Les Auto-encodeurs Variationnel

Pierre Dobeli, Camille Davoine, Mathieu Faessel

March 2025

1 Les autoencodeurs : une première approche de l'apprentissage non supervisé

1.1 Pourquoi utiliser des autoencodeurs ?

Les **autoencodeurs** sont des modèles d'apprentissage automatique utilisés pour apprendre une **représentation compacte** des données. The main objective of the study is to transform a given entrée x into a compressible version z, to be capable of reconstruire x from z.

Les applications des autoencodeurs sont nombreuses :

- **Réduction de dimension** : Apprendre une représentation de moindre dimension sans supervision.
- **Compression de données** : Transformer des données volumineuses en une version plus compacte.
- **Détection d'anomalies** : Apprendre la structure normale des données et détecter celles qui s'en écartent.
- **Modèles génératifs** : Générer de nouvelles données en échantillonnant dans un espace latent bien structuré.

1.2 Comment fonctionne un autoencodeur ?

Un **autoencodeur classique** est un réseau de neurones constitué de deux parties principales :

- **L'encodeur** : prend une donnée d'entrée x et l'encode dans une représentation plus compacte z.
- **Le décodeur** : reconstruit la donnée originale x' à partir de z.

Mathématiquement, cela revient à apprendre une transformation :

$$z = f_\phi(x)$$

où f_ϕ est l'encodeur paramétré par ϕ, et

$$x' = g_\theta(z)$$

où g_θ est le décodeur paramétré par θ.
L'objectif de l'autoencodeur est de minimiser l'erreur de reconstruction :

$$\mathcal{L}_{rec} = ||x - x'||^2$$

1.3 L'espace latent : une représentation interne des données

Lorsqu'un autoencodeur transforme une donnée x en z, il crée une **représentation interne** des données appelée **espace latent**. Cet espace contient une version plus compacte et plus informative des données d'origine.

Exemple intuitif : Imaginons que nous ayons un ensemble de photos de visages en haute résolution (1000x1000 pixels). Chaque image contient 1 million de pixels, ce qui représente une énorme quantité d'informations ! Pourtant, tous les visages partagent des caractéristiques communes (yeux, nez, bouche, etc.).

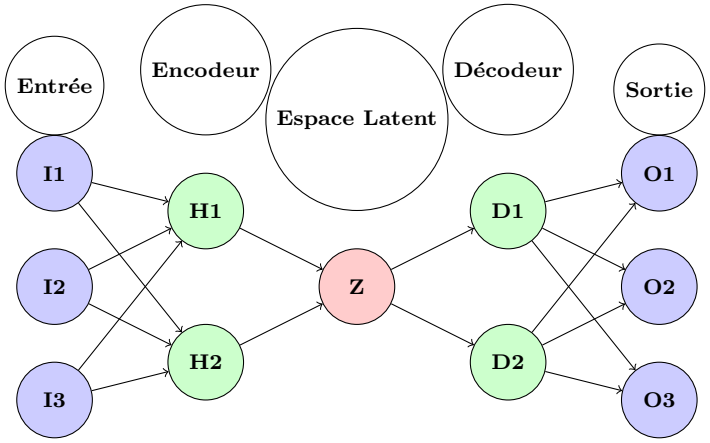
L'objectif d'un modèle d'apprentissage est de **trouver un espace plus compact où chaque visage est décrit par quelques variables seulement** (par exemple, un vecteur de taille 10 représentant les traits distinctifs de la personne). Cet espace réduit, où les images sont représentées par un petit vecteur z, est ce qu'on appelle un **espace latent**.

Pourquoi cet espace est utile ?

- Il permet de **résumer des données complexes** en quelques dimensions essentielles.
- Il sert de **base pour la génération de nouvelles données** : on peut manipuler ou échantillonner des points z pour générer de nouvelles instances.
- Une bonne structuration de l'espace latent permet d'organiser les données de manière plus pertinente.

1.4 Architecture d'un autoencodeur classique

Un autoencodeur suit une structure symétrique, avec une phase d'encodage, une couche latente, puis une phase de décodage. Voici un schéma illustrant cette architecture :



1.5 Limites des autoencodeurs classiques

Les autoencodeurs classiques ont plusieurs limitations :

- **L'espace latent n'est pas structuré** : Les points z sont placés arbitrairement dans l'espace latent, sans suivre de distribution particulière. Cela signifie que deux données similaires peuvent être envoyées dans des régions très différentes de l'espace latent. L'espace latent n'est donc **pas organisé de manière utile**.
- **Difficile de générer de nouvelles données** : Puisque l'autoencodeur encode chaque x en un z, mais sans contrainte sur la distribution des z, il est **impossible de savoir où prendre un nouveau** z pour générer une nouvelle donnée réaliste. Si on prend un z aléatoire, il peut tomber dans une zone non explorée par l'encodeur, et le décodeur risque de générer du bruit incohérent.
- **Risque de sur-apprentissage** : Le modèle peut apprendre à copier les données sans apprendre une vraie structure sous-jacente.

Les Autoencodeurs Variationnels (VAE) : une solution aux limites des autoencodeurs classiques

Contrairement aux autoencodeurs classiques, les **Variational Autoencoders (VAE)** imposent une structure à l'espace latent. Nous allons maintenant détailler la construction et l'entraînement des VAE.

2 Les Autoencodeurs Variationnels (VAE)

2.1 Pourquoi une approche probabiliste ?

Dans un autoencodeur classique, chaque donnée x est encodée en un point unique z. Cependant, cette approche pose un problème : si l'on perturbe

légèrement x, son encodage z peut changer de manière imprévisible, rendant l'espace latent difficilement exploitable.

L'idée du **Variational Autoencoder (VAE)** est d'adopter une **approche probabiliste** : au lieu d'associer un seul point z à chaque x, on cherche une **distribution** $q_\phi(z|x)$ qui capture l'incertitude sur z.

L'objectif est alors de **modéliser un espace latent structuré** en imposant une distribution prior sur z, généralement une loi normale $p(z) = \mathcal{N}(0, I)$.

2.2 Première approche: Formulation bayésienne du VAE

L'objectif du VAE est d'apprendre un modèle probabiliste où chaque donnée x est associée à une variable latente z.

Nous supposons que les données sont générées selon un processus probabiliste défini par :

- $p(z)$, la distribution **a priori** sur l'espace latent z, souvent choisie comme une loi normale standard $\mathcal{N}(0, I)$.
- $p_\theta(x|z)$, la distribution **conditionnelle** décrivant comment x est généré à partir de z.
- $p(x)$, la **vraisemblance marginale**, qui correspond à la probabilité d'observer une donnée x quelle que soit la valeur de z.

En appliquant la règle de Bayes, la distribution **a posteriori** de z sachant x s'écrit :

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

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 :

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

2.3 Pourquoi l'inférence exacte est-elle difficile ?

Le problème principal est que la vraisemblance marginale $p_\theta(x)$ est souvent **intractable** :

- L'intégrale $\int p_\theta(x|z)p(z)dz$ est **impossible à calculer exactement** lorsque z est de grande dimension.

- Cela rend le calcul de $p_\theta(z|x)$ (via Bayes) **inexploitable en pratique**.

Pour contourner ce problème, nous utilisons une **approximation variationnelle** : Plutôt que de calculer directement $p_\theta(z|x)$, nous introduisons une distribution auxiliaire $q_\phi(z|x)$ qui approxime cette postérieure.

2.4 Approche variationnelle : maximisation de l'ELBO

Plutôt que de calculer directement $p(x)$, nous utilisons une approche d'**inférence variationnelle**. L'idée est d'approximer la distribution **a posteriori** $p(z|x)$, qui est difficile à calculer, par une distribution $q_\phi(z|x)$, plus simple à manipuler.

Nous utilisons alors une **borne inférieure** (Evidence Lower Bound, ELBO) qui s'écrit :

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

où :

- $D_{\mathrm{KL}}(q_\phi(z|x)||p_\theta(z|x))$ 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 $\log p_\theta(x)$.

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|z)p(z)dz && (1)\\
&= \log \int q_\phi(z|x)\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}dz && (2)\\
&= \log E_{q_\phi(z|x)}\left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] && (3)\\
&\geq E_{q_\phi(z|x)}\left[\log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] && (4)\\
&= E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p(z)) && (5)\\
&= \mathcal{L}(x) && (6)
\end{aligned}$$

Nous avons donc :

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

L'objectif revient alors à textbfmaximiser l'ELBO, ce qui revient à maximiser :

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

2.5 Interprétation intuitive de l'ELBO

L'ELBO peut être vue comme la somme de deux termes :

- **Terme de reconstruction** $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$: Il mesure à quel point le décodeur peut reconstruire x à partir de z. Ce terme encourage le modèle à générer des données réalistes.
- **Terme de régularisation** $D_{\mathrm{KL}}(q_\phi(z|x)||p(z))$: Il force la distribution $q_\phi(z|x)$ à rester proche de la distribution prior $p(z) = \mathcal{N}(0, I)$. Cela évite que l'espace latent soit chaotique et améliore la génération de nouvelles données.

Maximiser l'ELBO permet donc d'apprendre :

- Une **bonne reconstruction** des données.
- Une **bonne structuration de l'espace latent** pour garantir la générabilité des nouvelles données.

Conclusion:

L'approche variationnelle permet de contourner le problème de l'inférence exacte en approximant $p(z|x)$ par $q_\phi(z|x)$. Maximiser l'ELBO est un moyen pratique et efficace d'optimiser un modèle probabiliste lorsque la vraisemblance marginale $p(x)$ est intractable.

Nous allons maintenant voir comment les VAE sont optimisés et comment ils permettent de générer des données de manière efficace.

2.6 Trick de reparamétrisation : rendre l'apprentissage possible

Un problème majeur dans l'entraînement des VAE est que l'encodeur produit une distribution $q_\phi(z|x)$ plutôt qu'un unique vecteur z. Cela implique que, pour chaque donnée x, l'encodeur **échantillonne** un z selon la loi $q_\phi(z|x)$, souvent une distribution gaussienne :

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

où :

- $\mu_\phi(x)$ est le vecteur des moyennes de z pour une entrée donnée.
- $\sigma^2_\phi(x)$ est le vecteur des variances.

2.6.1 Pourquoi l'échantillonnage pose-t-il problème ?

L'échantillonnage de z est une opération **stochastique** (aléatoire). Or, pour entraîner un réseau de neurones, nous avons besoin de calculer le **gradient** de la fonction de perte et de le propager dans tout le modèle.

Le problème est que **la fonction d'échantillonnage $z \sim q_\phi(z|x)$ casse le flux du gradient** :

- $\mu_\phi(x)$ et $\sigma^2_\phi(x)$ dépendent des poids du réseau.
- Mais z est échantillonné de manière non différentiable.
- Il est donc **impossible de rétropropager l'erreur** pour optimiser l'encodeur.

2.6.2 Solution : reformuler l'échantillonnage

Pour rendre l'échantillonnage différentiable, on utilise le **trick de reparamétrisation**, qui consiste à écrire :

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

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 ϵ, 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.

2.6.3 Pourquoi cela fonctionne-t-il ?

Cette reformulation fonctionne car elle permet de découpler :

- La **partie déterministe** ($\mu_\phi(x)$ et $\sigma_\phi(x)$).
- La **partie aléatoire** (ϵ).

Le réseau de neurones peut maintenant apprendre à ajuster $\mu_\phi(x)$ et $\sigma_\phi(x)$ pour optimiser l'ELBO, tout en gardant la génération aléatoire nécessaire pour explorer l'espace latent.

3 Entraînement du VAE

Une fois la structure du VAE définie et la borne ELBO introduite, il reste à optimiser notre modèle. L'entraînement d'un VAE repose sur la **maximisation de l'ELBO**, qui s'écrit :

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

Cette fonction objective contient deux termes :

- **Terme de reconstruction** $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$: Il mesure la capacité du décodeur à reconstruire x à partir de z.
- **Terme de régularisation** $D_{\mathrm{KL}}(q_\phi(z|x)||p(z))$: Il force la distribution $q_\phi(z|x)$ à rester proche de $p(z) = \mathcal{N}(0, I)$, garantissant un espace latent bien structuré.

3.1 Optimisation de l'ELBO

L'objectif de l'entraînement est de trouver les paramètres ϕ (encodeur) et θ (décodeur) qui maximisent l'ELBO. On utilise généralement la **descente de gradient stochastique** (SGD) ou une variante comme **Adam**, en procédant comme suit :

- On choisit un mini-batch de données x. (L'apprentissage se fait par batches successifs, et chaque passage complet sur le dataset s'appelle une époque.)
- 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 :

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

- On met à jour les paramètres ϕ et θ via la descente de gradient.

3.2 Pseudo-code de l'entraînement d'un VAE

L'entraînement d'un VAE peut être résumé par l'algorithme suivant :

3.3 Conclusion

L'entraînement d'un VAE repose sur la **maximisation de l'ELBO**, qui permet d'apprendre une représentation latente bien structurée. Grâce au **trick de reparamétrisation**, on peut rétropropager le gradient et optimiser le modèle efficacement avec la descente de gradient.

Dans la section suivante, nous allons explorer les applications pratiques des VAE et leurs avantages dans le cadre des modèles génératifs.

Algorithm 1 Entraînement d'un Autoencodeur Variationnel (VAE)

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

4 Applications et Conclusion

Les Variational Autoencoders (VAE) sont largement utilisés dans le domaine de l'apprentissage automatique, notamment pour la **génération de données réalistes**. Grâce à leur capacité à apprendre une **représentation structurée** des données, ils trouvent des applications dans plusieurs domaines.

4.1 Applications des VAE

Voici quelques cas d'utilisation notables des VAE :

4.1.1 Génération d'images

Les VAE sont capables de générer de nouvelles images réalistes après avoir été entraînés sur un dataset donné. Par exemple :

- Génération de **visages humains** après entraînement sur des bases comme CelebA.
- Synthèse d'**écritures manuscrites** après entraînement sur MNIST.
- Création d'objets réalistes (ex: chaussures, vêtements, maisons) après apprentissage sur des datasets spécialisés.

4.1.2 Interpolation et exploration de l'espace latent

Un VAE apprend un espace latent structuré où les données similaires sont proches les unes des autres. Cela permet de :

- Interpoler entre deux images en **faisant varier** z progressivement.
- Modifier des caractéristiques d'une image en naviguant dans l'espace latent (ex: rendre un visage plus souriant).

4.1.3 Débruitage et reconstruction de données

Les VAE peuvent être utilisés pour **restaurer des images dégradées** en exploitant leur capacité à capturer la structure sous-jacente des données. Exemples :

- Amélioration d'images floues ou bruitées.
- Restauration de photos anciennes.

4.1.4 Compression de données

Étant donné que l'espace latent est de plus faible dimension que l'espace d'origine, les VAE peuvent être utilisés comme une technique de compression. Ils permettent de stocker des informations dans un **format plus compact** tout en conservant les caractéristiques essentielles.

4.1.5 Génération de données dans des contextes scientifiques

Les VAE sont utilisés en :

- **Biologie** : génération de nouvelles molécules avec des propriétés spécifiques.
- **Médecine** : création de scans médicaux synthétiques pour l'entraînement de modèles.
- **Musique** : composition automatique de mélodies après entraînement sur un corpus musical.

4.2 Limitations des VAE

Bien que puissants, les VAE présentent certaines limites :

- **Qualité des images générées** : Les images produites par les VAE sont souvent plus floues que celles des GANs (Generative Adversarial Networks).
- **Contraintes sur l'espace latent** : La régularisation par la divergence KL peut limiter la diversité des données générées.
- **Difficulté à capturer des détails complexes** : Certains modèles plus avancés, comme les VAE hiérarchiques ou les VQVAE (Vector Quantized VAE), cherchent à améliorer cet aspect.

4.3 Conclusion

Les Variational Autoencoders (VAE) sont une approche puissante pour apprendre des représentations latentes structurées et générer de nouvelles données. Grâce à l'**optimisation de l'ELBO** et au **trick de reparamétrisation**, ils permettent un entraînement efficace et une structuration utile de l'espace latent.

Cependant, bien que performants, ils ne produisent pas toujours des résultats aussi nets que d'autres modèles génératifs comme les GANs. Toutefois, les VAE restent un outil fondamental en **intelligence artificielle**, avec des applications variées en vision par ordinateur, biologie, médecine et traitement du signal.

Perspectives : Les recherches récentes visent à améliorer la qualité des générations en combinant les VAE avec d'autres architectures, comme les VAE-GANs ou les VQVAE.

Les VAE constituent ainsi une **brique essentielle des modèles génératifs**, et leur compréhension est un atout majeur pour quiconque s'intéresse au deep learning et à l'intelligence artificielle.