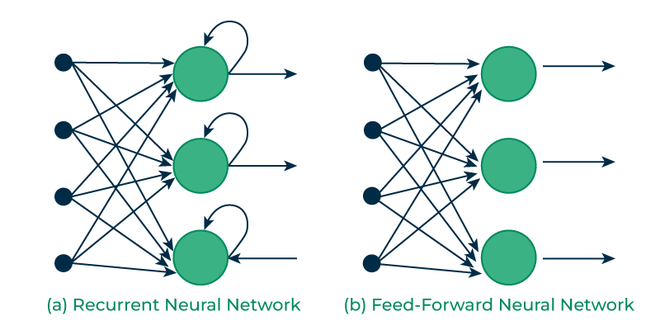
# TP8 : Prédictions sur le dataset IMDB avec un RNN - Ressources

Les **réseaux neuronaux récurrents (RNN)** sont une classe de réseaux neuronaux spécialisés dans le traitement des données séquentielles. Contrairement aux réseaux neuronaux classiques, les RNN possèdent des connexions récurrentes, ce qui signifie qu'ils peuvent maintenir une sorte de "**mémoire**" de l'information précédente dans la séquence.

## Architecture CNN:



## Fonctionnement des RNN

* + **Séquences et Boucles** : Un RNN traite les séquences d'entrée pas à pas. À chaque pas de temps, il prend en compte non seulement l'entrée actuelle mais aussi l'état précédent (la mémoire de ce qui a été vu jusqu'à présent).
  + **Cellules RNN** : Chaque cellule d'un RNN reçoit une entrée et l'état précédent, et génère une sortie ainsi qu'un nouvel état, qui sera utilisé pour le prochain pas de temps.

## Avantages des RNN :

* + Les RNN sont spécialement conçus pour gérer des séquences de données, ce qui les rend idéaux pour des tâches telles que la reconnaissance de la parole, la traduction automatique et la génération de texte.
  + Ils ont la capacité de retenir des informations provenant des entrées précédentes grâce à leurs boucles internes, ce qui permet de prendre en compte le contexte de la séquence.
  + Grâce à leur architecture récurrente, les RNN peuvent apprendre et modéliser des relations complexes dans les données séquentielles.

## Inconvénients des RNN :

* + Les RNN peuvent être plus lents à entraîner et à exécuter par rapport aux réseaux de neurones classiques, en raison de leur nature séquentielle et de la nécessité de conserver les états précédents.
  + Ils ont souvent des difficultés à capturer des dépendances à long terme dans les séquences, bien que des variantes comme les LSTM ont été développées pour pallier cette limitation.
  + Les RNN sont susceptibles de surapprentissage (overfitting), surtout lorsqu'ils sont utilisés sur de petites quantités de données ou avec des modèles trop complexes.

## Architecture CNN pour MNIST:

1. **Initialisation du Modèle**
   * **Modèle Séquentiel** : Nous commençons par créer une instance du modèle séquentiel, qui permet de construire un réseau couche par couche de manière linéaire.
2. **Ajout de la Couche d'Embedding (Embedding)**
   * **Couche d'Embedding** : Cette couche transforme les indices de mots en vecteurs de dimensions fixes. Elle permet de convertir chaque mot en un vecteur dense, facilitant ainsi l'apprentissage des relations entre les mots.
   * **Nombre de Mots (max\_features)** : Nous définissons le nombre maximal de mots uniques que nous considérons dans le vocabulaire (par exemple, 10 000 mots les plus fréquents).
   * **Dimensions des Vecteurs (embedding\_dim)** : La taille de chaque vecteur d'embedding, déterminant la dimension de l'espace d'embedding (par exemple, 128 dimensions).
   * **Longueur des Séquences (maxlen)** : La longueur des séquences d'entrée. Cela garantit que toutes les séquences ont la même longueur (par exemple, 500 mots par séquence).
3. **Ajout de la Couche RNN (SimpleRNN)**
   * **Couche RNN** : Cette couche traite les séquences de vecteurs d'embedding. Elle a des unités récurrentes capables de garder une mémoire des états précédents pour mieux comprendre le contexte des séquences.
   * **Nombre d'Unités (units)** : Le nombre d'unités dans la couche RNN, déterminant la capacité de mémoire de la cellule récurrente (par exemple, 128 unités).
4. **Ajout de la Couche Dense**
   * **Couche Dense** : Cette couche effectue la classification finale. Elle prend les sorties de la couche RNN et produit une prédiction.
   * **Nombre d'Unités (units)** : Le nombre d'unités dans la couche dense. Pour la classification binaire, nous utilisons une seule unité de sortie.
   * **Fonction d'Activation (activation)** : Nous utilisons la fonction d'activation sigmoid pour produire une probabilité (un nombre entre 0 et 1) indiquant si la critique est positive ou négative.
5. **Compilation du Modèle**
   * **Compilation** : Nous spécifions l'optimiseur, la fonction de perte et les métriques de performance.
   * **Optimiseur** : Un optimiseur adaptatif comme Adam est couramment utilisé pour ajuster les poids du modèle.
   * **Fonction de Perte** : La fonction de perte binaire cross-entropie est utilisée pour la classification binaire, mesurant la différence entre les prédictions du modèle et les étiquettes réelles.
   * **Métriques** : Nous utilisons la précision comme métrique pour évaluer les performances du modèle.