update_state(total_loss)
    self.reconstruction_loss_tracker.update_state(reconstruction_loss)
    self.kl_loss_tracker.update_state(kl_loss)

    return {
        "loss": self.total_loss_tracker.result(),
        "reconstruction_loss": self.reconstruction_loss_tracker.result(),
        "kl_loss": self.kl_loss_tracker.result(),
    }
```

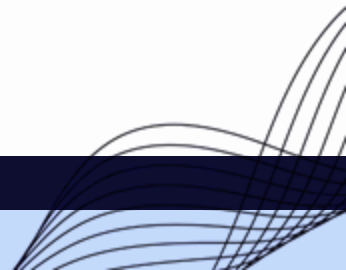
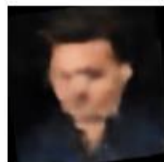
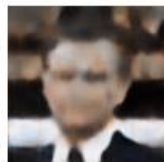
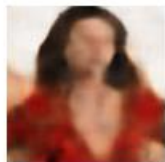
# Application Pratiques

## Résultats Compression/Décompression

Originales



Reconstructions

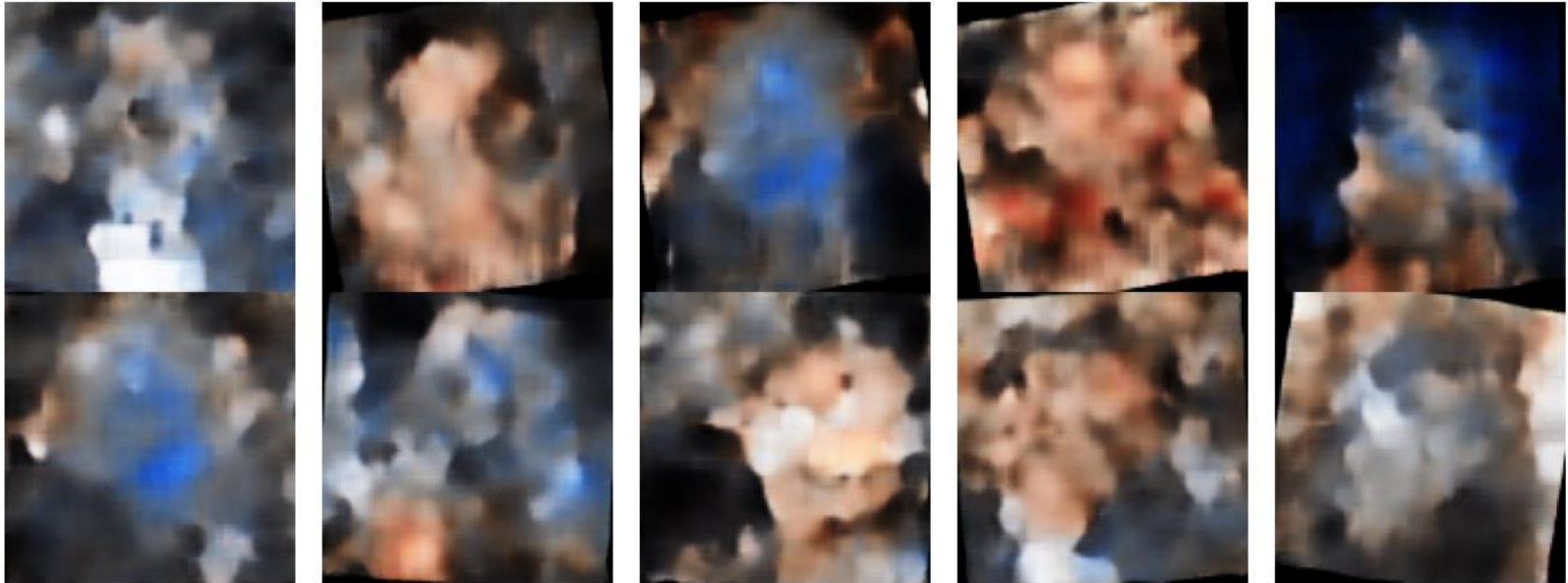




# Application Pratiques

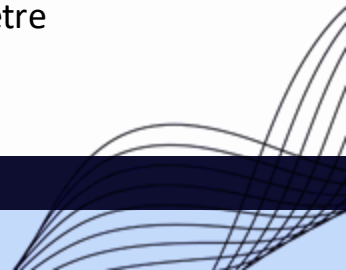
## Résultats Générations d'images

10 images générées par le VAE



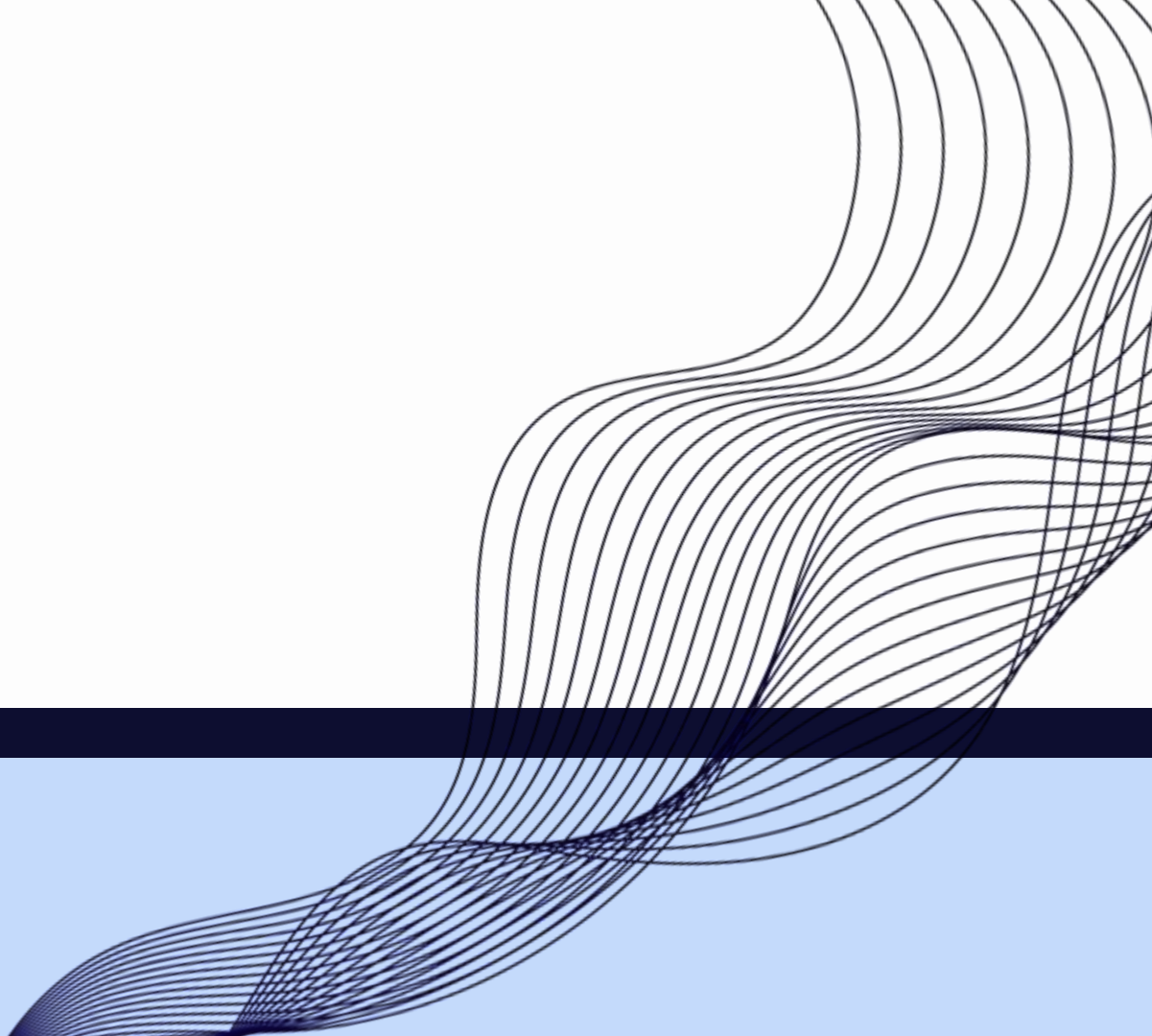
# Limitations

- Reconstruction floues
  - > Les VAE ont tendance à produire des images "moyennes" et peu nettes
- L'espace latent peu interprétable
  - > Le vecteur latent n'est pas toujours structuré de manière claire ni exploitable directement
- Compris reconstruction / régularisation
  - > La divergence KL peut nuire à la qualité des reconstructions si elle est trop forte
- Moins performants que les GAN pour la génération réaliste
  - > Les GAN produisent souvent des images plus nettes et plus naturelles
- Hypothèse forte sur la distribution latente
  - > Le VAE suppose que les variables latentes suivent une loi normale multivariée, ce qui peut être limitant



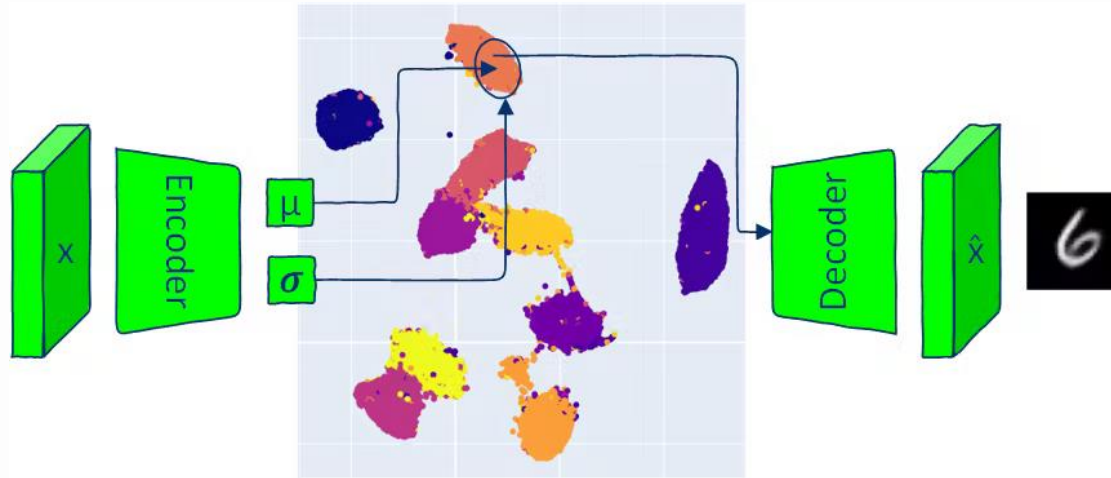
**06**

**Conclusion**



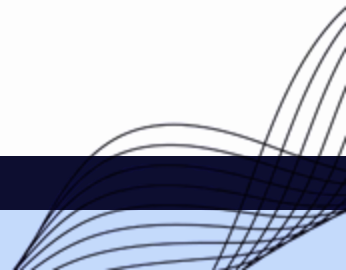
# Conclusions sur les VAE

- Approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données



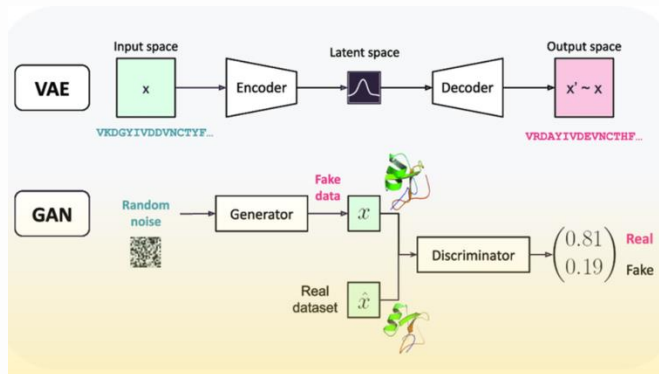
# Conclusions sur les VAE

- Approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données
- Optimisation de l'ELBO et au trick de reparamétrisation: entraînement efficace et structuration utile de l'espace latent



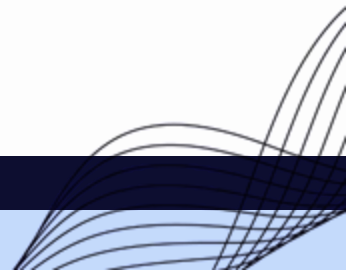
# Conclusions sur les VAE

- Approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données
- Optimisation de l'ELBO et au trick de reparamétrisation: entraînement efficace et structuration utile de l'espace latent
- **Mais:** production de résultats pas autant probants que d'autres modèles génératifs comme les GANs



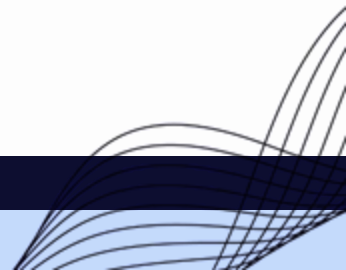
# Conclusions sur les VAE

- Approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données
- Optimisation de l'ELBO et au trick de reparamétrisation: entraînement efficace et structuration utile de l'espace latent
- **Mais:** production de résultats pas autant probants que d'autres modèles génératifs comme les GANs
- Outil fondamental en IA avec des applications variées en vision par ordinateur, biologie, médecine et traitement du signal



# Conclusions sur les VAE

- Approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données
- Optimisation de l'ELBO et au trick de reparamétrisation: entraînement efficace et structuration utile de l'espace latent
- **Mais:** production de résultats pas autant probants que d'autres modèles génératifs comme les GANs
- Outil fondamental en IA avec des applications variées en vision par ordinateur, biologie, médecine et traitement du signal
- Perspectives amélioration de la qualité des générations en combinant les VAE avec d'autres architectures, comme les VAE- GANs ou les VQVAE





# Conclusions sur les VAE

- Approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données
- Optimisation de l'ELBO et au trick de reparamétrisation: entraînement efficace et structuration utile de l'espace latent
- **Mais:** production de résultats pas autant probants que d'autres modèles génératifs comme les GANs
- Outil fondamental en IA avec des applications variées en vision par ordinateur, biologie, médecine et traitement du signal
- Perspectives amélioration de la qualité des générations en combinant les VAE avec d'autres architectures, comme les VAE- GANs ou les VQVAE
- **Finalement:** essentiel pour les modèles génératifs, et leur compréhension est un atout majeur pour quiconque s'intéresse au deep learning et à l'intelligence artificielle.

# Nous vous remercions pour votre attention.

Pierre Dobeli, Camille Davoine, Mathieu Faessel

