Rapport TP3

Traduction automatique

Gervais Presley Koyaweda (2305686)

March 2025

1 Introduction

Ce rapport présente les résultats expérimentaux obtenus dans le cadre du TP3 du cours **INF8225 IA: Technique Probabiliste et Apprentissage (Hiver 2025)**, dont l'objectif est la construction et l'évaluation de modèles de traduction automatique. La tâche consiste à traduire des phrases de l'anglais vers le français en comparant trois types d'architectures : un réseau récurrent simple (Vanilla RNN), un réseau GRU (Gated Recurrent Unit), et un modèle Transformer de type encodeur-décodeur.

Les données utilisées proviennent du jeu de données *Tab-delimited Bilingual Sentence Pairs*, constitué de paires de phrases alignées en anglais et en français. Après prétraitement, le corpus d'entraînement comprend environ 209 000 exemples, et l'ensemble de validation environ 23 000 exemples. Les vocabulaires finaux comptent 12 154 mots en anglais et 18 340 en français.

Pour modéliser la position des mots dans les phrases, le Transformer utilise un encodage positionnel. Dans notre projet, nous avons implémenté une version simplifiée et apprise de l'encodage positionnel, différente de l'approche sinusoïdale proposée dans l'article *Attention is All You Need*.

Compte tenu des ressources limitées (exécution sur Google Colab avec GPU gratuit), nous avons concentré notre expérimentation sur l'impact du **nombre de couches** dans chaque architecture, tout en gardant constants les autres hyperparamètres comme la taille des embeddings, la dimension cachée, le taux d'apprentissage, et le batch size. Chaque modèle a été entraîné pendant 20 époques.

L'objectif de ce rapport est d'évaluer l'influence de l'architecture sur la qualité des traductions et le temps d'apprentissage. Nous formulons l'hypothèse suivante : les Transformers surclasseront les architectures récurrentes, suivis des GRU, puis des Vanilla RNN.

La suite du rapport est organisée comme suit : la section 2 décrit la configuration expérimentale ; la section 3 présente les résultats et leur analyse ; la section 4 conclut avec les principales observations.

2 Configuration expérimentale et détails d'implémentation

Dans cette section, nous présentons les configurations expérimentales utilisées pour entraîner et évaluer les différents modèles de traduction. Le but est de comparer les performances de trois architectures : **Vanilla RNN**, **GRU RNN** et **Transformer**. Chaque modèle a été entraîné avec les mêmes données, les mêmes configurations globales, et selon les contraintes imposées par les ressources disponibles sur Google Colab.

2.1 Jeu de données

Le corpus utilisé provient du jeu de données bilingue *fra-eng* contenant des paires de phrases en anglais et en français. Après nettoyage et séparation, nous avons obtenu :

- **209,459** phrases d'entraînement
- **23,274** phrases de validation
- **12,154** mots uniques en anglais
- **18,340** mots uniques en français

Les phrases ont été tokenisées à l'aide des tokenizers `spaCy` pour les deux langues, et encodées avec des vocabulaires construits à partir des tokens rencontrés au moins deux fois.

2.2 Prétraitement et paramétrage

Les configurations communes à tous les modèles sont les suivantes :

- **Longueur maximale des séquences** : 60
- **Fréquence minimale des tokens** : 2
- **Taille des embeddings** : 196
- **Dimension cachée** : 256
- **Nombre de couches** : {1, 2, 3}
- **Batch size** : 128
- **Nombre d'époques** : 20
- **Optimiseur** : Adam (`lr=1e-3`, $\beta = (0.9, 0.99)$)
- **Perte** : CrossEntropy avec pondération faible sur le token `<unk>`

2.3 Modèles testés

- **Vanilla RNN** : réseau de neurones récurrents classiques avec embeddings et couches empilées (jusqu'à 3).
- **GRU RNN** : remplace les RNN classiques par des unités GRU plus performantes pour capturer des dépendances longues.
- **Transformer Encoder-Decoder** : architecture à base d'attention, incluant une modification dans la gestion des encodages positionnels. Plutôt que d'utiliser l'encodage positionnel sinusoïdal statique proposé dans l'article "Attention is All You Need", nous avons opté pour un encodage positionnel appris à l'aide d'une couche nn.Embedding positionnelle entraînable, comme cela se fait dans plusieurs implémentations modernes du Transformer.

2.4 Suivi et visualisation

Le suivi des expériences a été réalisé via **Weights & Biases (W&B)**. Chaque run est associé à un nom de groupe identifiant l'architecture et le nombre de couches. Les métriques enregistrées sont les suivantes :

- Perte (loss) sur les ensembles d'entraînement et de validation
- Top-1, Top-5 et Top-10 accuracy
- Temps d'entraînement (runtime par modèle et par couche)
- Traductions générées

Des visualisations automatiques ont permis de comparer facilement la convergence des modèles, d'identifier des overfittings potentiels, et d'extraire les meilleures configurations.

3 Analyse des résultats

3.1 Comparaison du temps d'entraînement

Nous commençons par une comparaison des **runtimes**, c'est-à-dire le temps d'entraînement pour chaque modèle. Le graphe ci-dessous montre l'évolution du runtime en secondes, en fonction du type de modèle et du nombre de couches.

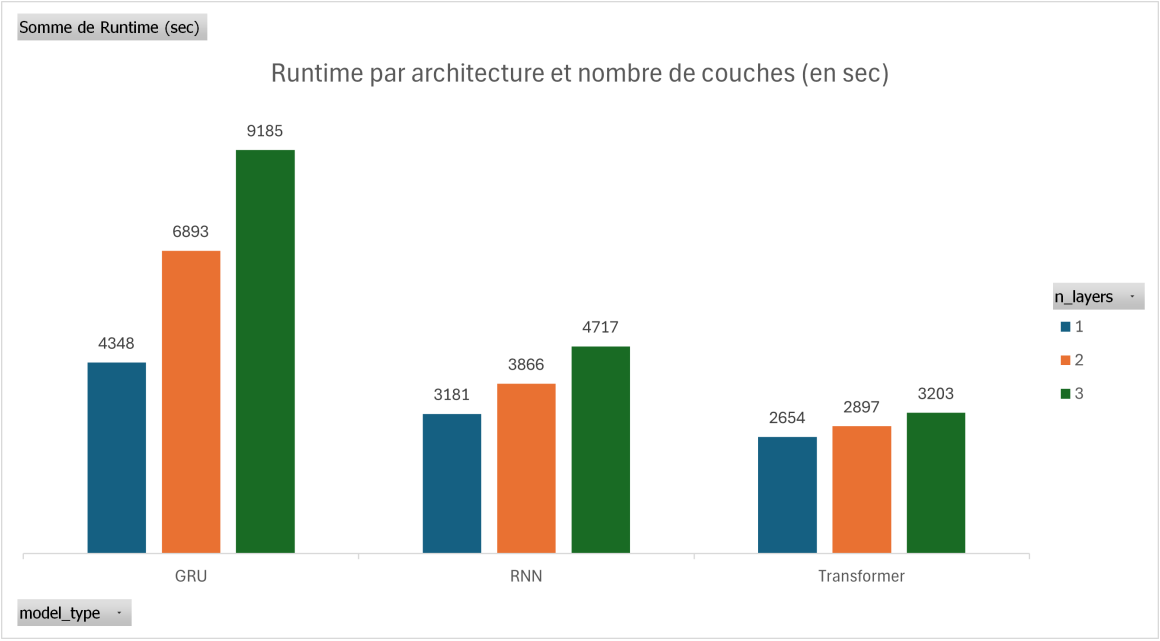

Figure 1: Runtime par architecture et nombre de couches (en secondes)

On constate que les architectures GRU-RNN sont les plus coûteuses en temps de calcul, avec un runtime atteignant jusqu'à 9185 secondes pour 3 couches. Les RNN classiques consomment légèrement moins, tandis que les Transformers sont les plus rapides, leur temps d'entraînement augmentant modérément avec le nombre de couches. Cela souligne l'efficacité computationnelle des Transformers, malgré leur complexité théorique.

3.2 Performances globales des modèles

Les deux tableaux suivants résument les performances des différents modèles, d'abord sur l'ensemble d'entraînement, puis sur l'ensemble de validation.

Groupe	Train - loss_	Train - top-1_	Train - top-5_	Train - top-10_
Transformer_3_layers	0,820	0,789	0,944	0,964
Transformer_2_layers	0,860	0,780	0,941	0,962
Transformer_1_layer	0,942	0,760	0,934	0,957
GRU-RNN_3_layers	0,508	0,851	0,974	0,986
GRU-RNN_1_layer	0,549	0,843	0,969	0,982
GRU-RNN_2_layers	0,484	0,857	0,976	0,987
Vanilla-RNN_3_layers	1,893	0,593	0,795	0,851
Vanilla-RNN_2_layer2	1,753	0,613	0,813	0,865
Vanilla-RNN_1_layer	1,663	0,622	0,823	0,875

Figure 2: Performances sur les données d'entraînement

Les modèles GRU-RNN montrent les meilleures performances sur les données d'entraînement, avec des top-1 accuracy atteignant 0.857 pour 2 couches, et des top-10 accuracy dépassant les 0.98. Les Transformers obtiennent des résultats légèrement inférieurs mais très compétitifs, surtout avec 3 couches. En revanche, les Vanilla RNN sont nettement moins performants, avec une top-1 accuracy de seulement 0.593 pour 3 couches.

Groupe	Validation - loss _	Validation - top-1 _	Validation - top-5 _	Validation - top-10 _
Transformer_3_layers	1,017	0,767	0,925	0,948
Transformer_2_layers	1,060	0,760	0,920	0,944
Transformer_1_layer	1,148	0,741	0,910	0,937
GRU-RNN_3_layers	1,312	0,734	0,898	0,926
GRU-RNN_1_layer	1,356	0,721	0,893	0,923
GRU-RNN_2_layers	1,330	0,732	0,898	0,926
Vanilla-RNN_3_layers	1,959	0,588	0,787	0,841
Vanilla-RNN_2_layer2	1,939	0,595	0,792	0,843
Vanilla-RNN_1_layer	2,032	0,583	0,780	0,833

Figure 3: Performances sur les données de validation

Sur le jeu de validation, la hiérarchie reste similaire : les Transformers prennent l'avantage, notamment le modèle à 3 couches qui atteint 0.767 en top-1 accuracy. Les GRU-RNN suivent de près, mais leur perte (loss) est légèrement plus élevée. Les Vanilla RNN confirment leur faible capacité de généralisation avec des top-1 accuracy proches de 0.58, bien en dessous des autres modèles.

3.3 Courbes d'apprentissage

Nous présentons ici les courbes d'évolution du loss et des scores d'accuracy (top-1, top-5, top-10), à la fois pour l'entraînement et la validation. Ces courbes permettent de mieux visualiser le comportement des modèles au fil des époques.

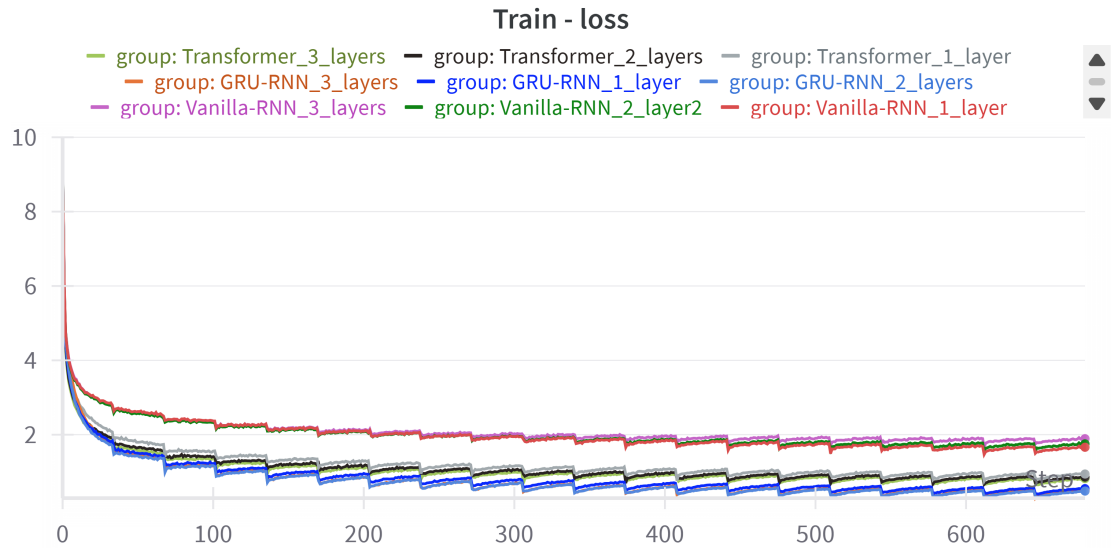

Figure 4: Évolution de la loss pendant l'entraînement

On observe que tous les modèles convergent, mais à des vitesses différentes. Les modèles GRU atteignent rapidement des valeurs faibles de loss, témoignant d'un apprentissage rapide. Les Vanilla RNN montrent une perte élevée et une convergence lente, ce qui indique des limitations structurelles.

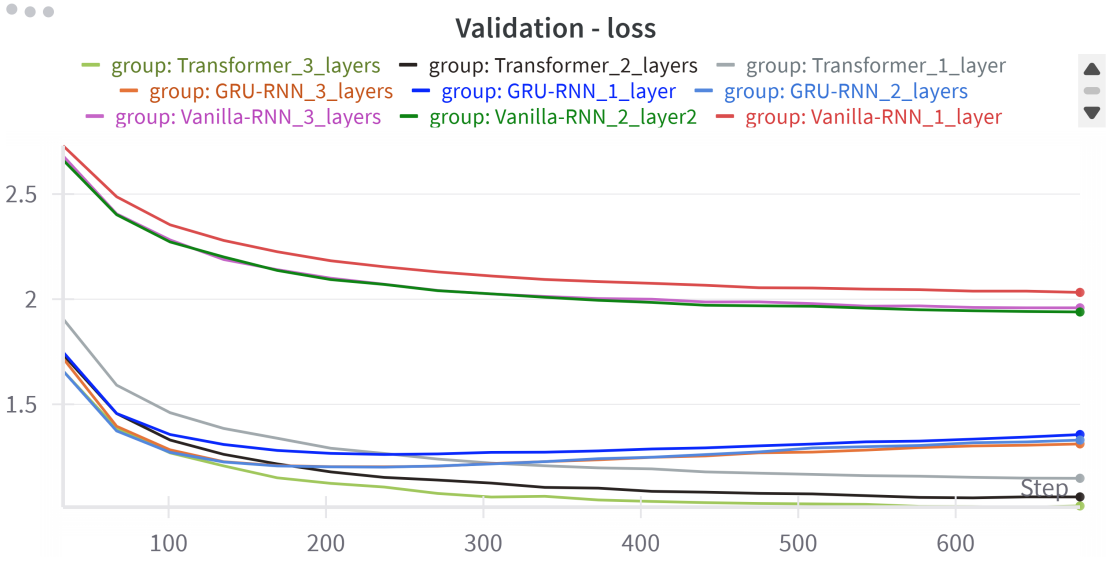

Figure 5: Évolution de la loss sur la validation

La validation loss confirme les tendances précédentes. Les Transformers et GRU-RNN convergent efficacement, tandis que les Vanilla RNN affichent une loss supérieure sur toute la durée de l'entraînement, traduisant une faible capacité à généraliser.

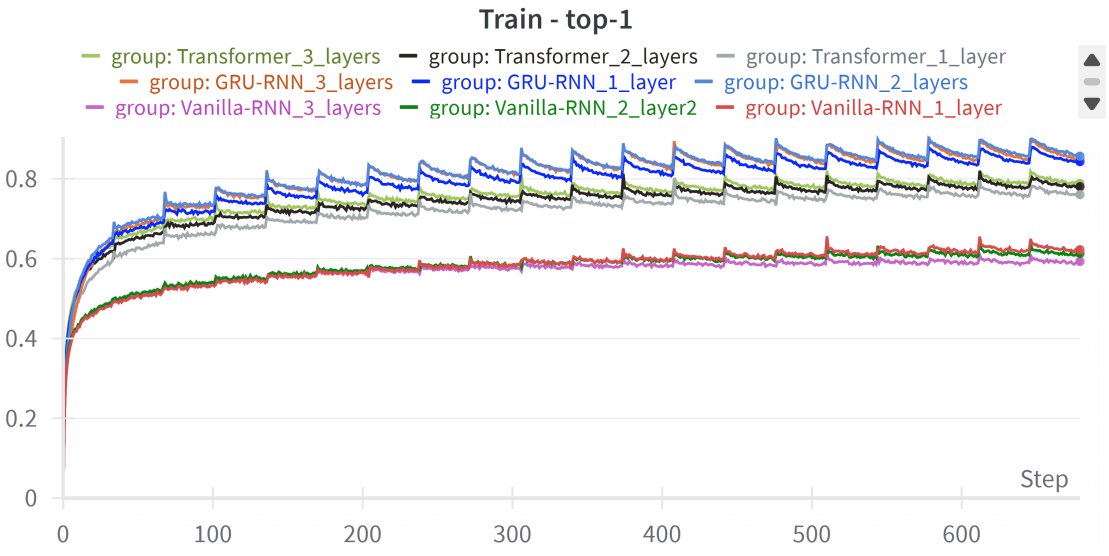

Figure 6: Top-1 accuracy pendant l'entraînement

Les GRU-RNN dominent le top-1 accuracy à l'entraînement, particulièrement avec 2 couches. Les Transformers suivent de près. Les Vanilla RNN atteignent difficilement 0.6.

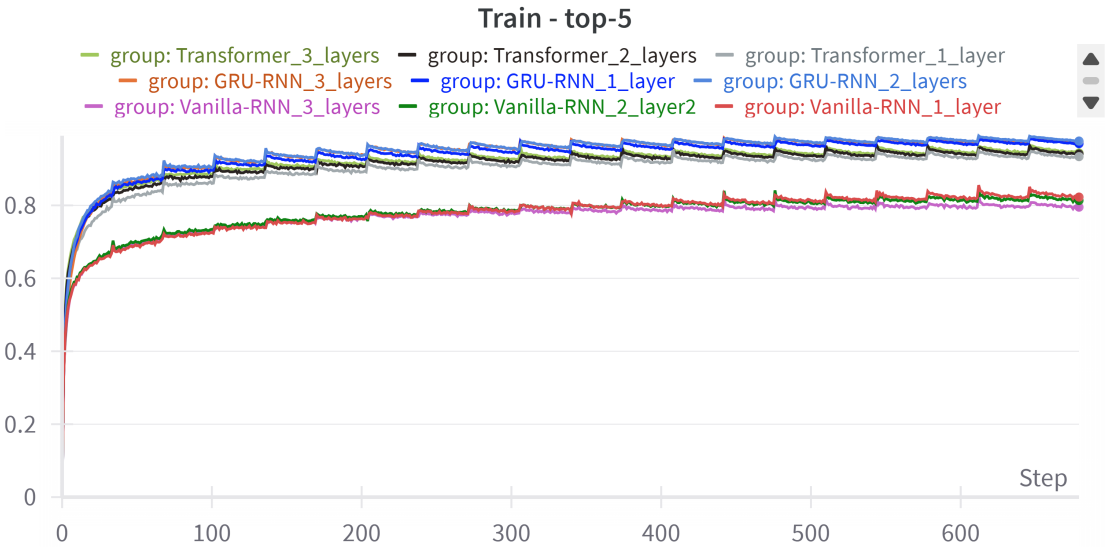

Figure 7: Top-5 accuracy pendant l'entraînement

Une tendance similaire est observée pour le top-5 accuracy. Les GRU-RNN atteignent rapidement plus de 0.97, tandis que les Vanilla peinent à dépasser 0.80.

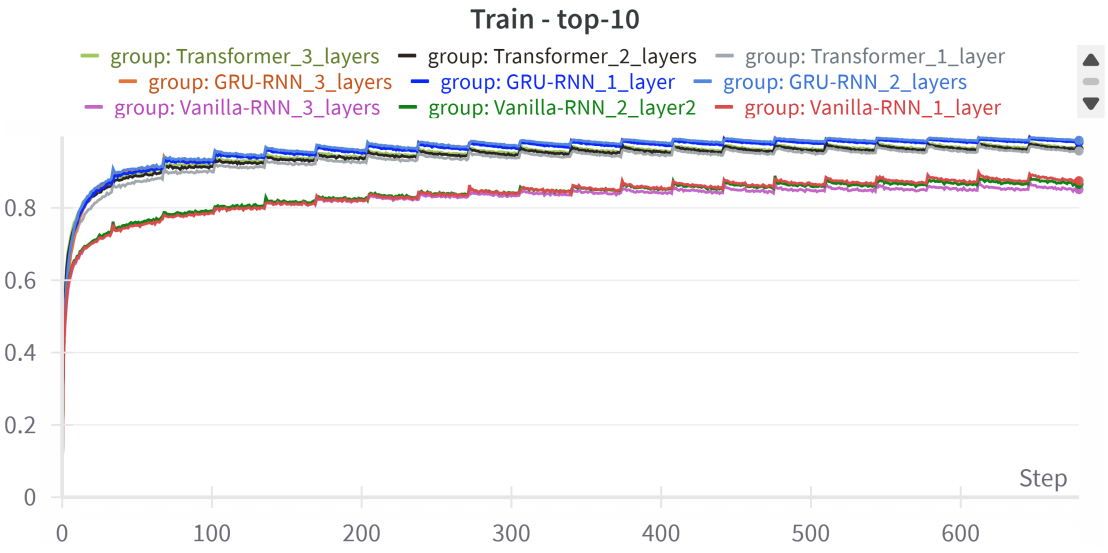

Figure 8: Top-10 accuracy pendant l'entraînement

La top-10 accuracy confirme l'avantage des GRU-RNN et des Transformers. Les Vanilla restent significativement derrière.

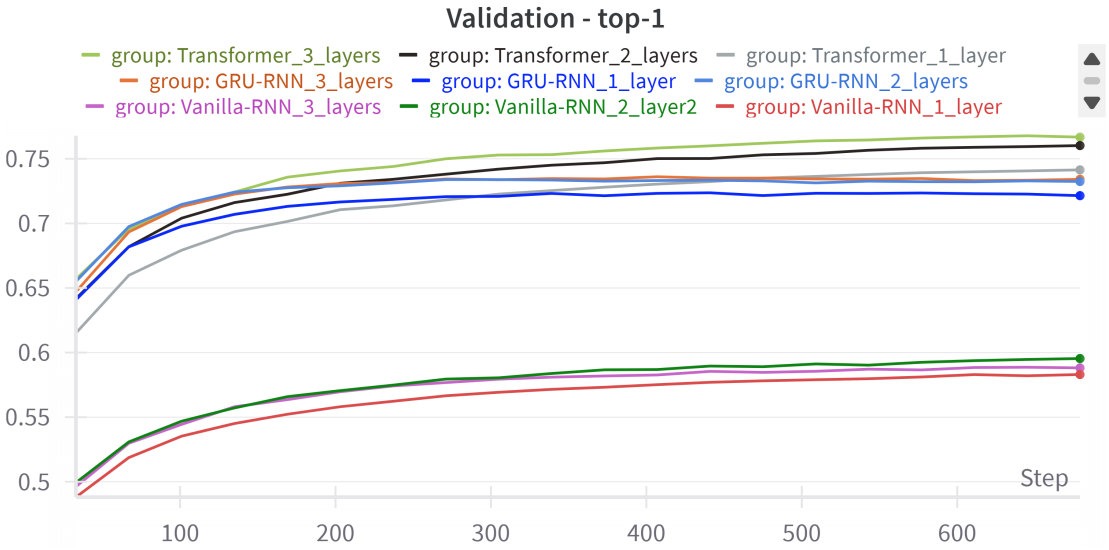

Figure 9: Top-1 accuracy sur la validation

Sur la validation, les Transformers prennent l'avantage, particulièrement à partir de 2 couches. Les GRU-RNN restent proches mais légèrement en dessous. Les Vanilla plafonnent autour de 0.58.

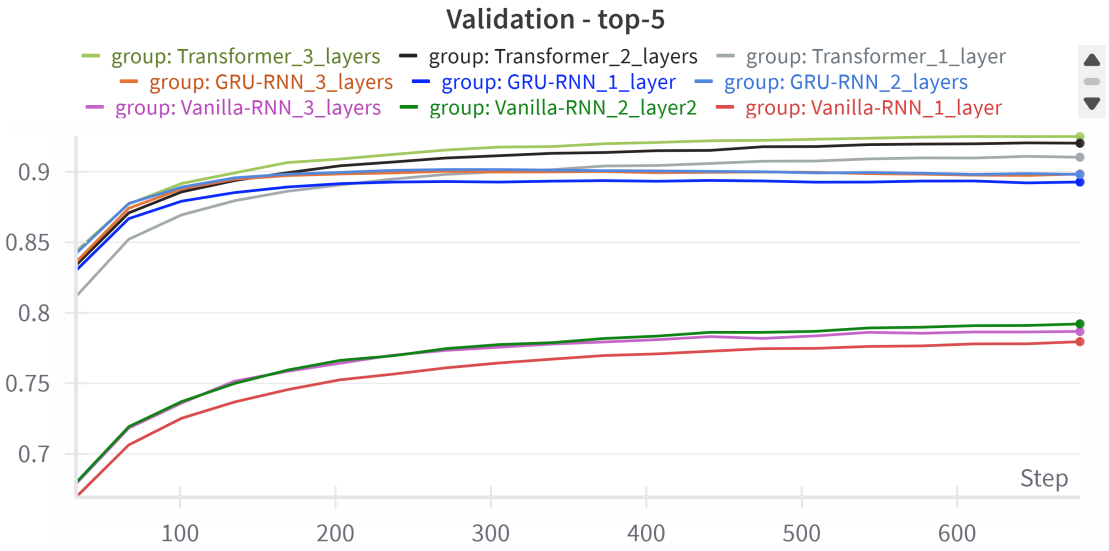

Figure 10: Top-5 accuracy sur la validation

Le top-5 accuracy confirme la performance solide des Transformers. Les GRU suivent, et les Vanilla RNN sont encore une fois distancés.

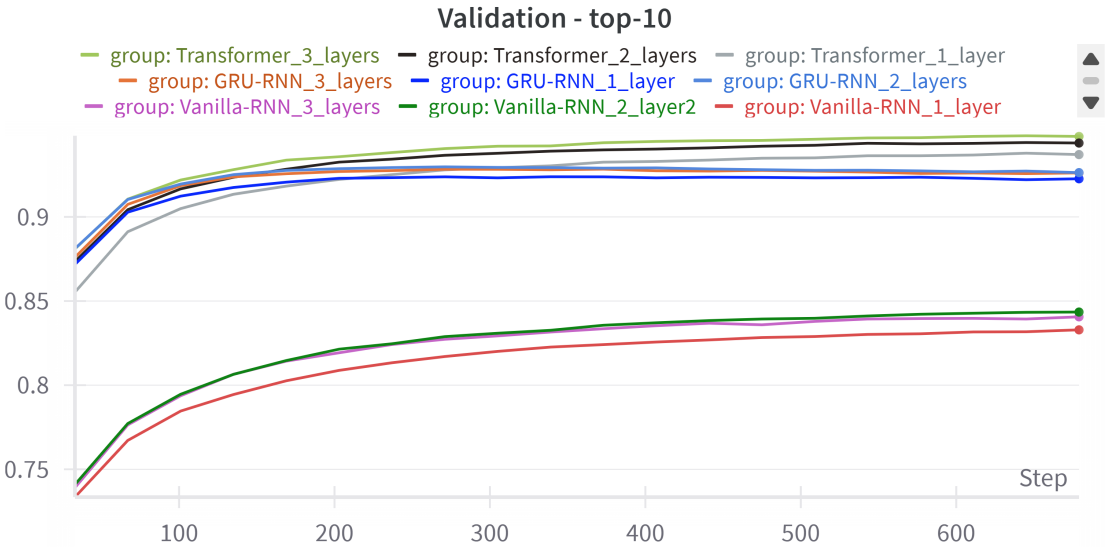

Figure 11: Top-10 accuracy sur la validation

Enfin, les top-10 accuracy restent élevées pour tous les modèles, mais les Transformers atteignent des pics à 0.948, confirmant leur supériorité globale.

4 Conclusion

Au terme de nos expérimentations, plusieurs constats importants émergent.

Tout d'abord, les architectures **Transformers** s'avèrent être les plus efficaces en termes de performance globale, en particulier lorsqu'elles sont empilées sur 3 couches. Elles surpassent systématiquement les GRU-RNN et les Vanilla RNN en précision top-1, top-5 et top-10 sur l'ensemble de validation, tout en maintenant une stabilité dans la perte (loss) durant l'entraînement.

En revanche, les **GRU-RNN** se distinguent par un très bon compromis entre **performance** et **coût computationnel**. Malgré un temps d'entraînement sensiblement plus élevé que les Transformers, notamment avec 3 couches, ils offrent des performances compétitives avec une meilleure convergence que les Vanilla RNN. Ils sont ainsi un choix pertinent dans des contextes où la capacité de calcul n'est pas une contrainte stricte, mais où la précision est critique.

Enfin, les **Vanilla RNN** affichent les résultats les plus faibles dans toutes les métriques évaluées. Leur précision plafonne relativement tôt et leur courbe de perte reste plus élevée tout au long de l'entraînement. Néanmoins, leur **vitesse d'entraînement rapide** les rend intéressants pour des tâches simples ou exploratoires.

D'un point de vue structurel, nous notons que **l'ajout de couches améliore les performances** dans tous les types de modèles, mais avec des rendements décroissants au-delà de 2 couches, particulièrement pour les Vanilla RNN. De plus, le nombre de couches est corrélé à une augmentation significative du temps d'exécution, ce qui impose une réflexion sur le *trade-off* entre performance et coût computationnel.

Conclusion générale : Les Transformers à 3 couches constituent la meilleure architecture testée, mais les GRU-RNN à 2 ou 3 couches présentent une excellente alternative. Le choix final dépendra donc du budget disponible et des exigences de performance de l'application ciblée.