Apprentissage autosupervisé

Redha Moulla

Janvier 2026

Plan de la formation

- Principes de l'apprentissage auto-supervisé
- Apprentissage des représentations
- Apprentissage contrastif
- Explicabilité des modèles de deep learning

Principes de l'apprentissage autosupervisé

Machine learning

L'apprentissage automatique est une branche de l'intelligence artificielle qui consiste à doter les machines de la capacité d'apprendre à partir de données sans que celles-ci ne soient explicitement programmées pour exécuter des tâches spécifiques.

Le machine learning englobe classiquement deux grandes familles d'apprentissage :

- **Supervisé** : Les algorithmes apprennent à partir de données étiquetées pour faire des prédictions.
- **Non supervisé** : L'apprentissage est effectué sur des données non étiquetées pour trouver des structures cachées.

L'apprentissage supervisé

L'apprentissage supervisé consiste à apprendre un modèle qui associe une étiquette (*label*) à un ensemble de caractéristiques (*features*).

- **Inputs** : un jeu de données *annotées* pour entraîner le modèle.
  - Exemple : des textes (tweets, etc.) avec les *sentiment* associés, positifs ou négatifs.
- **Output** : une étiquette pour un point de donnée inconnu par le modèle.

L'apprentissage supervisé se décline lui-même en deux grandes familles :

- **La classification** : prédire une catégorie ou une classe.
  - Exemple : prédire l'étiquette d'une image (chat, chien, etc.), le sentiment associé à un texte, le centre d'intérêt d'un client à partir de ses commentaires, etc.
- **La régression** : prédire une valeur continue (un nombre réel typiquement).
  - Exemple : prédire le prix d'un appartement, la lifetime value d'un client, etc.

L'apprentissage non supervisé

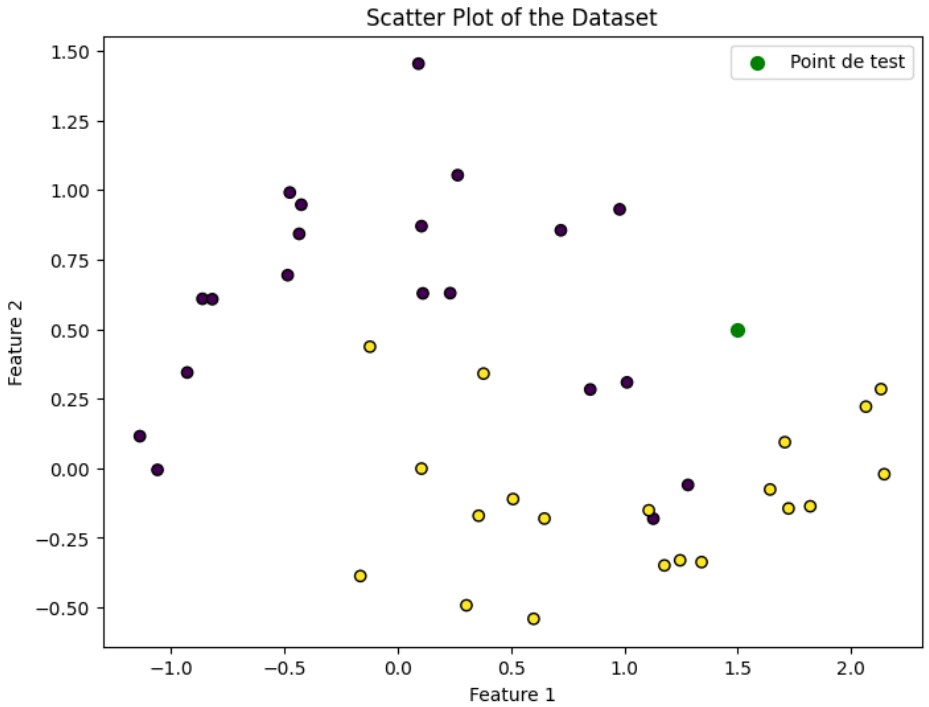
L'apprentissage non supervisé se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

Principales typologies de l'apprentissage non supervisé :

- **Clustering** : Regroupement de points de données similaires ensemble. Exemple : segmentation de marché, regroupement social. (Kmeans, DBSCAN, etc.).
- **Réduction de dimension** : Projeter les données sur une sous-espace de plus petite dimension que celui des caractéristiques originelles, notamment pour la visualisation (ACP, Autoencodeurs, etc.)
- **Détection d'anomalies** : Détecter des observations dont les caractéristiques sont inhabituelles par rapport à la majorité (Isolation forest, etc.).

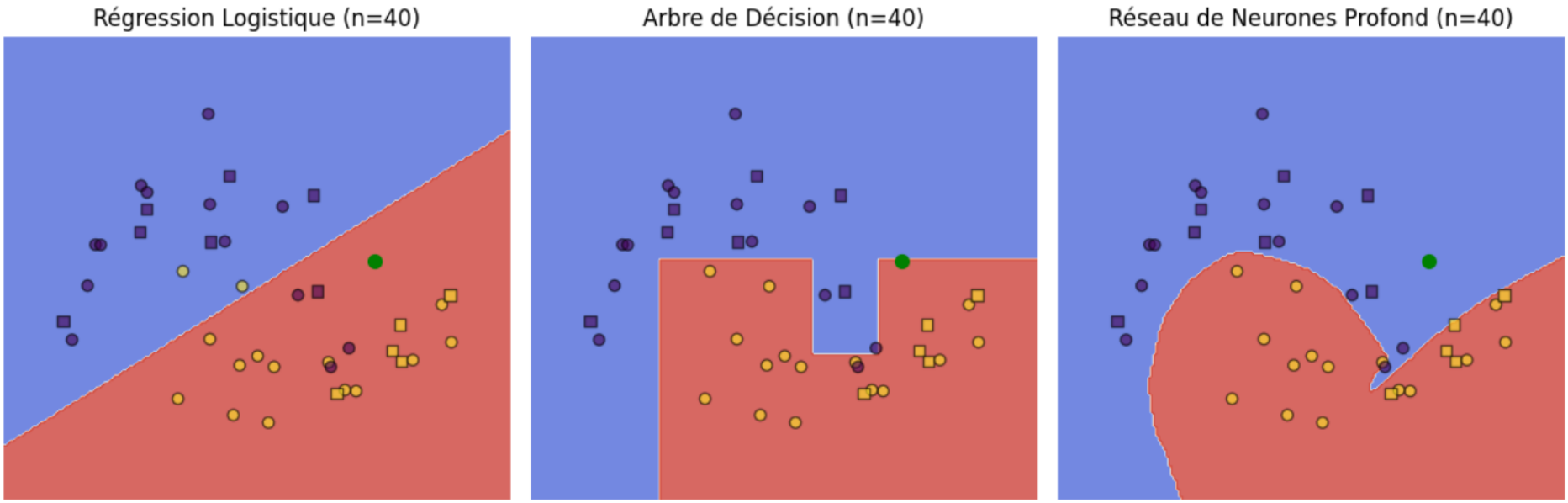
L'apprentissage supervisé comme un problème d'induction

- **Définition** : L'apprentissage supervisé consiste à apprendre une fonction f qui mappe les entrées X aux sorties y, à partir d'un ensemble d'exemples d'entraînement (X, y).
- **Induction** : Le modèle induit une règle générale à partir de données particulières, dans le but de généraliser à de nouvelles instances.
- **Problème de généralisation** : Comment garantir que le modèle apprend une règle qui s'applique à de nouvelles données ?



Une indétermination intrinsèque pour le choix du modèle

Il y a une infinité de manières d'induire un modèle à partir d'un échantillon de données d'entraînement.



Les limites structurelles de l'apprentissage supervisé

L'apprentissage supervisé constitue le paradigme historique du machine learning moderne. Son efficacité empirique masque toutefois des limites profondes, non accidentelles mais **structurelles**, liées à sa dépendance au label.

Dans l'apprentissage supervisé :

- l'apprentissage consiste à approximer une fonction entrée $\rightarrow$ sortie
- le réel est réduit à une consigne explicite

Le modèle n'apprend pas la structure du monde, mais à satisfaire une norme définie a priori.

Un apprentissage instrumental, non exploratoire

Le modèle supervisé apprend :

- ce qui est utile pour prédire le label
- pas ce qui est structurellement présent dans les données

Conséquence :

- absence d'exploration autonome
- pas de découverte de facteurs latents
- dépendance totale à la tâche définie

L'apprentissage est orienté vers la réponse, pas vers la compréhension.

Biais humains encapsulés dans les labels

Un label reflète :

- une culture
- une époque
- une norme sociale ou industrielle

Le modèle ne corrige pas ces biais :

- il les apprend
- il les amplifie
- il les généralise avec confiance

Le biais devient mathématiquement consolidé.

Le coût et le goulot d'étranglement de l'annotation

Les données brutes sont abondantes. Les labels sont :

- coûteux
- lents à produire
- dépendants d'experts humains

Dans de nombreux domaines (vision médicale, NLP métier, industrie), le véritable goulot d'étranglement n'est pas le calcul, mais la supervision.

L'échelle devient économiquement asymétrique.

Non supervisé et auto-supervisé : une confusion fréquente

Après avoir identifié les limites du supervisé, une question naturelle émerge :

Peut-on apprendre sans labels ?

Deux paradigmes sont souvent confondus :

- l'apprentissage non supervisé
- l'apprentissage auto-supervisé

Ils reposent pourtant sur des logiques profondément différentes.

Apprentissage non supervisé : principe général

L'apprentissage non supervisé vise à :

- découvrir des structures latentes
- sans objectif explicite de prédiction
- sans signal de supervision

Exemples classiques :

- PCA
- k-means
- modèles de mélanges

L'objectif est descriptif, pas prédictif. On va l'utiliser le plus souvent pour analyser les données et comprendre s'il y a des motifs intéressants.

Limites intrinsèques du non supervisé

Les méthodes non supervisées :

- dépendent fortement d'hypothèses statistiques
- imposent une forme a priori (linéarité, clusters sphériques, etc.)
- produisent des représentations peu transférables

Elles n'optimisent pas directement une notion d'utilité pour une tâche future.

Le non supervisé explore, mais sans direction fonctionnelle pour résoudre un problème.

Apprentissage auto-supervisé : un changement de cadre

L'apprentissage auto-supervisé introduit :

- un objectif explicite
- construit à partir des données elles-mêmes

Il repose sur des **tâches prétextes** :

- prédire une partie des données à partir d'une autre
- exploiter des relations internes

La supervision est endogène (interne aux données). Le modèle apprend des invariances (une structure du monde), pas seulement des statistiques globales.

Idée de l'apprentissage autosupervisé

How Much Information is the Machine Given during Learning?

Y. LeCun

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Tâche prétexte

Une tâche prétexte est :

- une tâche de prédiction auto-générée
- construite uniquement à partir des données brutes
- sans annotation humaine explicite

Le signal de supervision est extrait :

- des relations internes aux données
- de leur structure spatiale, temporelle ou sémantique

Le goulot d'étranglement n'est plus la labellisation des données, mais la puissance de calcul.

Tâche prétexte vs tâche downstream

Tâche prétexte :

- utilisée uniquement pendant le pré-entraînement
- jamais évaluée directement
- souvent sans intérêt applicatif

Tâche downstream :

- tâche finale d'intérêt
- supervisée ou faiblement supervisée
- mesure l'utilité des représentations

La tâche prétexte n'est jamais neutre

Toute tâche prétexte impose :

- une vision implicite du monde
- des hypothèses sur ce qui est invariant
- une notion de similarité

Elle encode un biais inductif.

L'auto-supervisé n'est pas sans hypothèses : il les déplace dans la conception de la tâche.

Une tâche prétexte mal conçue peut conduire à :

- des solutions triviales
- l'effondrement des représentations (feature collapse)

Typologies de tâches prétexte

Les tâches prétexte exploitent des régularités internes aux données :

- spatiales
- temporelles
- contextuelles
- multimodales

Elles ne sont pas universelles : elles dépendent fortement du type de données et du monde que l'on suppose.

Vision – Prédire une partie manquante

Autre famille centrale :

- inpainting
- prédiction de patches masqués
- Masked Autoencoders

Le modèle doit reconstruire une information absente à partir du contexte fourni : une tâche de prédiction généralement.

Hypothèse :

- le monde visuel est redondant et structuré

Vision – Prédire une transformation

Exemples de tâches prétexte en vision :

- prédire la rotation appliquée à une image
- détecter un flip horizontal
- identifier une permutation de patches

Hypothèse implicite :

- la structure sémantique est stable sous certaines transformations

Le modèle apprend la géométrie globale des objets.

Vision – Tâches contrastives

Principe :

- créer deux vues d'une même image
- rapprocher leurs représentations
- éloigner celles d'autres images

La tâche prétexte est implicite :

- reconnaître une identité sous transformations

On n'apprend pas une classe, mais une invariance.

NLP – Masquage et prédiction

En langage, la tâche prétexte est presque naturelle :

- masquer un mot ou un token
- le prédire à partir du contexte

Hypothèse :

- le sens est distribué dans le contexte

Le langage fournit spontanément sa propre supervision.

NLP – Prédiction séquentielle

Autre approche :

- prédire le prochain token
- modéliser la dynamique du langage

La tâche prétexte coïncide presque avec la tâche finale.

Le modèle apprend une représentation dynamique du monde linguistique.

Multimodal – Alignement de modalités

En multimodal :

- associer une image et un texte
- prédire si deux modalités correspondent

Hypothèse :

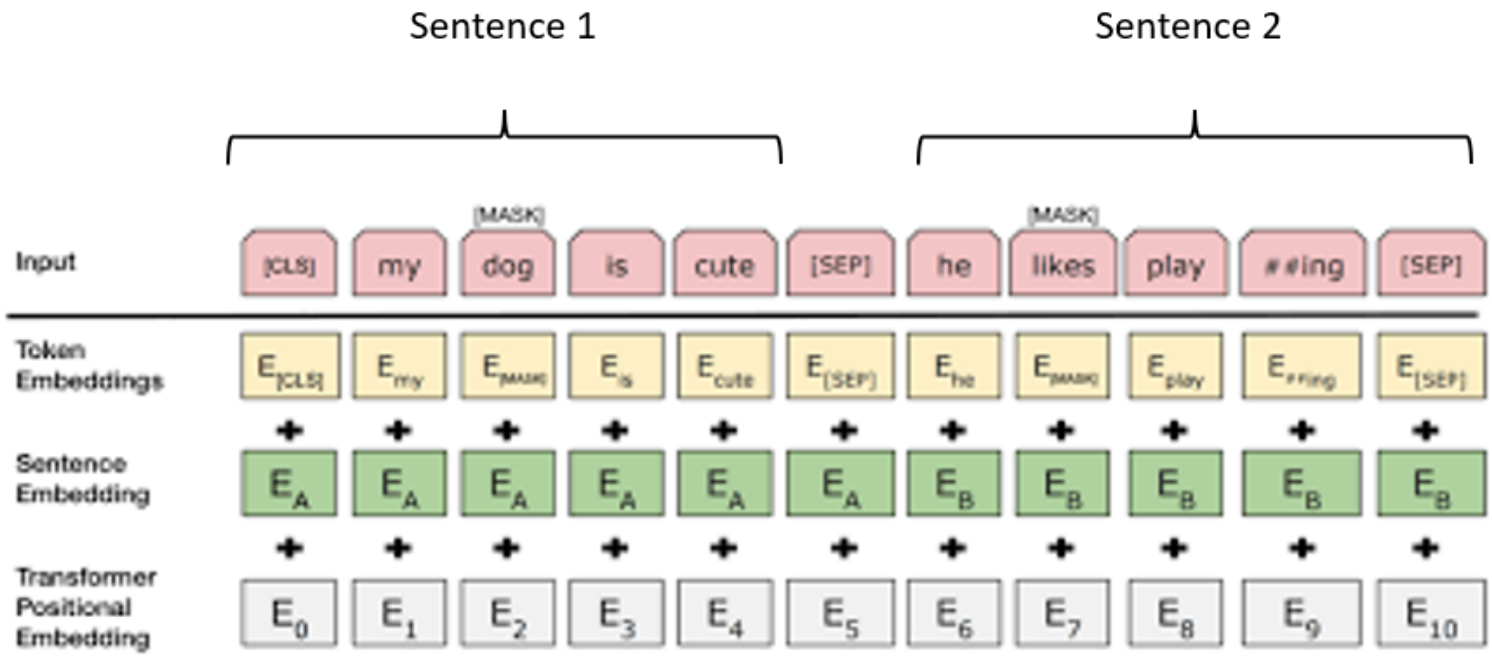
- différentes modalités décrivent une même réalité latente

La tâche prétexte crée un espace de représentation partagé.

Exemple : BERT

BERT (Bidirectional Encoder Representations from Transformers) représente une avancée majeure dans le NLP, introduisant une approche novatrice pour la modélisation de langage. Il utilise la bidirectionnalité pour comprendre le contexte des mots, permettant une compréhension plus fine du langage.

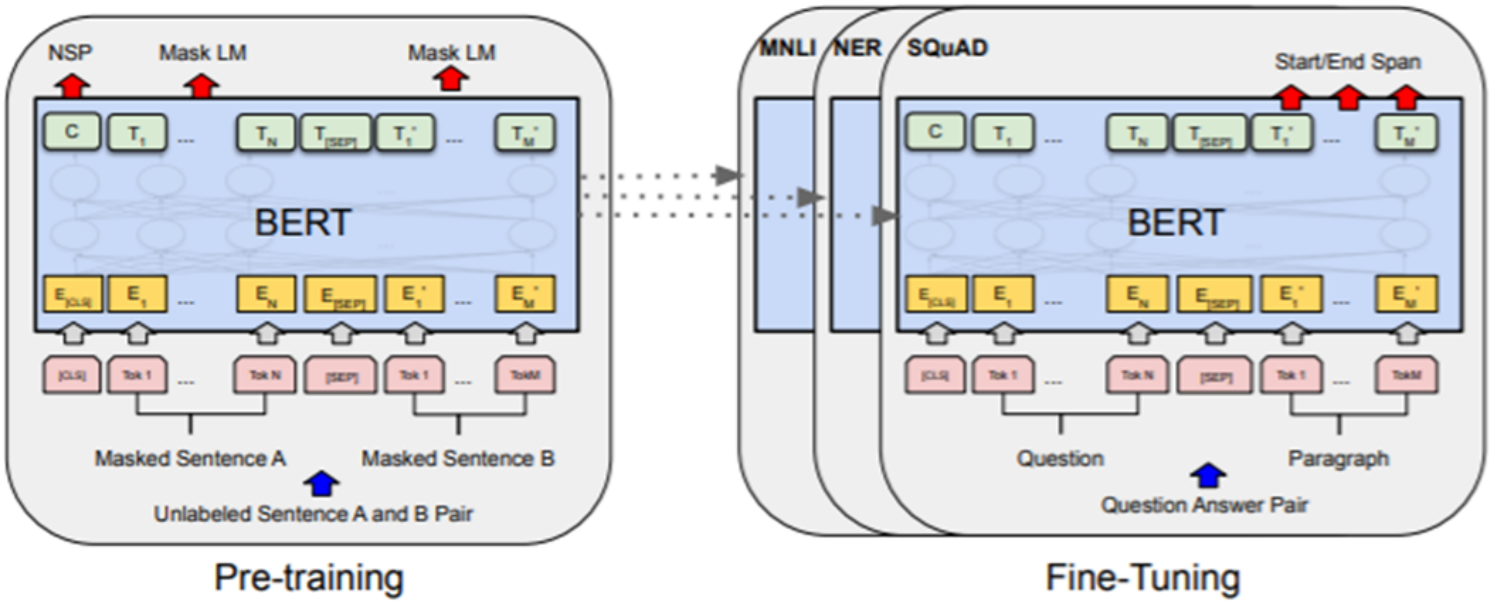
Pré-entraînement



Fine-tuning de BERT pour des tâches downstream

Après pré-entraînement, BERT est affiné pour des tâches spécifiques:

- Ajout d'une couche de sortie spécifique à la tâche (classification, NER, QA).
- Fine-tuning de tous les paramètres du modèle pré-entraîné sur le corpus de la tâche.



Performances de BERT

BERT a été pré-entrainé sur le BookCorpus (800 millions de mots) et English Wikipedia (2,500 millions de mots).

Deux modèles :

- **BERT Base** : 12 couches avec 110 millions de paramètres.
- **BERT Large** : 24 couches avec 340 millions de paramètres.

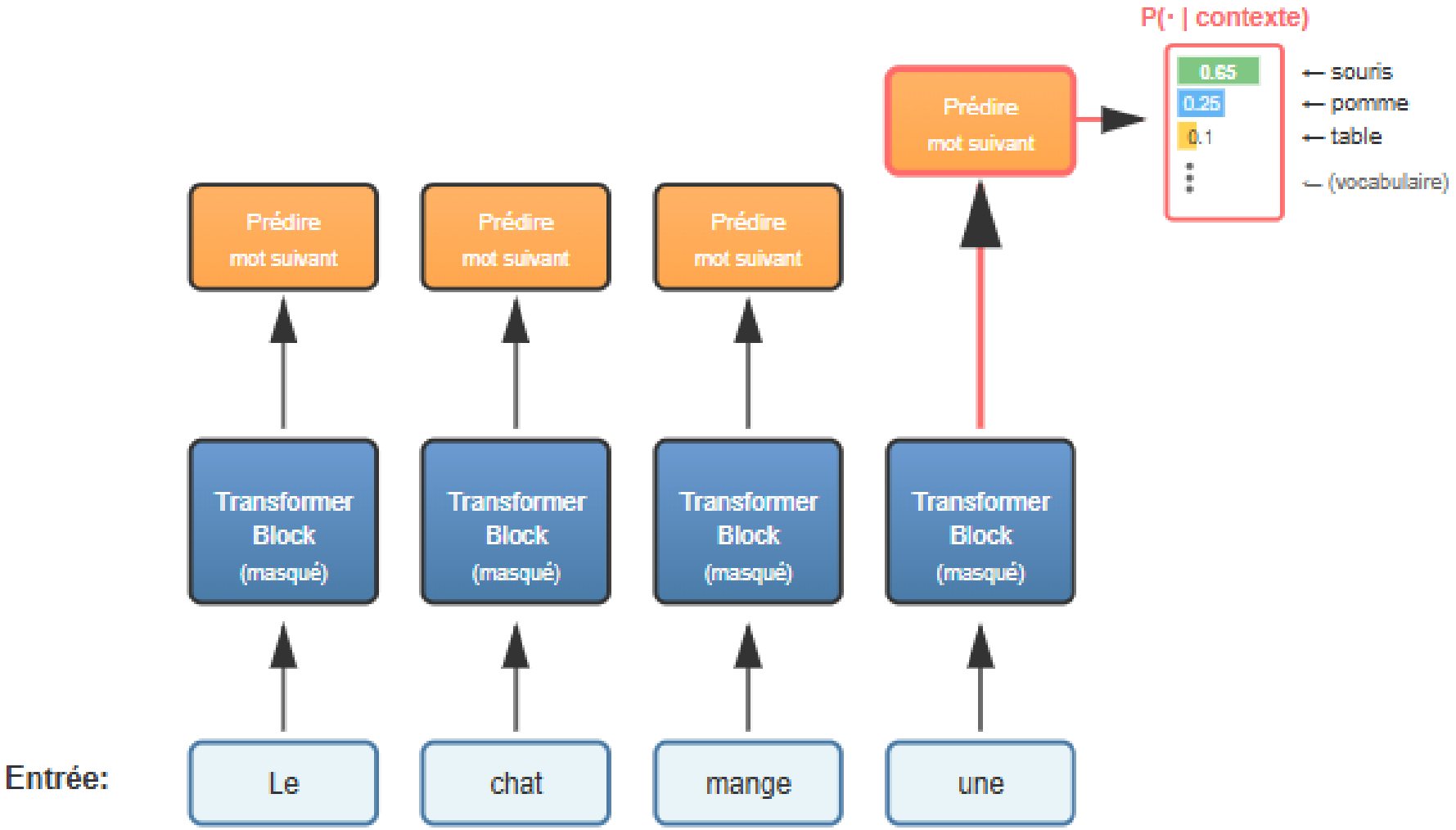
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Modèles génératifs (GPT)

Ce que je ne peux pas créer,
je ne le comprends pas.

– Richard Feynman

Fonctionnement des modèles génératifs



GPT-3

- Développé par OpenAI, publié en 2020.
- Modèle auto-régressif basé sur l'architecture Transformer.

Caractéristique	**Valeur / Détail**
Date	2020
Paramètres	175 milliards
Corpus d'entraînement	Environ 300 milliards de tokens
Contexte maximum	2048 tokens
Ressources GPU	Plusieurs centaines de GPU (type V100)
Coût d'entraînement (estim.)	4,6 M$ à 12 M$ (selon les sources)

- **Applications** : génération de texte, Q&R, résumé, traduction, assistance à la programmation.

Le few-shot learning

- Les grands modèles (comme GPT-3) peuvent réaliser des tâches **sans être explicitement réentraînés** dessus.
- Le principe repose sur **l'utilisation de quelques exemples** (exemples étiquetés) directement dans le prompt ou l'entrée.
- **Zero-shot** : aucune démonstration fournie, le modèle doit comprendre la tâche à partir de sa connaissance générale.
- **One-shot / Few-shot** : on inclut un nombre très réduit d'exemples (un ou quelques-uns) pour guider le modèle dans la résolution de la tâche.
- Exemple :

English: "cat" $\rightarrow$ French: "chat"
English: "house" $\rightarrow$ French: "maison"
English: "car" $\rightarrow$ French: "voiture"

English: "tree" $\rightarrow$ French:

Apprentissage de représentations : Autoencodeurs

Introduction aux autoencodeurs

Définition et objectif :

- Les autoencodeurs sont des réseaux de neurones utilisés pour apprendre des représentations efficaces des données, généralement dans un objectif de compression et de reconstruction.
- Ils sont composés d'un encodeur et d'un décodeur :
  - L'**encodeur** compresse l'entrée dans une représentation dans l'espace latent.
  - Le **décodeur** reconstruit les données d'entrée à partir de cette représentation compressée.

Applications en traitement d'images :

- Réduction de dimension
- Réduction du bruit
- Détection d'anomalies dans les images

Objectif des autoencodeurs :

- Minimiser l'erreur de reconstruction, c'est-à-dire la différence entre l'entrée originale et la sortie obtenue après encodage et décodage.

Architecture de base d'un autoencodeur 1/2

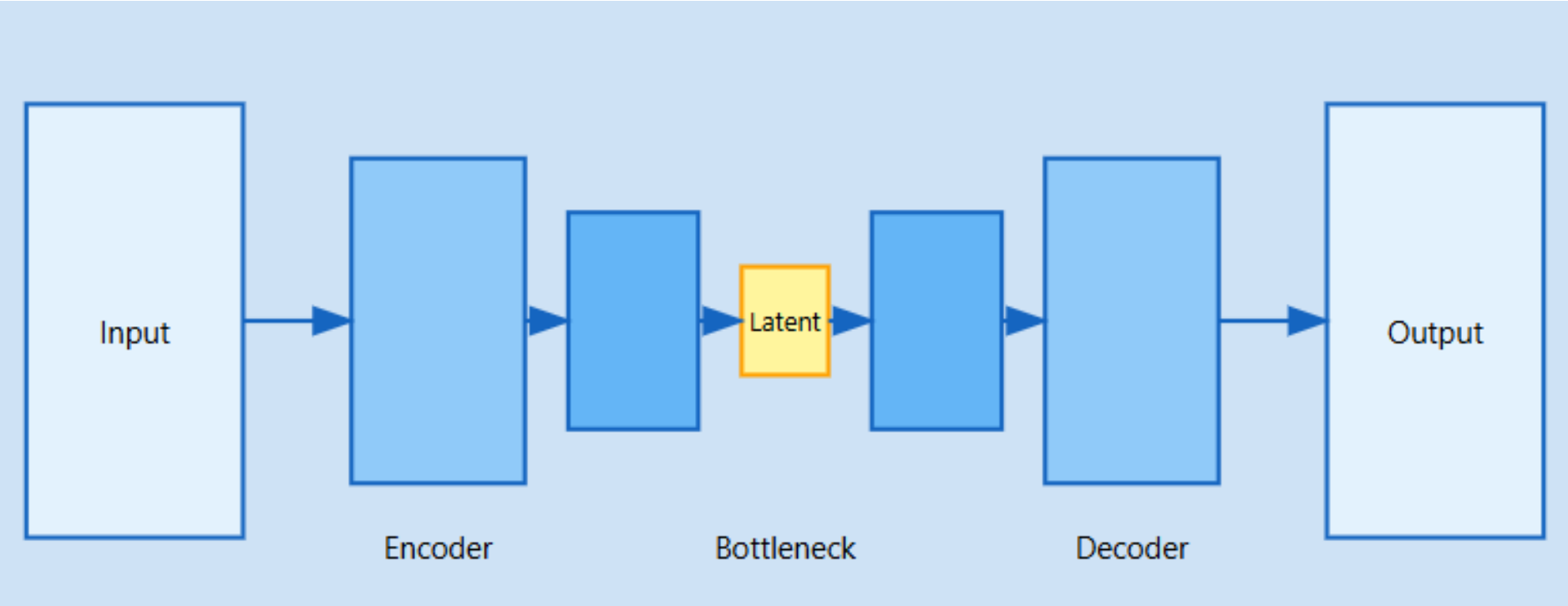
Vue d'ensemble de l'architecture :

- Les autoencodeurs se composent de trois éléments principaux :
  - **Encodeur** : projette l'entrée dans un espace latent de dimension plus faible.
  - **Espace latent (goulot d'étranglement)** : contient une version compressée des données d'entrée.
  - **Décodeur** : reconstruit les données d'entrée à partir de la représentation latente.

Détails de chaque composant :

- **Encodeur** :
  - Succession de couches réduisant progressivement la dimension de l'entrée.
  - Inclut généralement des couches convolutionnelles pour les images afin de capturer des caractéristiques locales.
- **Goulot d'étranglement** :
  - Couche de plus faible dimension du réseau, représentant les caractéristiques essentielles de l'entrée.
  - Agit comme une contrainte, forçant le modèle à apprendre des représentations compactes.
- **Décodeur** :
  - Succession de couches qui augmentent progressivement la résolution pour

Architecture de base d'un autoencodeur 2/2



Encodage avec des couches convolutionnelles

Rôle des couches convolutionnelles dans l'encodeur :

- Les couches convolutionnelles permettent de capturer des hiérarchies spatiales en apprenant des motifs locaux dans l'image d'entrée.
- Elles sont particulièrement adaptées aux images, car elles permettent de détecter des contours, des textures et des formes de manière hiérarchique.

Fonctionnement des couches convolutionnelles :

- **Filtres** : petites matrices (par exemple 3x3 ou 5x5) qui se déplacent sur l'image pour apprendre des caractéristiques locales.
- **Stride et padding** :
  - **Stride** : contrôle le pas de déplacement du filtre, influençant la taille de la sortie.
  - **Padding** : conserve les dimensions spatiales en ajoutant une bordure de zéros autour de l'image.
- **Fonctions d'activation** : la fonction ReLU est couramment utilisée pour introduire de la non-linéarité et permettre l'apprentissage de motifs complexes.

Couches convolutionnelles

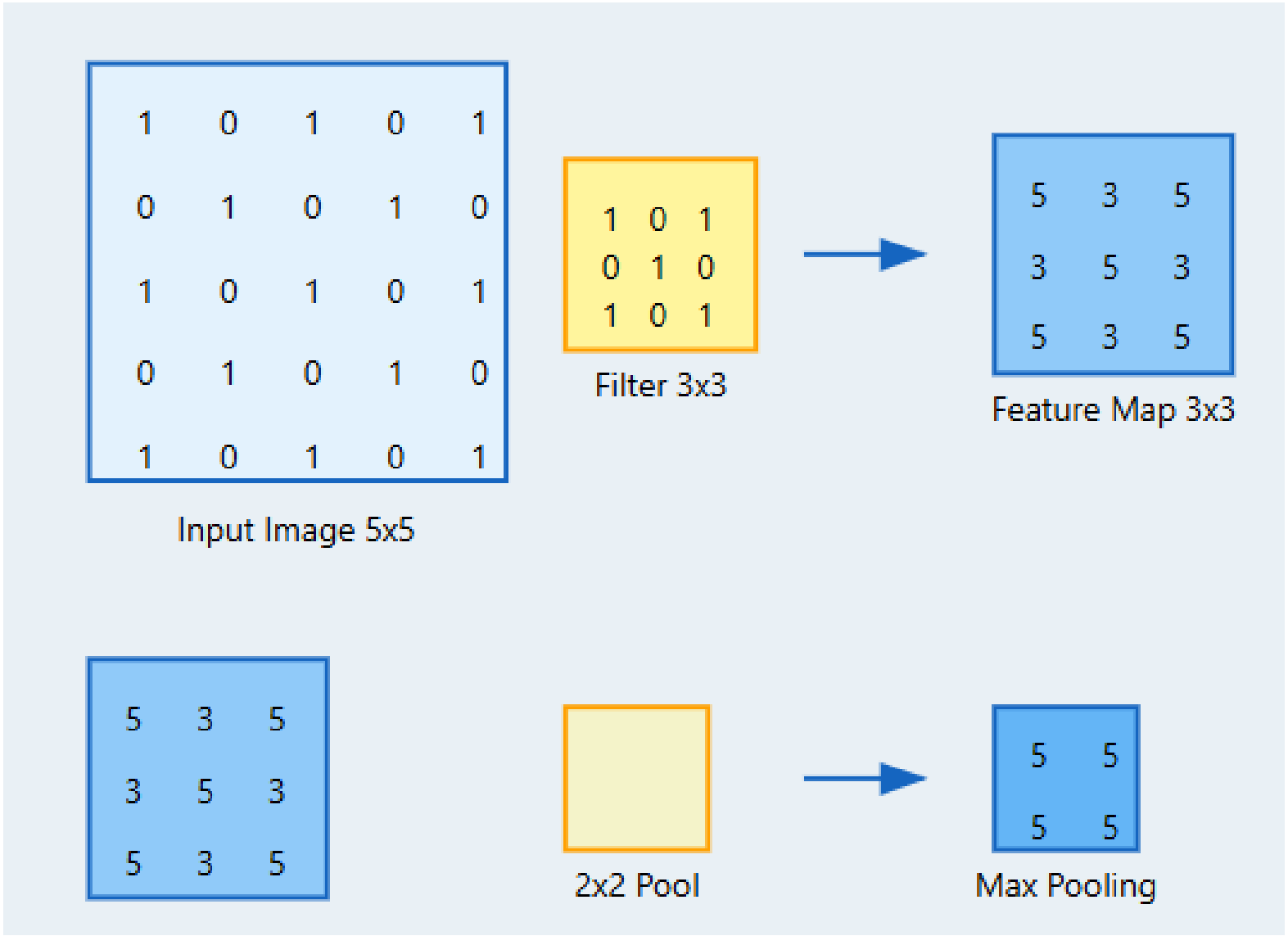
Fonctionnement des filtres :

- **Filtres (noyaux)** : petites matrices (par exemple 3x3, 5x5) utilisées pour détecter des motifs spécifiques dans des régions locales de l'entrée.
- **Processus de convolution** :
  - Le filtre se déplace sur l'image d'entrée, effectuant une multiplication élément par élément suivie d'une sommation pour produire une carte de caractéristiques.
  - Plusieurs filtres sont utilisés afin d'extraire différents types de caractéristiques (contours, textures, etc.).

Pooling pour la réduction de dimension :

- **Max pooling** : réduit les dimensions spatiales des cartes de caractéristiques tout en conservant les informations les plus saillantes, ce qui diminue la charge de calcul.
- Les couches de pooling sont généralement ajoutées après les couches convolutionnelles afin de réduire progressivement la taille de l'image et de se concentrer sur les caractéristiques les plus importantes.

Couche convolutionnelle : exemple



Couches de convolution transposée (déconvolution)

Objectif de la convolution transposée :

- Les convolutions transposées (aussi appelées déconvolutions) sont utilisées pour augmenter la résolution des cartes de caractéristiques dans le décodeur.
- Elles sont essentielles pour reconstruire des images à partir de la représentation compressée de l'espace latent.

Fonctionnement de la convolution transposée :

- Contrairement à la convolution classique, la convolution transposée augmente les dimensions spatiales.
- **Inversion du processus de convolution** :
  - Des zéros sont généralement insérés entre les pixels de la carte de caractéristiques avant d'appliquer la convolution.
  - Cette technique permet d' étendre les dimensions de l'entrée et de générer une sortie de plus haute résolution.

Stride et padding en convolution transposée :

- **Stride** : contrôle le facteur de sur-échantillonnage en définissant l'espacement entre les pixels de sortie.
- **Padding** : comme pour la convolution classique, il permet de contrôler la taille finale de la sortie.

Espace latent (goulot d'étranglement)

Rôle de l'espace latent :

- L'espace latent, également appelé goulot d'étranglement, contient une représentation compressée des données d'entrée.
- Il contraint l'autoencodeur à apprendre les caractéristiques les plus essentielles, en éliminant les détails moins pertinents.

Caractéristiques de la représentation latente :

- L'espace latent possède une dimension plus faible que l'entrée, ce qui pousse le modèle à capturer des caractéristiques de haut niveau.
- Les variables latentes ne sont généralement pas directement interprétables, mais elles représentent des structures ou motifs importants présents dans les données.

Introduction aux autoencodeurs variationnels (VAE)

Que sont les autoencodeurs variationnels (VAE) ?

- Les VAE sont des modèles génératifs qui apprennent à encoder les données dans un espace latent structuré et à les décoder afin d'approximer l'entrée originale.
- Contrairement aux autoencodeurs classiques, les VAE modélisent l'espace latent comme une distribution probabiliste, ce qui permet d'obtenir des représentations plus flexibles et plus riches.

Différences clés avec les autoencodeurs standards :

- Les VAE introduisent une dimension probabiliste en encodant l'entrée sous la forme d'une distribution plutôt que d'un point unique.
- Cela permet de générer de nouvelles données en échantillonnant dans l'espace latent, ce qui rend les VAE particulièrement adaptés aux tâches de génération de données.

Applications des VAE :

- Génération et synthèse d'images (par exemple, création de nouvelles images réalistes).
- Interpolation dans l'espace latent, permettant des transitions continues et

Architecture d'un autoencodeur variationnel (VAE)

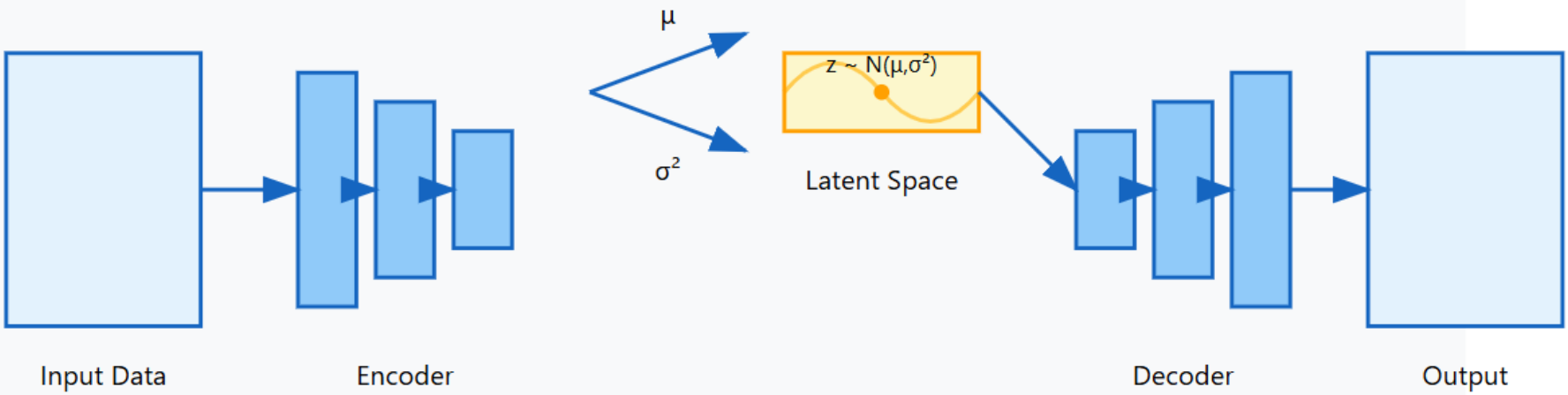
Vue d'ensemble de la structure d'un VAE :

- Comme un autoencodeur classique, un VAE se compose d'un encodeur, d'un espace latent et d'un décodeur.
- Toutefois, dans un VAE, l'encodeur projette l'entrée vers une distribution dans l'espace latent plutôt que vers un point fixe.

Composants d'un VAE :

- **Encodeur** :
  - Projette les données d'entrée vers une distribution latente en produisant une moyenne (μ) et une variance (σ^2) pour chaque dimension de l'espace latent.
- **Espace latent (probabiliste)** :
  - Représente les données sous forme de distribution, généralement gaussienne, à partir de laquelle de nouveaux points peuvent être échantillonnés.
  - Permet une génération structurée des données et des transitions continues dans l'espace latent.
- **Décodeur** :
  - Reconstruit l'entrée à partir d'un point échantillonné dans la distribution latente, en générant de nouvelles données à partir de la structure apprise.

Architecture d'un VAE



Key Components:

- Encoder progressively reduces dimensions to latent space
- Latent space captures probabilistic distribution
- Decoder progressively expands dimensions to reconstruction

Fonction de perte des autoencodeurs variationnels (VAE)

Vue d'ensemble de la fonction de perte :

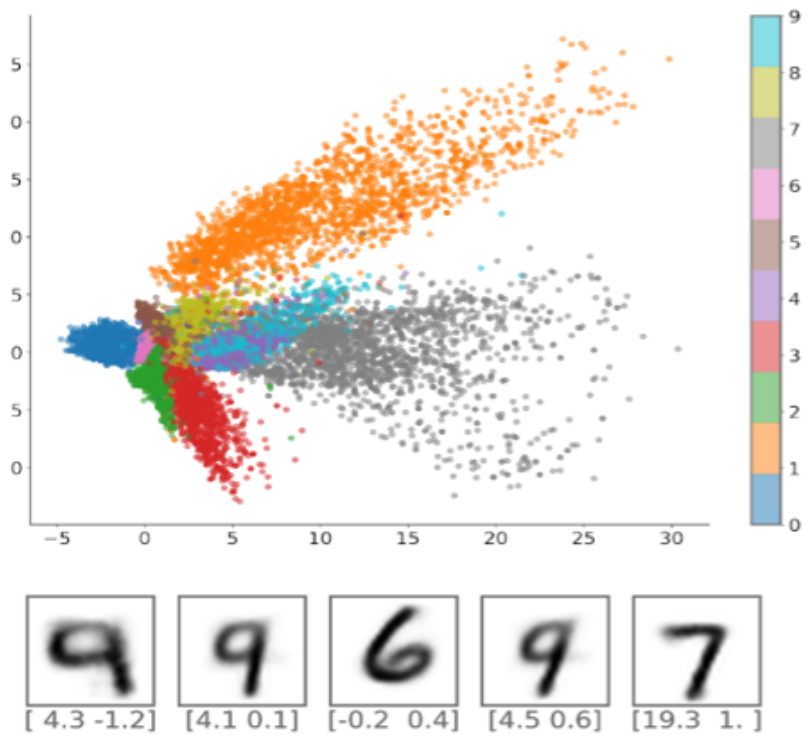
- La fonction de perte d'un VAE combine deux termes : la **perte de reconstruction** et la **divergence de Kullback-Leibler (KL)**.
- Cette combinaison permet d'apprendre à la fois une reconstruction fidèle et un espace latent structuré.

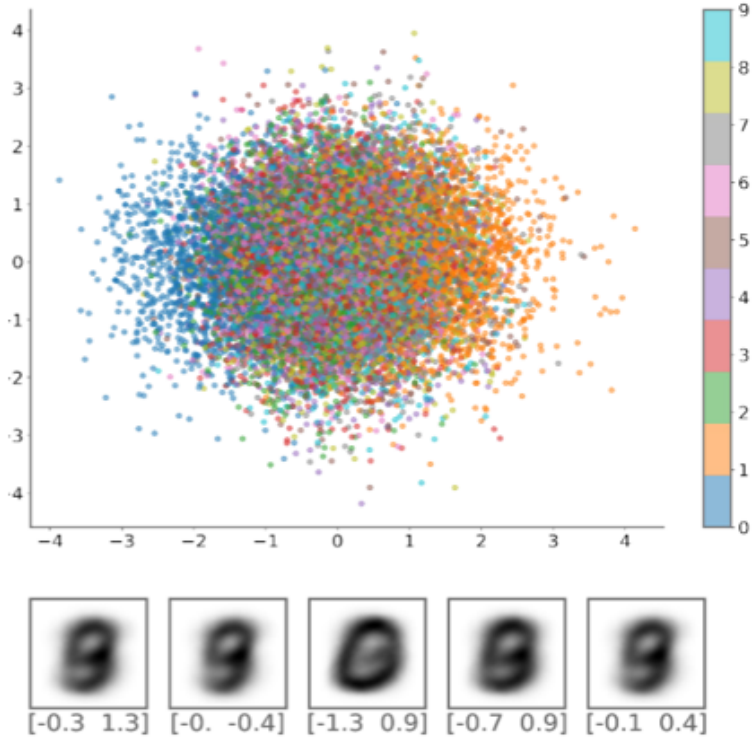
$$\mathcal{L}_{\text{VAE}}(x) = \mathbb{E}_{q_\phi(z|x)} \left[-\log p_\theta(x|z) \right] + \text{KL} \left(q_\phi(z|x) \,\|\, p(z) \right)$$

1. **Perte de reconstruction** :
   - Mesure l'écart entre l'entrée et sa reconstruction, généralement à l'aide de l'erreur quadratique moyenne (MSE) ou de l'entropie croisée binaire.
   - Elle incite le décodeur à produire des sorties proches de l'entrée originale.

2. **Perte de divergence KL** :
   - Mesure l'écart entre la distribution latente apprise et une distribution normale standard ($N(0, 1)$).
   - Joue un rôle de régularisation en forçant l'espace latent à être lisse, structuré et proche d'une loi normale.
   - Formule : $\text{KL}(q(z|x) \,\|\, p(z)) = \int q(z|x) \log \frac{q(z|x)}{p(z)} \, dz$

Décomposition de la fonction de perte





Entraînement et optimisation des VAE 1/2

Objectif de l'entraînement :

- L'objectif est de minimiser la perte globale du VAE, en équilibrant la qualité de reconstruction et la structuration de l'espace latent.
- Le processus d'apprentissage ajuste les poids de l'encodeur et du décodeur afin d'optimiser simultanément les deux termes de la perte.

Astuce de reparamétrisation (reparameterization trick) :

- Afin de permettre la rétropropagation à travers l'opération d'échantillonnage dans l'espace latent, on utilise l'astuce de reparamétrisation.
- Au lieu d'échantillonner directement $z \sim q(z|x)$, on échantillonne $\epsilon \sim N(0, 1)$ et on calcule $z = \mu + \sigma \cdot \epsilon$.
- Cette reformulation permet aux gradients de circuler à travers μ et σ, rendant l'apprentissage différentiable.

Entraînement et optimisation des VAE 2/2

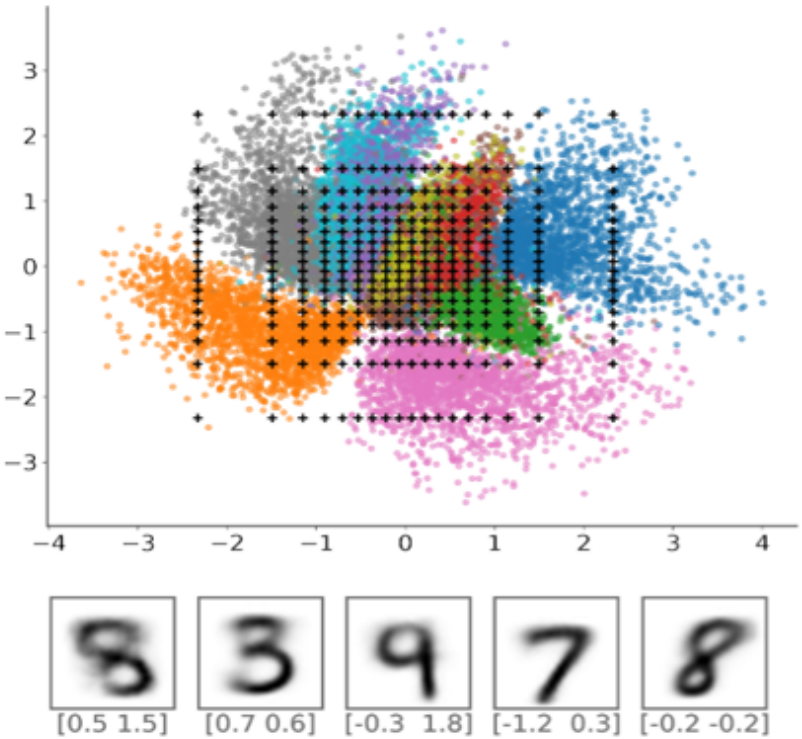
Processus d'optimisation :

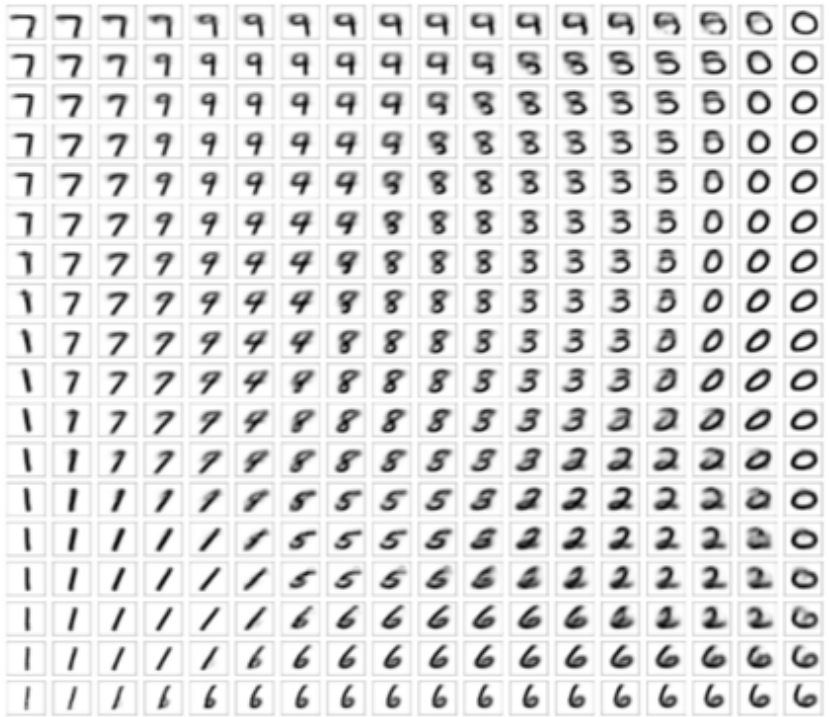
- La descente de gradient stochastique (SGD) ou l'optimiseur Adam est généralement utilisé pour minimiser la perte du VAE.
- L'entraînement consiste à ajuster itérativement les poids de l'encodeur et du décodeur afin d'affiner la distribution latente apprise et la qualité de reconstruction.

Compromis entre reconstruction et régularisation :

- L'ajustement du coefficient β associé à la divergence KL permet de contrôler l'équilibre entre fidélité de reconstruction et régularisation de l'espace latent.
- Une valeur élevée de β favorise un espace latent plus lisse et plus structuré, mais peut dégrader la qualité de reconstruction.

Régularisation des VAE





Merci de votre attention

redha_moulla@yahoo.fr