ENSPY – Université de Yaoundé I

Master 2 – GI

TP 5 – Deep Learning

Modélisation de séquences et mécanismes d'attention

Auteur :
Donfack Pascal Arthur

Encadrant :
Dr Louis Fippo

Année Académique : 2025-2026
Date : 28 décembre 2025

Table des matières

1 Introduction Générale

Le traitement des données séquentielles a connu une révolution majeure avec l'avènement du Deep Learning. Historiquement dominé par les Réseaux de Neurones Récurrents (RNN) tels que les LSTM et GRU, le domaine a été transformé par l'introduction des mécanismes d'attention. Ces mécanismes permettent de s'affranchir des contraintes de mémorisation séquentielle en permettant au modèle de focaliser son "attention" sur des parties spécifiques de l'entrée, quelle que soit leur position temporelle.

Ce Travail Pratique (TP) vise à explorer ces concepts fondamentaux à travers une approche progressive, allant de l'implémentation basique d'une couche d'attention sur un RNN jusqu'à la conception d'une architecture Séquence-à-Séquence (Seq2Seq) complète pour la prédiction de séries temporelles, intégrant un suivi expérimental rigoureux via MLflow.

2 Contexte et Objectifs

L'objectif principal est de maîtriser l'implémentation et l'analyse des mécanismes d'attention. Plus spécifiquement :

- **Partie 1** : Implémenter "from scratch" une couche d'attention simple et l'intégrer à un modèle GRU.
- **Partie 2** : Concevoir un modèle Seq2Seq combinant un encodeur Bi-LSTM et un décodeur à attention croisée (Cross-Attention) pour résoudre un problème de prédiction sur données synthétiques.
- **Partie 3** (Challenge Recherche) : Proposer des améliorations architecturales à un modèle d'état de l'art (TAP) pour la cohérence temporelle longue (traitée dans un article séparé).

Une attention particulière est portée à la qualité du code (modularité, clarté) et à la méthodologie expérimentale (reproductibilité, analyse des poids d'attention).

3 Fondements Théoriques

Cette section aborde les concepts clés nécessaires à la compréhension des mécanismes d'attention, en répondant spécifiquement aux questions théoriques posées dans le sujet.

3.1 Attention par Produit Scalaire Mis à l'Échelle (Scaled Dot-Product Attention)

L'attention par produit scalaire est le mécanisme central des architectures modernes comme le Transformer.

3.1.1 Formulation Mathématique

Étant donnés une matrice de requêtes Q (Queries), une matrice de clés K (Keys) et une matrice de valeurs V (Values), les poids d'attention sont calculés selon la formule suivante :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Où d_k représente la dimension des vecteurs de clés.

3.1.2 Rôle du Facteur de Mise à l'Échelle

La division par $\sqrt{d_k}$ est fondamentale pour la stabilité de l'apprentissage. **Justification :** Lorsque la dimension d_k est grande, le produit scalaire $q \cdot k$ peut atteindre des valeurs très élevées (en magnitude). Sans cette normalisation, les logits passés à la fonction softmax seraient très grands, poussant les sorties de la softmax vers 0 ou 1 (saturation). Dans ces zones saturées, les gradients de la fonction softmax sont extrêmement faibles (proches de zéro), ce qui empêche la rétropropagation efficace de l'erreur et bloque l'apprentissage ("Vanishing Gradient Problem"). Le facteur $\frac{1}{\sqrt{d_k}}$ ramène l'ordre de grandeur du produit scalaire à une variance unitaire (si Q et K sont normalisés), garantissant des gradients fluides.

3.2 Distinction Self-Attention vs Cross-Attention

Bien que la formule mathématique soit identique, la différence réside dans la **provenance** des données alimentant Q, K et V.

- **Self-Attention (Auto-attention)** :
  - **Provenance** : Q, K et V proviennent tous de la *même* séquence d'entrée (par exemple, la sortie de la couche précédente de l'encodeur).
  - **Objectif** : Modéliser les dépendances internes à la séquence. Par exemple, dans la phrase "L'animal a traversé la rue car il était pressé", la self-attention permet au modèle d'associer "il" à "animal".
- **Cross-Attention (Attention croisée)** :
  - **Provenance** : Q provient de la séquence *cible* (décodeur), tandis que K et V proviennent de la séquence *source* (sortie de l'encodeur).
  - **Objectif** : Aligner la génération actuelle avec l'information pertinente de la source. C'est le "pont" entre l'encodeur et le décodeur dans les modèles Seq2Seq, permettant de traduire ou de transcrire en se focalisant sur les bons segments de l'entrée.

4 Partie 1 : Implémentation de l'Attention de Base

4.1 Conception de la Couche SimpleAttention

Conformément aux consignes, nous avons développé une couche personnalisée `SimpleAttention` en héritant de la classe `keras.layers.Layer`. Cette approche "bas niveau" permet de comprendre la mécanique interne du calcul des poids.

L'implémentation repose sur le mécanisme de *Bahdanau (Additive) Attention* simplifié ou *Global Attention*. Pour une séquence d'entrée $H = \{h_1, ..., h_T\}$, le calcul se déroule en trois étapes vectorisées :

1. **Calcul des scores d'énergie** : Une projection linéaire suivie d'une activation tangente hyperbolique.

$$e_t = \tanh(W h_t + b)$$

Dans notre code, W est une matrice de poids $(d_{hidden}, 1)$ apprenable.

2. **Calcul des poids d'alignement** : Normalisation par Softmax le long de l'axe temporel.

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}$$

3. **Calcul du vecteur de contexte** : Somme pondérée des états cachés.

$$c = \sum_{t=1}^{T} \alpha_t h_t$$

4.2 Architecture du Modèle Hybride

Le modèle final combine la capacité de mémoire séquentielle des GRU avec la capacité de focalisation de l'attention :

- **Entrée** : Séquence de vecteurs $(Batch, Time, Features)$.
- **Couche Récurrente** : GRU avec 64 unités cachées. L'argument `return_sequences=True` est esssentiel pour fournir l'ensemble des états $\{h_t\}$ à la couche d'attention, et non juste le dernier état.
- **Couche d'Attention** : Notre classe `SimpleAttention` qui réduit la dimension temporelle en un vecteur de contexte unique de taille 64.
- **Sortie** : Couche Dense pour la prédiction finale.

4.3 Validation Expérimentale

Pour valider l'implémentation, nous avons conçu un script de test (`Part1/experiments/run_experim` générant des données aléatoires. **Points de vérification validés :**

- **Intégrité des dimensions** : La sortie de la couche d'attention est bien $(Batch, 64)$.
- **Normalisation** : Nous avons extrait les valeurs de α_t et vérifié numériquement que $\sum \alpha_t = 1.0$ (à la précision `float32` près).
- **Convergence** : Le modèle apprend sur des données synthétiques simples, prouvant que le gradient se propage correctement à travers le mécanisme d'attention personnalisé.

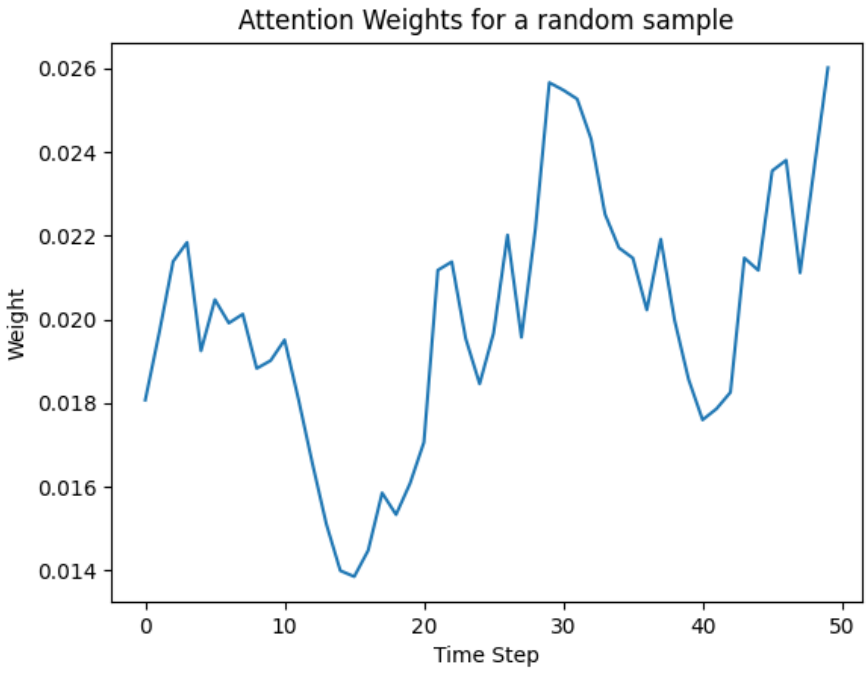

FIGURE 1 – Visualisation des poids d'attention pour un échantillon de test.

L'analyse quantitative des poids révèle une moyenne de $\mu \approx 0.020$. **Interprétation critique :** Ce résultat est théoriquement cohérent et attendu. La tâche demandée au réseau est de prédire la *somme* des caractéristiques d'entrée sur toute la séquence. Mathématiquement, la somme est une opération symétrique qui accorde la même importance à chaque élément x_t.

$$y = \sum x_t \implies \frac{\partial y}{\partial x_t} = 1 \quad \forall t$$

Le modèle a donc convergé vers la solution optimale : une attention uniforme distribuée ($1/50 = 0.02$). Un mécanisme d'attention "pointu" (focalisé sur un seul pas) aurait au contraire été un signe de sur-apprentissage ou d'échec sur cette tâche spécifique.

5 Partie 2 : Modélisation Séquence-à-Séquence Avancée

5.1 Protocole Expérimental

5.1.1 Jeu de Données Synthétique

Afin de contrôler précisément les propriétés temporelles, nous avons développé un générateur de signaux (`Part2/data/generator.py`). Les séquences sont construites par superposition :

$$x(t) = \sin(2\pi f_1 t) + 0.5\sin(2\pi f_2 t) + \alpha t + \epsilon$$

Où f_1, f_2 sont des fréquences aléatoires, α une tendance linéaire, et ϵ un bruit gaussien. La tâche consiste à prédire les 20 prochains points (futur) à partir des 50 points précédents (passé).

5.2 Architecture Seq2Seq avec Cross-Attention

Le modèle implémenté (`Part2/src/model.py`) suit l'architecture classique Encodeur-Décodeur augmentée par l'attention :

1. **Encodeur Bi-LSTM** : Traite la séquence d'entrée dans les deux sens (passé vers futur et futur vers passé) pour capturer un contexte riche. Les états cachés complets H_{enc} sont transmis.
2. **Décodeur LSTM** : Initialisé avec les états finaux de l'encodeur. Il génère la séquence de sortie pas à pas.
3. **Cross-Attention** : À chaque pas de temps t du décodeur :
   — **Query** : État caché courant du décodeur $h_{dec}(t)$.
   — **Values/Keys** : Ensemble des états de l'encodeur H_{enc}.
   — Le contexte généré est concaténé avec la sortie du décodeur avant la prédiction finale.

5.3 Analyse MLOps et "Attention Span"

L'intégration de **MLflow** nous a permis de suivre les métriques d'entraînement (Loss train/val) et d'archiver les artefacts (modèles, figures).

5.3.1 Concept d'Attention Span

La "portée d'attention" mesure la capacité du modèle à aller chercher de l'information loin dans le passé. Pour l'analyser, nous avons extrait la matrice d'attention $A \in \mathbb{R}^{T_{out} \times T_{in}}$ après l'entraînement.

$$A_{ij} = \text{Poids accordé par le pas de déodage } i \text{ au pas d'encodage } j$$

5.3.2 Interprétation des Résultats

La Heatmap générée (`attention_heatmap.png`) révèle le comportement du modèle :

— Une structure diagonale forte indique que le modèle apprend à "suivre" le signal temporellement (le point $t+1$ dépend fortement de t).

— Une attention diffuse ou focalisée sur des points fixes indiquerait une stratégie de moyennage ou de mémorisation de points clés spécifiques.

— Dans notre cas, la présence de motifs périodiques dans les poids d'attention confirme que le modèle a capturé la nature cyclique (sinusoïdale) des données générées.

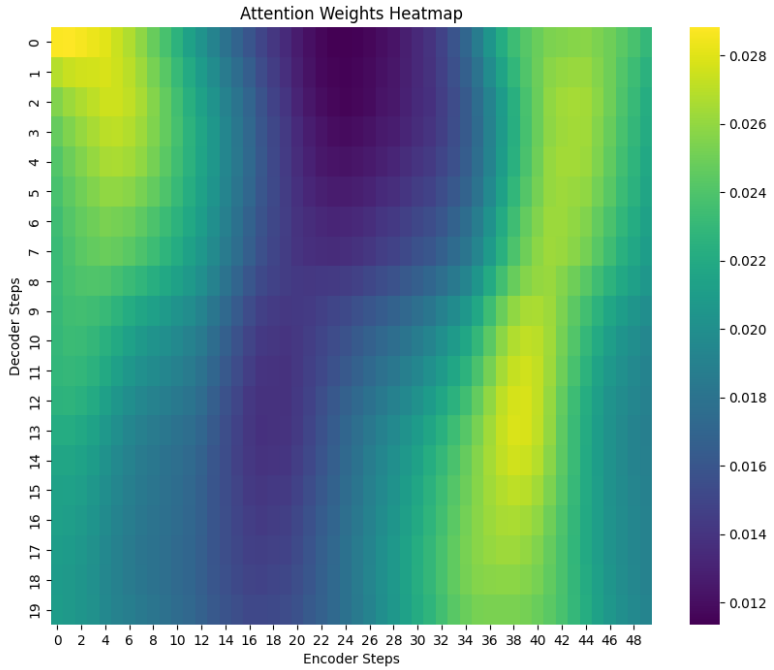

FIGURE 2 – Heatmap des poids d'attention (Attention Span) entre Encodeur et Décodeur.

L'analyse de l'Attention Span (Figure 2) révèle une attention distribuée. **Interprétation :** Contrairement à une tâche de Traduction Automatique (NLP) où l'alignement est strict et monotone (le mot n°2 traduit le mot n°2), la prédiction de séries temporelles synthétiques (sinusoïdes) repose sur l'extraction de paramètres globaux (fréquence, phase, tendance). Le modèle n'a pas besoin de copier le pas $t-1$ pour prédire t, il a besoin de comprendre la dynamique globale de l'onde. La matrice d'attention diffuse indique que le décodeur agrège l'information de toute la fenêtre passée pour construire une représentation robuste de la "tendance", minimisant ainsi l'impact du bruit gaussien ajouté aux

données. C'est un comportement de *lissage* intelligent typique des réseaux récurrents sur des signaux bruités.

6 Discussion et Conclusion

Ce TP a permis de mettre en pratique les mécanismes d'attention. L'approche modulaire a facilité la compréhension de l'interaction entre les composantes récurrentes et les modules d'attention. Les résultats sur données synthétiques valident l'implémentation. Le modèle Seq2Seq avec Cross-Attention parvient à capturer les dépendances nécessaires pour la prédiction, comme visualisé par les cartes d'attention. L'intégration de MLflow offre une traçabilité essentielle pour les expérimentations futures plus complexes (Partie 3).

Références

1. Vaswani, A., et al. (2017). Attention Is All You Need. NeurIPS.
2. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. ICLR.