Chapitre 13 : Réseaux de Neurones

26 février 2026

Plan du chapitre

Deux approches de l'apprentissage

L'utilisation d'un espace de redescription (SVM) et l'approche des réseaux de neurones s'opposent conceptuellement :

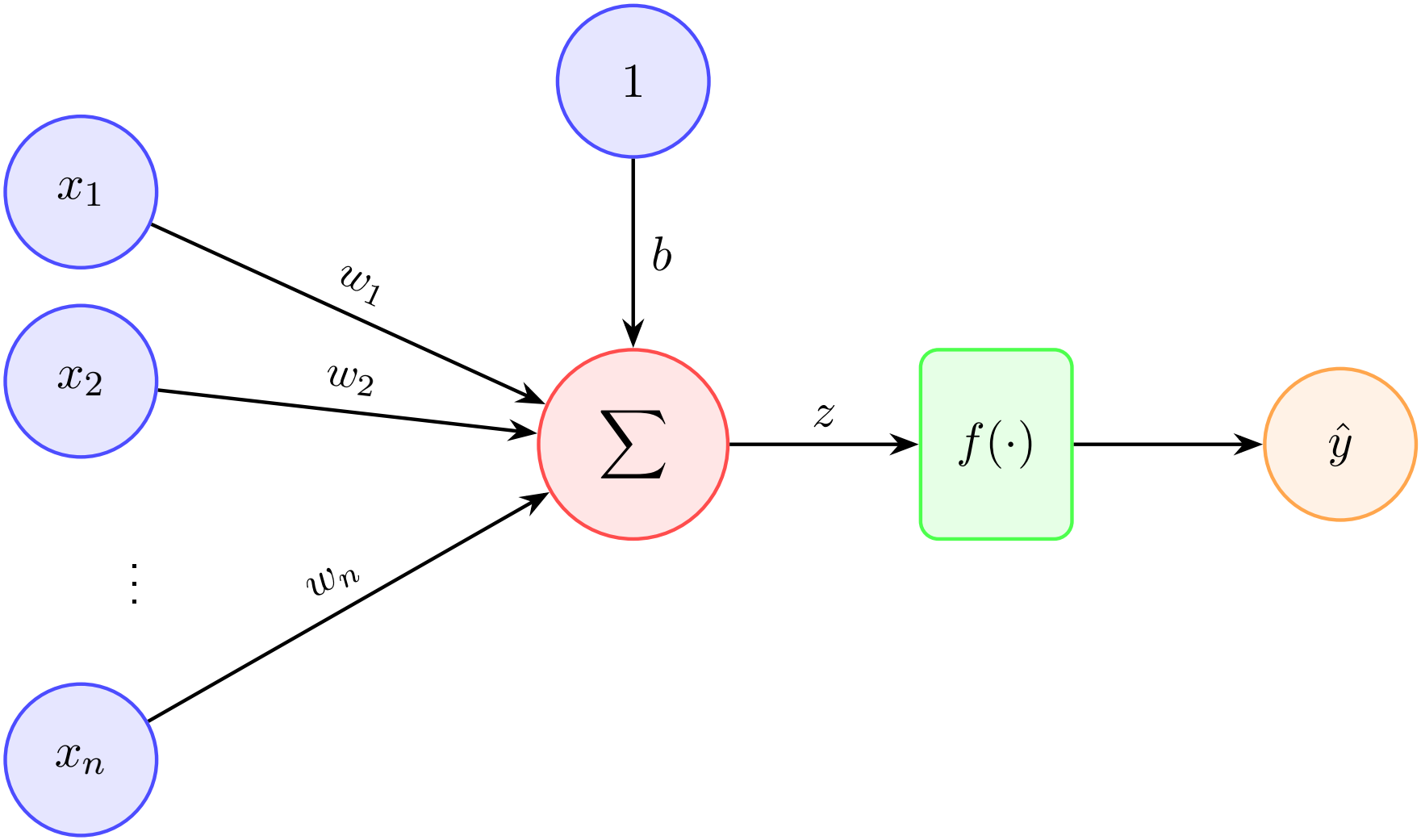
↑ L'approche classique (SVM à noyaux)

Ascendante : Choix *a priori* des caractéristiques. On augmente la dimension pour séparer linéairement les données dans un grand espace (redescription non-linéaire).

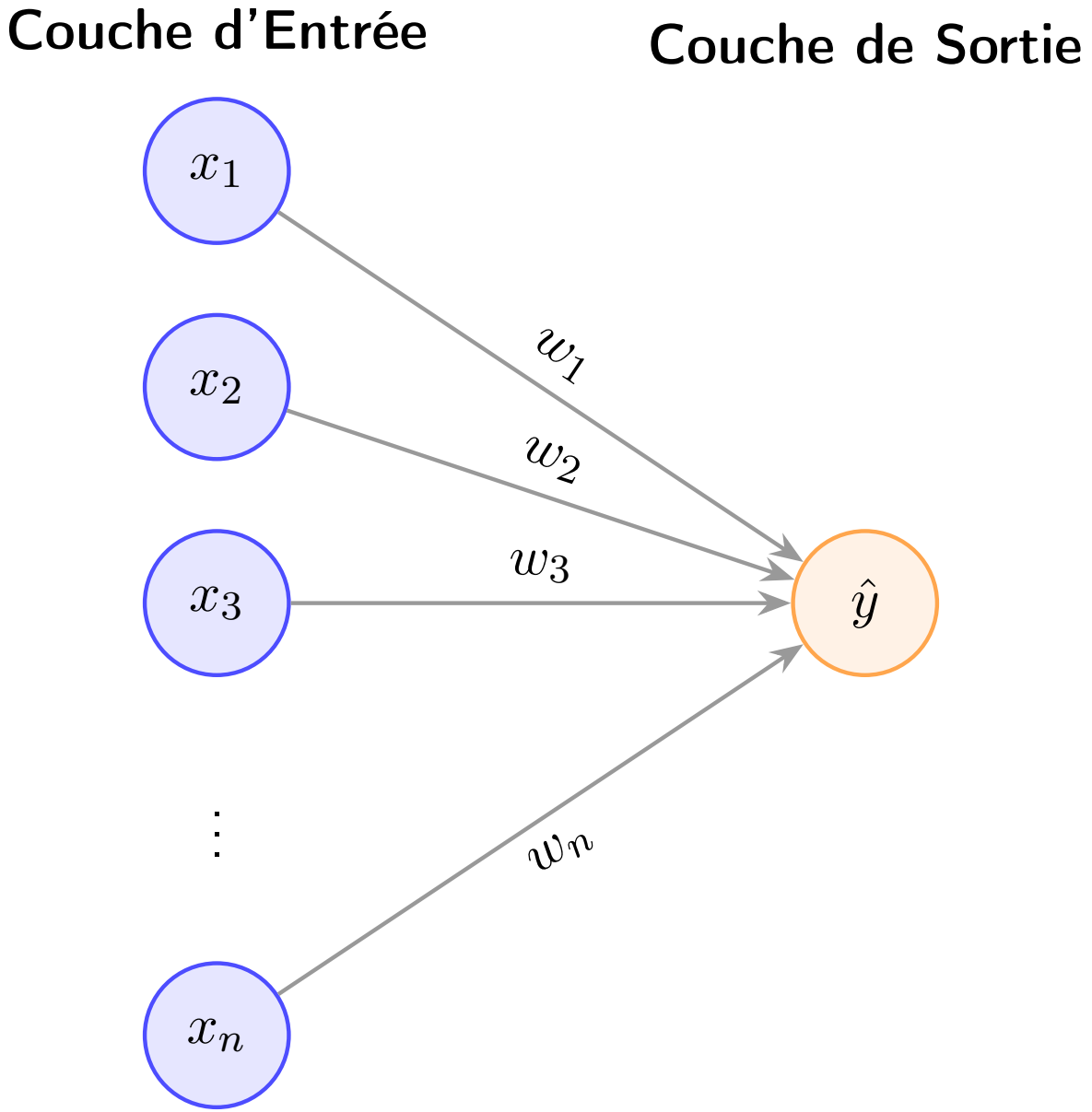
↓ Les Réseaux de Neurones

Descendante : Le réseau *apprend* la meilleure représentation. Il extrait lui-même les caractéristiques et réduit la dimension pour trouver une frontière dans un espace de représentation minimal.

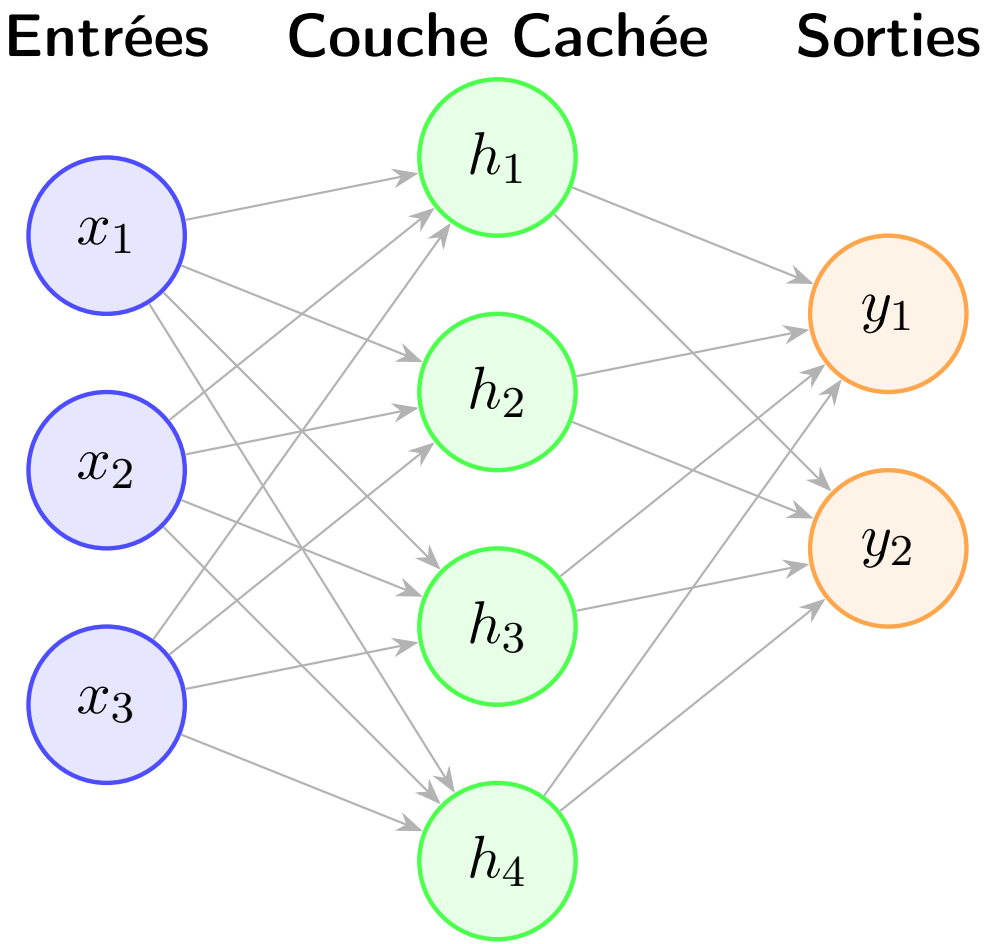
Neurone artificiel



Perceptron simple



Perceptron Multicouche (MLP)



Le Perceptron Multicouche (MLP)

Le perceptron simple est limité aux problèmes linéairement séparables (il échoue sur la fonction XOR). Le **MLP** (Multi-Layer Perceptron) résout cela via des **couches cachées**.

Pour une couche cachée de m neurones :

1. **Calcul de la couche cachée** : Pour chaque neurone j, on calcule :

$$h_j(\mathbf{x}) = \sigma\left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + b_j^{(1)}\right)$$

2. **Calcul de la sortie** :

$$f(\mathbf{x}) = \psi\left(\sum_{j=1}^{m} w_j^{(2)} h_j(\mathbf{x}) + b^{(2)}\right)$$

La fonction d'activation σ (Sigmoïde, ReLU) est **fondamentale**. Sans elle, le réseau se réduirait à une simple transformation linéaire !

Théorème d'Approximation Universelle

Pourquoi les réseaux de neurones sont-ils si puissants ?

Théorème (Cybenko 1989, Hornik 1991)

Soit σ une fonction d'activation continue et non constante. Pour toute fonction continue f sur un compact $K \subset \mathbb{R}^d$ et pour tout $\epsilon > 0$, il existe un réseau de neurones à **une seule couche cachée** avec N neurones tel que :

$$\sup_{\mathbf{x} \in K} |f(\mathbf{x}) - G(\mathbf{x})| < \epsilon$$

Lien avec la théorie VC : Le **biais** d'un réseau peut être rendu arbitrairement petit (il peut tout approximer). Mais augmenter N fait exploser la dimension VC, et donc la **variance**. Le défi du Deep Learning est d'optimiser ces paramètres sans surapprendre.

La Rétropropagation du Gradient (Backpropagation)

Entraîner le réseau consiste à minimiser une fonction de coût $\mathscr{L}$. Calculer le gradient des couches profondes se fait via la **règle de la chaîne**.

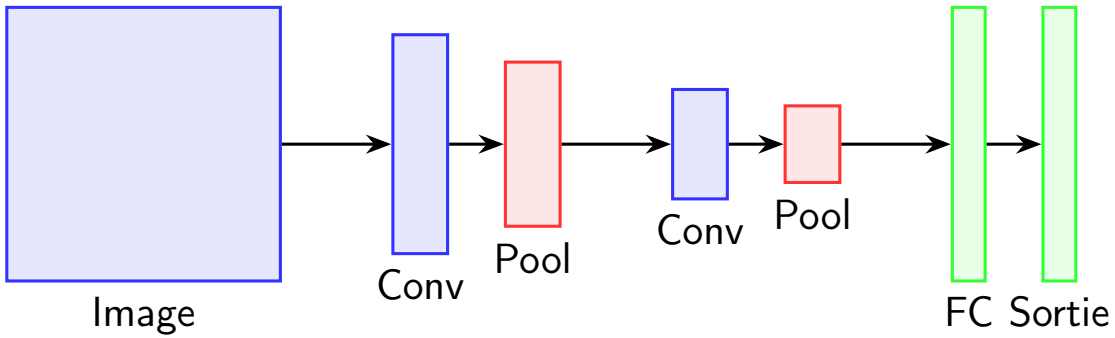
Exemple pour un poids de la première couche :

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \underbrace{\frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}}}_{\delta^{(2)}} \cdot \underbrace{\frac{\partial z^{(2)}}{\partial a_j^{(1)}}}_{w_j^{(2)}} \cdot \underbrace{\frac{\partial a_j^{(1)}}{\partial z_j^{(1)}}}_{\sigma'(z_j^{(1)})} \cdot \underbrace{\frac{\partial z_j^{(1)}}{\partial w_{ji}^{(1)}}}_{x_i}$$

L'algorithme en 2 étapes :

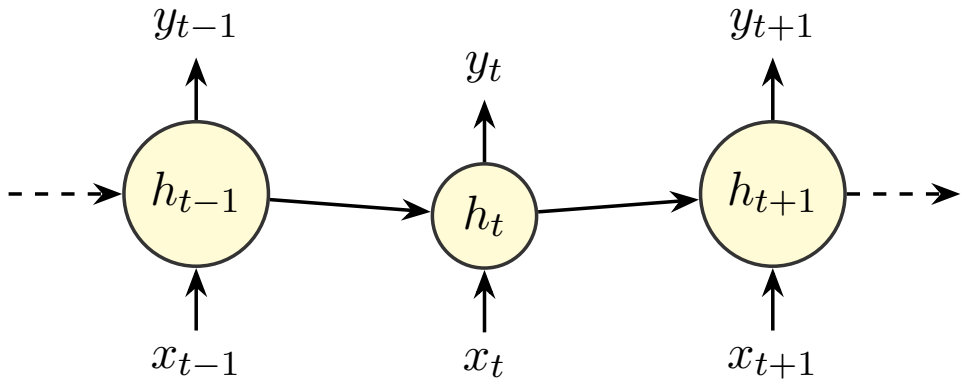
1. **Passe avant (Forward)** : Propagation de $\mathbf{x}$ jusqu'à la prédiction $\hat{y}$ et calcul de l'erreur E.
2. **Passe arrière (Backward)** : Rétropropagation de l'erreur de la sortie vers l'entrée pour calculer tous les gradients, puis mise à jour : $w \leftarrow w - \eta \frac{\partial E}{\partial w}$.

1. CNN (Réseaux de Neurones Convolutifs)



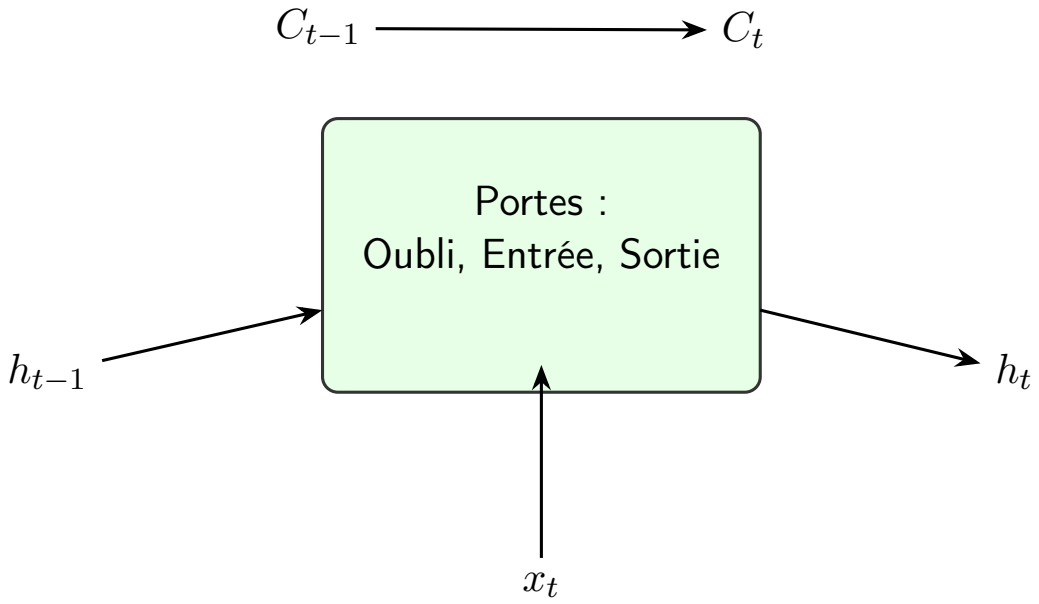
- **Données :** Grilles spatiales (Images 2D, Vidéos 3D, Spectrogrammes).
- **Tâches :** Classification d'images, détection d'objets, segmentation.
- **Forces :** Invariance aux translations, partage des poids (réduit drastiquement le nombre de paramètres), capture la hiérarchie spatiale (bords $\rightarrow$ formes $\rightarrow$ objets).
- **Limites :** Nécessite des entrées de taille fixe, perd le contexte global spatial absolu (sans ajout d'attention).

2. RNN (Réseaux de Neurones Récurrents)



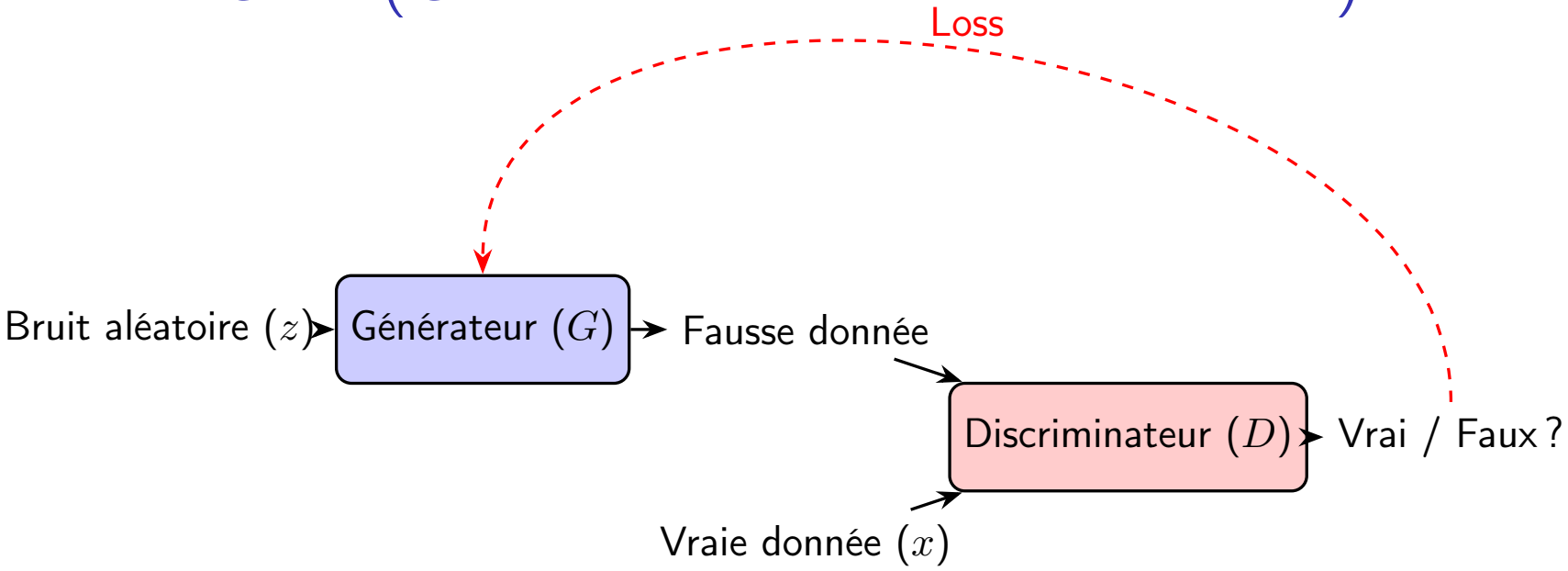
- **Données :** Séquences temporelles, Séries chronologiques, Texte.
- **Tâches :** Traduction, prédiction boursière, analyse de sentiment.
- **Forces :** Gère des entrées de taille variable, conserve une "mémoire" (état caché h_t) des étapes précédentes.
- **Limites :** Problème de la **disparition du gradient** (vanishing gradient) : oublie très vite les informations anciennes. Impossible à paralléliser.

3. LSTM (Long Short-Term Memory)



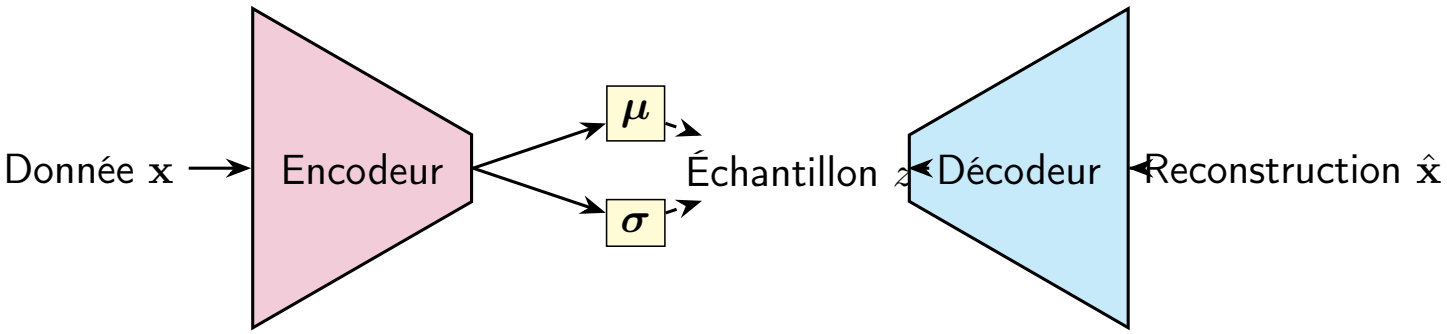
- **Données/Tâches :** Identiques aux RNN.
- **Le concept :** Une cellule complexe avec un flux d'information principal (C_t) protégé par des "portes" (forget, input, output gates) apprenant quoi garder ou oublier.
- **Forces :** Résout le problème de disparition du gradient des RNN. Capture d'excellentes dépendances à long terme.
- **Limites :** Très coûteux, séquentiel (pas de parallélisation), supplanté aujourd'hui par les Transformers en NLP.

4. GAN (Generative Adversarial Networks)



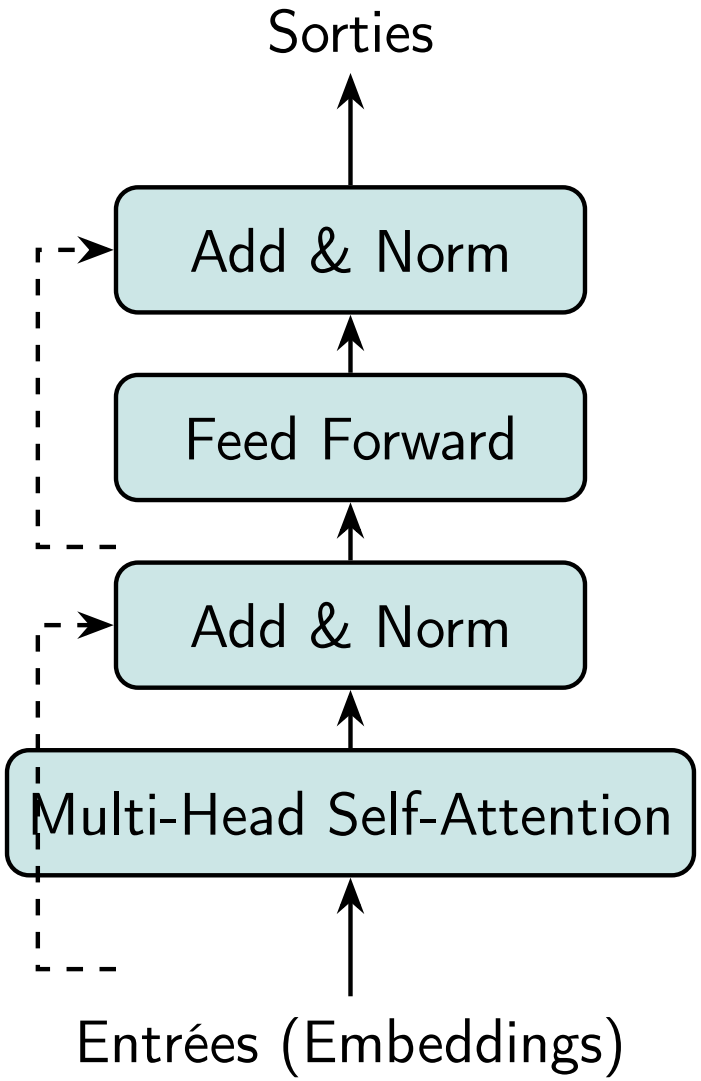
- **Données :** Principalement Images, Vidéos, Audio.
- **Tâches :** Génération de données photoréalistes, Deepfakes, Super-résolution.
- **Forces :** Qualité des images. N'exige pas de modéliser explicitement la densité de probabilité.
- **Limites :** Difficile à entraîner (instabilité minimax), problème d'effondrement de mode (*mode collapse*), pas d'espace latent structuré explicite.

5. VAE (Variational Autoencoders)



- **Données :** Images, Séries temporelles, Molécules.
- **Tâches :** Génération, Détection d'anomalies, Apprentissage de représentations.
- **Forces :** Base mathématique solide (Inférence bayésienne), espace latent continu et structuré (permet d'interpoler facilement entre deux images).
- **Limites :** A tendance à générer des images floues (comparé aux GANs ou aux modèles de Diffusion) à cause de la perte MSE/BCE utilisée pour la reconstruction.

6. Transformers



6. Transformers

- **Données :** Texte (LLMs), Images (ViT), Audio.
- **Tâches :** NLP (ChatGPT, Traduction), Vision par ordinateur.
- **Forces :** L'Attention (Self-Attention) permet au modèle de regarder **tout le contexte** d'un coup. Hautement parallélisable sur GPU (contrairement aux RNN).
- **Limites :** Complexité quadratique $O(N^2)$ par rapport à la taille de la séquence (gourmand en VRAM). Nécessite des volumes de données titanesques pour converger.