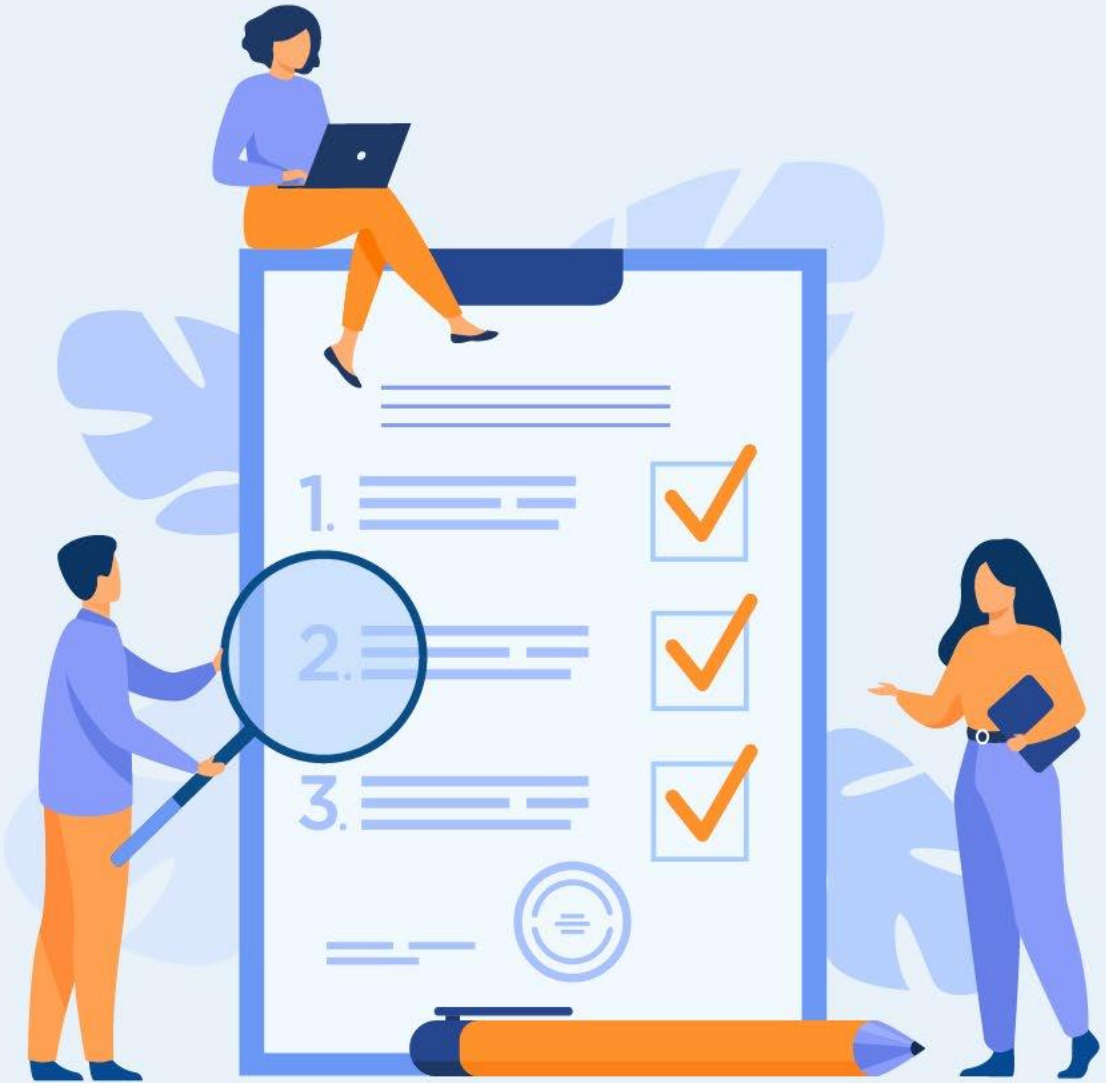
PARTIE 3

EXPLORER LES RÉSEAUX RÉCURRENTS

Dans ce module, vous allez :

- Comprendre les réseaux récurrents
- Appliquer les réseaux de mémoire à long terme (LSTM) et les unités récurrentes à portes (GRU)

24 heures



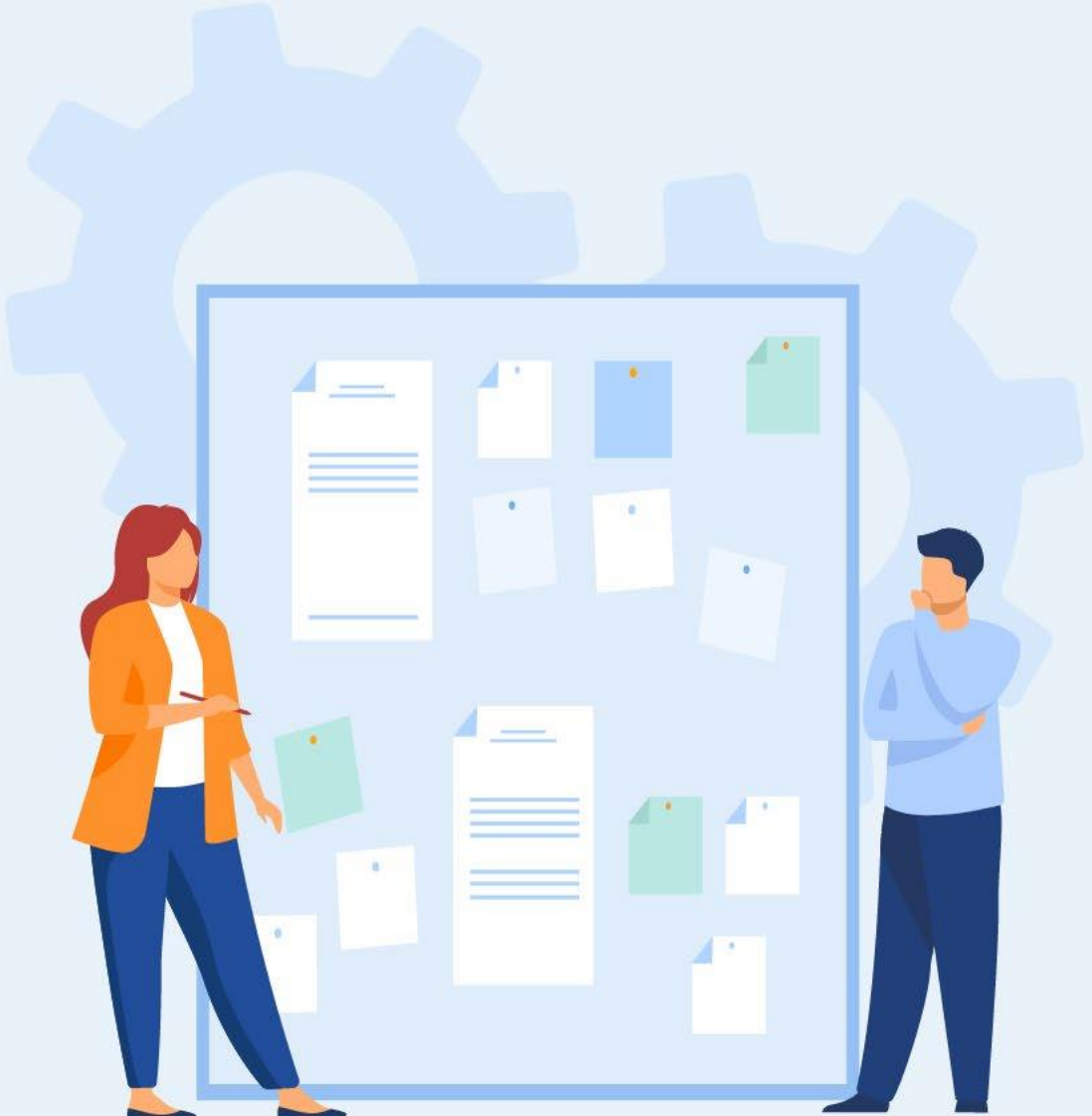
CHAPITRE 1

COMPRENDRE LES RÉSEAUX RÉCURRENTS

Ce que vous allez apprendre dans ce chapitre :

- La récurrence d’un réseau de neurones récurrent
- Les différents réseaux de neurones récurrents

12 heures



CHAPITRE 1

COMPRENDRE LES RÉSEAUX RÉCURRENTS

1. **Introduction aux réseaux récurrents**
2. **LSTM**

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Introduction aux réseaux récurrents

Dans les chapitres précédents, on s'est focalisé principalement sur les problèmes de prédiction avec des entrées et sorties de taille fixe.

Problématique : Que peut-on faire si l'entrée et/ou la sortie est une séquence de longueur variable ?

Classer un restaurant, un film ou un avis sur un produit comme positif ou négatif :

- ✓ "La nourriture était vraiment bonne"
- ✓ "La machine à laver est tombée en panne après deux semaines"
- ✓ "Le film avait des parties lentes, mais dans l'ensemble, cela valait la peine d'être vu"

Problématique : Quelle représentation de caractéristiques ou structure de prédicteur peut-on utiliser pour ce problème ?

Cela peut être une tâche de simplement classer les tweets en sentiments positifs et négatifs. Donc, ici, l'entrée serait un tweet de longueurs variables, tandis que la sortie est d'un type et d'une taille fixes.

PARTIE 3

Définition

Un réseau de neurones récurrents (RNN, Recurrent Neural Network en anglais) est un type de réseau de neurones artificiels qui est conçu pour traiter des données séquentielles ou temporelles. Contrairement aux réseaux de neurones classiques, les RNN ont une architecture récurrente qui leur permet de conserver une mémoire interne et de prendre en compte le contexte précédent lors du traitement de chaque élément d'une séquence. Les RNN sont largement utilisés dans de nombreuses applications telles que :

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Introduction aux réseaux récurrents

Pourquoi les RNN ?

Les données séquentielles (texte, séries temporelles, audio…) contiennent une dépendance dans le temps : ce qui se passe maintenant dépend souvent de ce qui s'est passé auparavant. Les réseaux de neurones « classiques » (perceptron multicouche, CNN) traitent chaque donnée de manière indépendante. Les RNN sont conçus pour « se souvenir » d'informations passées et les utiliser pour interpréter la suite de la séquence.

L'Idée de Base : La Mémoire Artificielle

Imaginez lire un livre :

- Pour comprendre la phrase "Il pleuvait, donc il prit son _____", votre cerveau se souvient de "pleuvait" pour deviner "parapluie".
- Un réseau de neurones classique (comme un perceptron) oublie chaque entrée après traitement.
- Un RNN, lui, garde une trace du passé grâce à une mémoire interne. C'est comme un assistant qui prend des notes pendant une réunion.

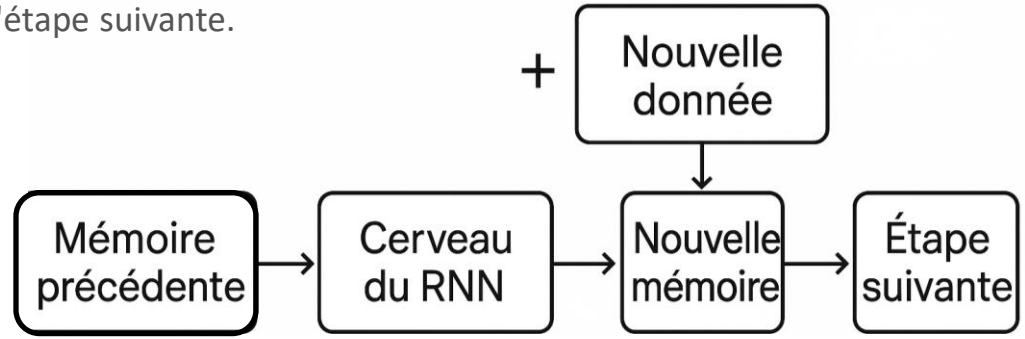
01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Introduction aux réseaux récurrents

Structure d'un RNN

À chaque étape :

- Le RNN reçoit une nouvelle donnée (ex: un mot dans une phrase).
- Il combine cette donnée avec ce dont il se souvient (la "mémoire" de l'étape précédente).
- Il produit :
  - Une sortie (ex: une prédiction).
  - Une nouvelle version de sa mémoire pour l'étape suivante.



PARTIE 3

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Introduction aux réseaux récurrents

Le Problème Majeur : La Mémoire à Court Terme

Exemple :

"Je suis né au Maroc... J'ai vécu à Casablanca 10 ans... Je parle couramment ____."

Un RNN simple oublie « Maroc" après 20 mots → ne peut pas deviner « darija".

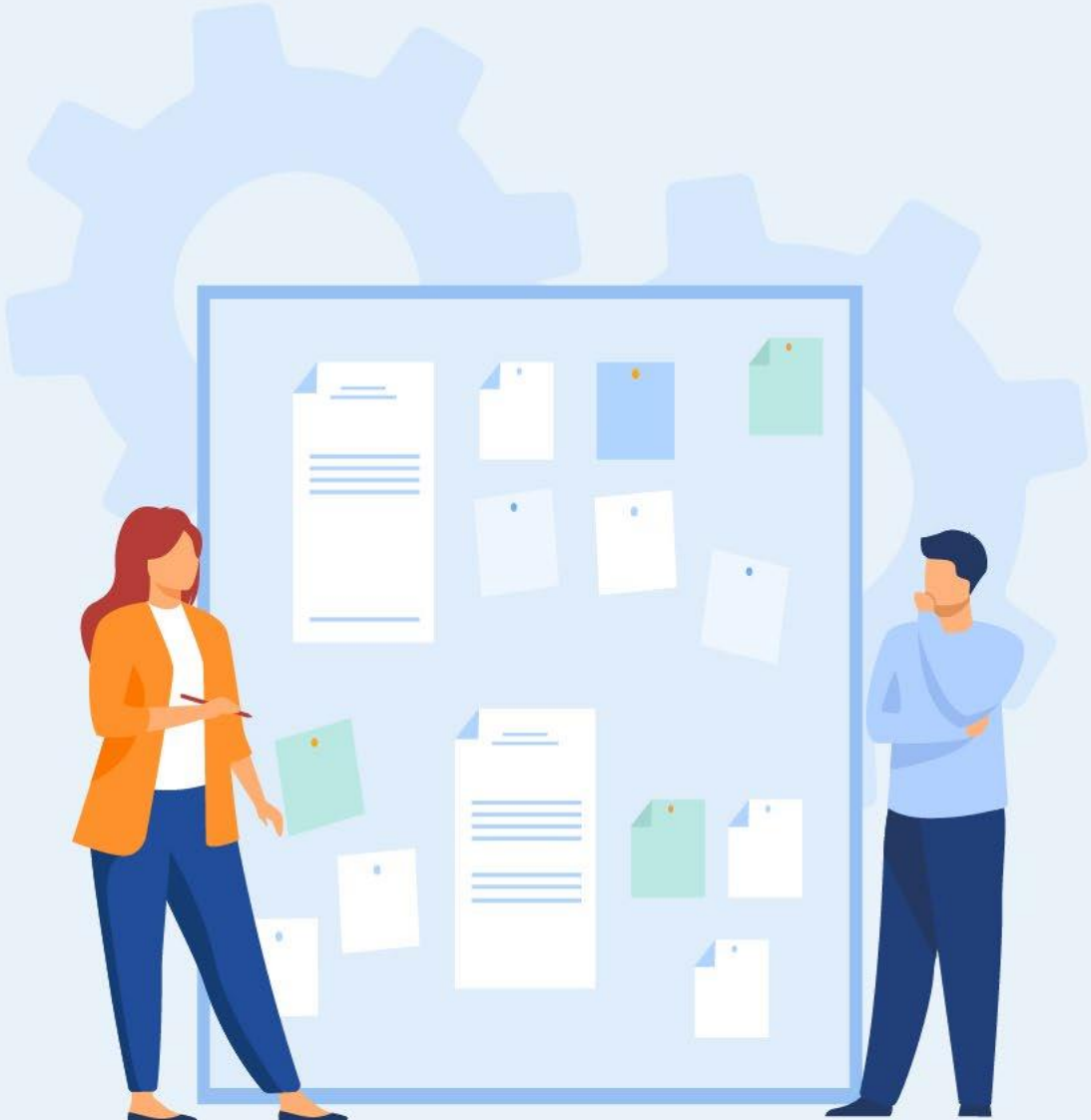
Solutions : LSTM et GRU (Des RNN Améliorés)

LSTM (Long Short-Term Memory) :

- Comme un secrétaire avec un bloc-notes.
- Il décide quoi oublier, quoi noter, et quoi relire via des "portes".
- Exemple : Retient « Maroc" pendant toute la phrase.

GRU (Gated Recurrent Unit) :

- Version simplifiée du LSTM, tout aussi efficace.
- Moins de calculs → plus rapide.



CHAPITRE 1

COMPRENDRE LES RÉSEAUX RÉCURRENTS

1. Introduction aux réseaux récurrents
2. **LSTM**

Les 3 Portes du LSTM

Porte	Rôle	Analogie
Porte d'Oubli (Forget Gate)	Décide ce qu'il faut **effacer** de la mémoire	"Ce vieux dossier est inutile → je le jette !"
Porte d'Entrée (Input Gate)	Décide ce qu'il faut **enregistrer** dans la mémoire	"Cette info est cruciale → je l'archive !"
Porte de Sortie (Output Gate)	Décide quelle partie de la mémoire **utiliser maintenant**	"Pour répondre à cette question, je consulte le dossier X"

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS
LSTM

Comment ça Fonctionne ? Exemple avec une Phrase

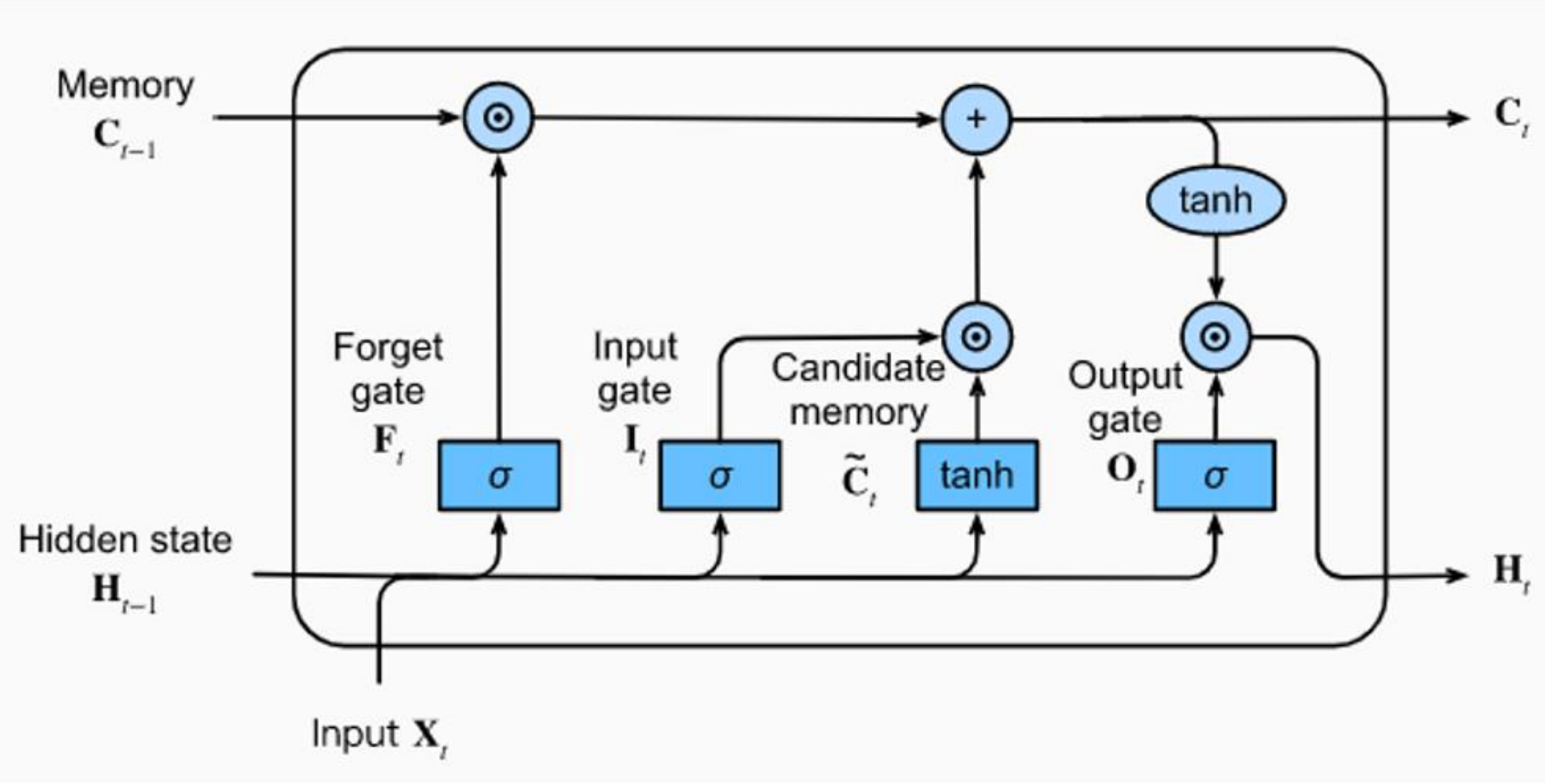
"Hier il a plu toute la journée... Aujourd'hui, le ciel est _____."

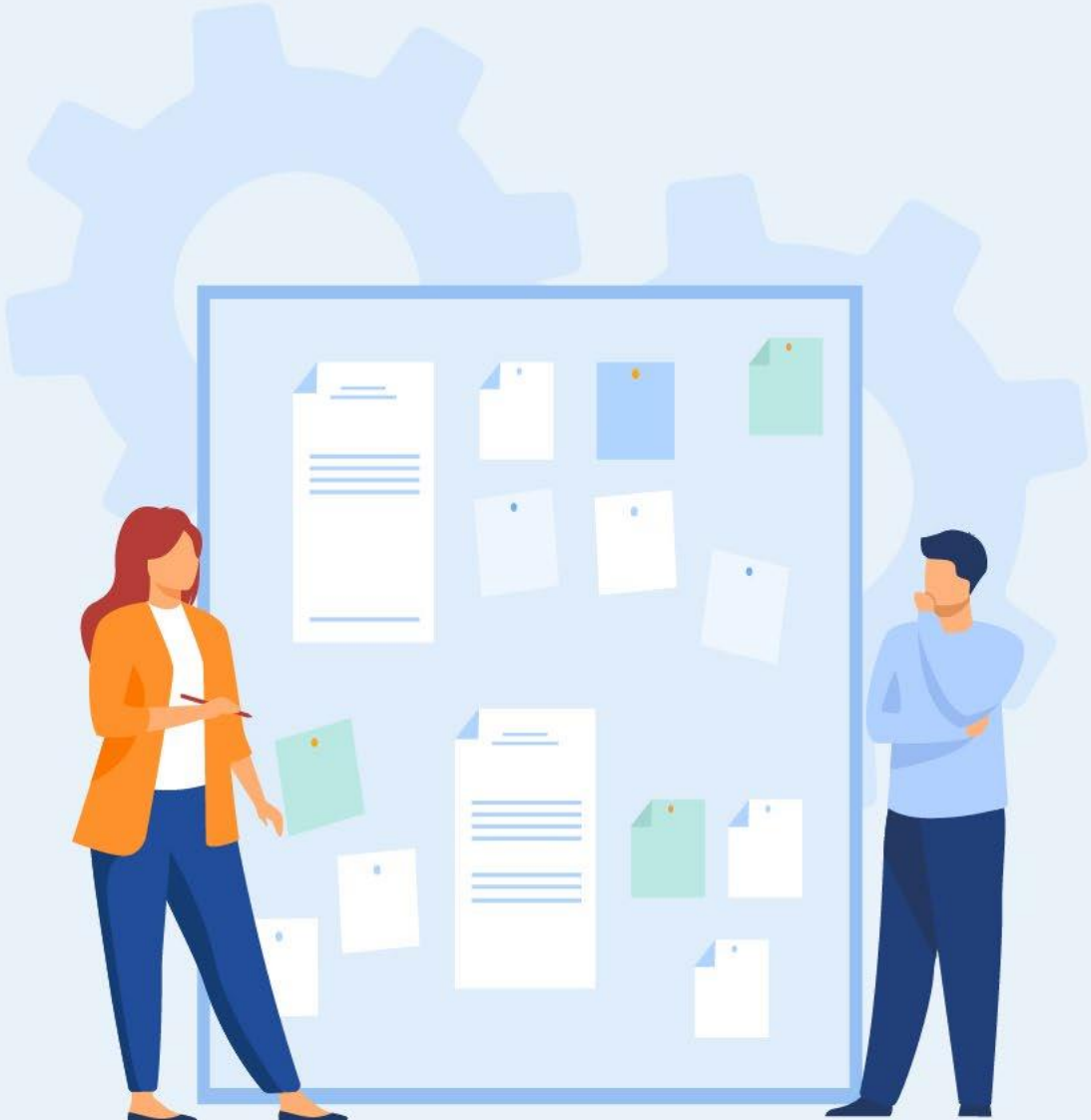
Étape	Porte d'Oubli	Porte d'Entrée	Porte de Sortie	Mémoire Long-Terme
1	Garde tout (rien à oublier)	Enregistre **"pluie"**	Pas encore utilisé	Contient : "pluie"
2	Garde "pluie" (important)	Enregistre **"ciel"**	Utilise "pluie" + "ciel"	Contient : "pluie" + "ciel"
Résultat	→ Prédit **"couvert"** ou **"gris"**			

PARTIE 3

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

LSTM





CHAPITRE 1

COMPRENDRE LES RÉSEAUX RÉCURRENTS

1. Introduction aux réseaux récurrents
2. LSTM
3. **Les Séquences et la Structuration des Données pour les RNN**

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Les Séquences et la Structuration des Données pour les RNN

Qu'est-ce qu'une Séquence ?

Définition : Une suite ordonnée de données où l'ordre et la position sont cruciaux.

Exemples :

- Texte : ["Bonjour", "comment", "allez", "vous", "?"]
- Série temporelle : [8h, 9h, 10h]

Le Concept de Timestep (Pas de Temps)

Définition : Un timestep est une unité de position dans la séquence.

- Pour du texte : Chaque mot = 1 timestep
- Pour des données horaires : Chaque heure = 1 timestep

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Les Séquences et la Structuration des Données pour les RNN

Structuration des Données d'Entraînement

Les RNN nécessitent une structure spécifique en 3 dimensions :

X.shape = (n_exemples, time_step, n_features)

Y.shape = (n_exemples,)

Exemples Concrets de Structuration

Données Boursières

- Données brutes : [156.3, 157.1, 155.9, 158.4] (4 jours)
- Découpage (timesteps=3) :

X (entrée) = [[156.3, 157.1, 155.9]] → y (cible) = [158.4]

PARTIE 3

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Les Séquences et la Structuration des Données pour les RNN

Préparation Typique des Données

1. **Découpage :**

Créer des fenêtres glissantes de longueur fixe (timesteps)

Données : [A, B, C, D, E] → timestamp : 3

Fenêtre 1: [A,B,C] -> cible D

Fenêtre 2: [B,C,D] -> cible E

2. **Normalisation :**
   Mettre les données à l'échelle (ex: entre 0 et 1)

3. **Redimensionnement :**

X = X.reshape((X.shape[0], timesteps, features)

Exemple de code

```python
# Données brutes : [1, 2, 3, 4, 5, 6] avec timesteps=3
X = np.array([
    [[1], [2], [3]],  # Séquence 1
    [[2], [3], [4]],  # Séquence 2
    [[3], [4], [5]]   # Séquence 3
])  # Forme : (3, 3, 1)

y = np.array([4, 5, 6])  # Valeurs cibles
```

01 - COMPRENDRE LES RÉSEAUX RÉCURRENTS

Les Séquences et la Structuration des Données pour les RNN

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential([
    # Couche LSTM avec 50 neurones (return_sequences=False par défaut)
    LSTM(50, activation='tanh', input_shape=(timesteps, features)),

    # Couche dense pour la sortie
    Dense(1)  # Régression (1 sortie)
])

model.compile(optimizer='adam', loss='mse')
model.summary()
```

Paramètres clés :

- **units** : Nombre de cellules LSTM
- **activation** : Fonction d'activation (tanh par défaut)
- **input_shape** : (timesteps, features) (sans batch_size)
- **return_sequences** : True si couche LSTM suivante

```python
#Code référence de création de séquences
from sklearn.preprocessing import MinMaxScaler

# 1. Normalisation des données
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data)

# 2. Création des séquences
def create_dataset(
    dataset,           #Remplacez par votre array NumPy (shape = (n_samples, n_features))
    time_step=1,       #Nombre de pas de temps à prendre en entrée
    input_cols=[0],    #Index(es) des colonne(s) d'entrée (ex. [0] pour univarié, [0,1,2] pour multivarié)
    target_col=0       #Index de la colonne à prédire (ex. 0 pour première colonne)
):
    dataX, dataY = [], []
    n_samples = len(dataset)
    for i in range(n_samples - time_step - 1):
        # 1) Fenêtre d'entrée (time_step, n_features_entrée)
        window = dataset[i : i + time_step, input_cols]
        dataX.append(window)
        # 2) Valeur cible
        target = dataset[i + time_step, target_col]
        dataY.append(target)
    return np.array(dataX), np.array(dataY)
#Exemple
X, y = create_dataset(
    dataset=data_scaled,
    time_step=48,
    input_cols=[0,1,2],
    target_col=0
)
```