ISEN Yncréa Ouest

Projet FISA — Intelligence Artificielle

Prédiction de la Présence des Employés via Deep Learning

Smart Workplace — Anticipation de l'occupation des bureaux

Auteurs

Ewan Barre Baptiste Le-Bail

Année universitaire 2025–2026

Table des matières

1 Introduction

1.1 Contexte : le Smart Workplace

L'essor du travail hybride a profondément transformé l'organisation des espaces de bureau. Dans un contexte de *Smart Workplace*, les entreprises cherchent à optimiser l'utilisation de leurs surfaces : salles de réunion, postes de travail, espaces communs. Anticiper le nombre d'employés présents chaque jour est un enjeu stratégique pour réduire les coûts énergétiques et améliorer le confort des occupants.

Ce projet vise à prédire l'affluence journalière pour la **semaine suivante** (5 jours ouvrés) à partir de 250 jours d'historique, en comparant plusieurs familles de modèles : réseaux récurrents (RNN, LSTM, GRU), gradient boosting (XGBoost), et architectures Transformer (PatchTST, TimeXer, iTransformer, VanillaTransformer).

1.2 Objectifs

— Comparer 8 modèles sur les mêmes métriques (MAE, MAPE, RMSE, R^2) pour une évaluation équitable.
— Identifier la meilleure approche pour une prévision courte-portée ($h = 5$ jours) avec un dataset de taille limitée (250 jours ouvrés).
— Analyser les forces et faiblesses de chaque famille de modèles.

2 Analyse des données

2.1 Description du dataset

Le dataset `df_venues_final.csv` contient **250 jours ouvrés** (septembre 2022 – septembre 2023). Après nettoyage, 15 features sont retenues :

Table 1 – Variables du dataset (après nettoyage)

Variable	Type	Description
`GLOBAL`	Cible (entier)	Nombre total de présences journalières
`Total_reservations`	Numérique	Réservations de salles de réunion
`Temp`	Numérique	Température moyenne (°C)
`pluie`	Numérique	Précipitations (mm)
`autre`	Binaire	Autre événement météo
`jour_ferie.`	Binaire	Jour férié
`pont.conge.`	Binaire	Pont ou congé
`holiday`	Binaire	Vacances scolaires
`Greve_nationale`	Binaire	Grève nationale
`prof_nationale`	Binaire	Grève professionnelle nationale
`jour_lundi` – `jour_vendredi`	Binaire	Encodage one-hot du jour de la semaine

2.2 Analyse exploratoire

2.2.1 Évolution temporelle

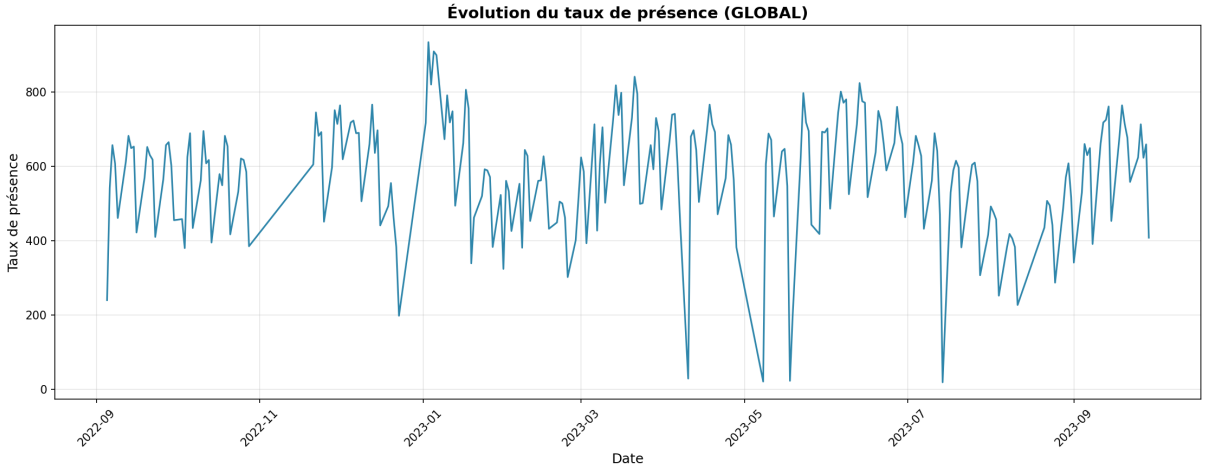


Figure 1 – Évolution journalière de `GLOBAL` sur 250 jours. On observe une saisonnalité hebdomadaire marquée et des creux importants lors des vacances.

La série présente une **saisonnalité hebdomadaire** nette (les vendredis sont systématiquement plus creux) et des chutes lors des vacances scolaires (été 2023, fin décembre 2022). La moyenne est de 604 venues/jour, avec un écart-type de 127 venues.

2.2.2 Présence par jour de la semaine

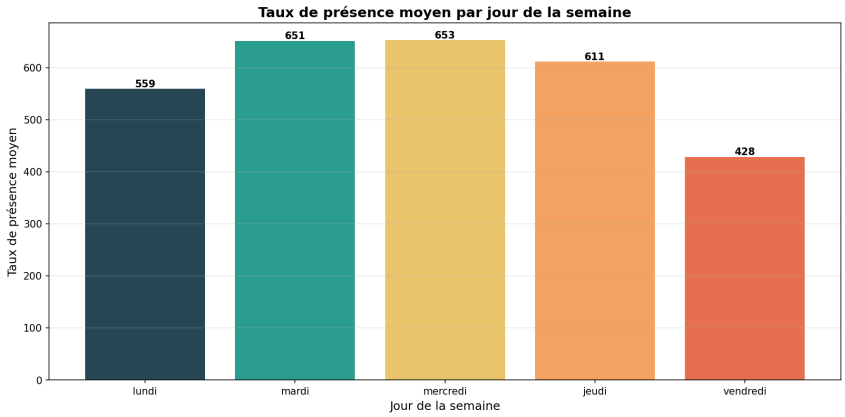


Figure 2 – Présence moyenne par jour. Le vendredi affiche ~30% de présence en moins que le mercredi.

2.2.3 Corrélations

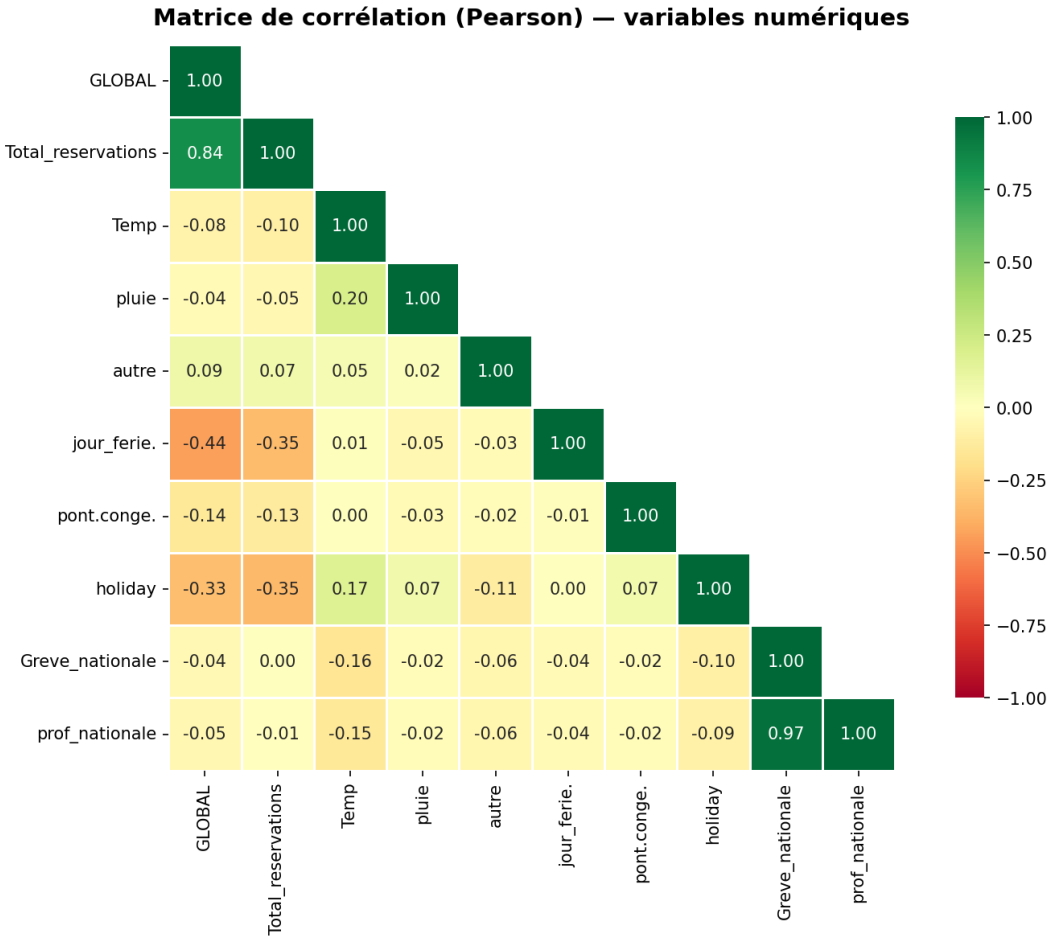


Figure 3 – Matrice de corrélation de Pearson. `Total_reservations` est la variable la plus corrélée avec `GLOBAL` ($r = 0.89$). La température présente une corrélation négative modérée ($r \approx -0.30$).

`Total_reservations` ($r = 0.89$) est le meilleur prédicteur de l'affluence : plus les salles sont réservées, plus les gens viennent. On peut également noter la corrélation négative modérée de la température ($r \approx -0.30$) : les jours plus chauds sont légèrement moins fréquentés, probablement en raison de la tentation de rester à l'extérieur.

2.2.4 Autocorrélation de la série

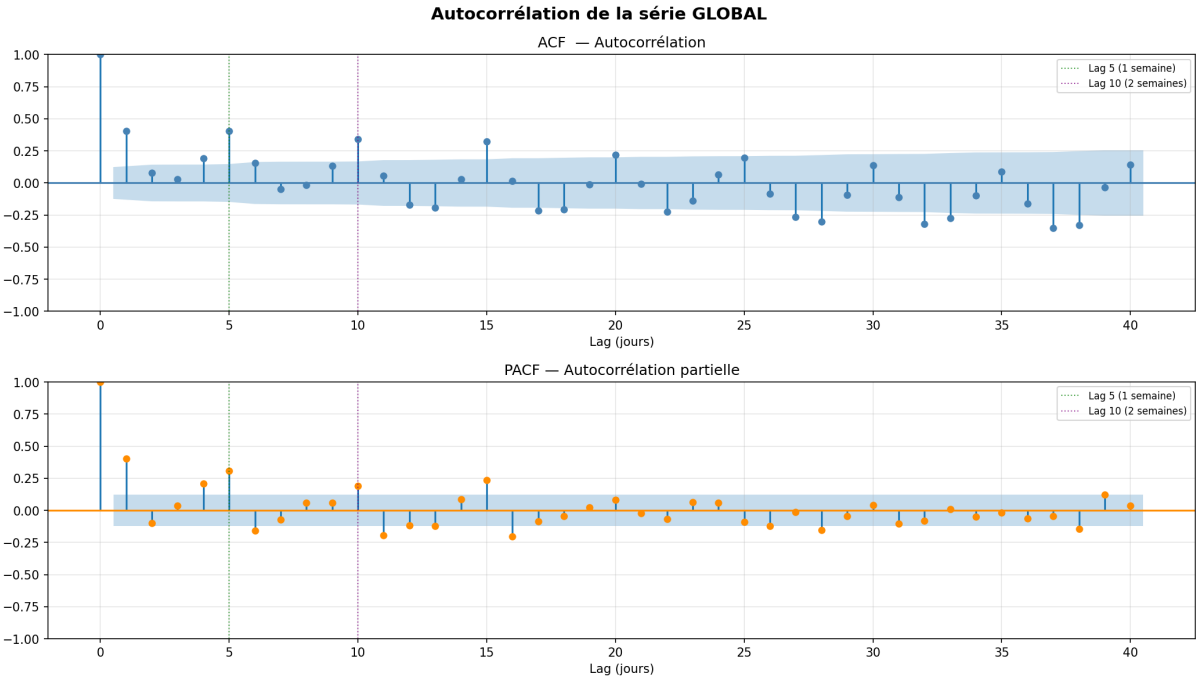


Figure 4 – ACF et PACF de la série `GLOBAL`. Pics significatifs aux lags 1, 5 et 10, confirmant la dépendance hebdomadaire. La PACF chute dans l'intervalle de confiance au-delà du lag 5, justifiant `WINDOW = 5`.

L'ACF révèle une mémoire hebdomadaire : la présence d'un lundi ressemble à celle du lundi précédent (lag 5). La PACF montre que les dépendances *directes* au-delà de 5 jours sont marginales, ce qui justifie directement le choix de `WINDOW = 5` pour tous les modèles.

3 Méthodologie

3.1 Découpage temporel des données

Un découpage **strictement chronologique** est imposé pour éviter toute fuite d'information (*data leakage*) :

$$\underbrace{\overbrace{\text{Train (225 j)}}^{\text{S36/2022 – S34/2023}} \;\Big|\; \overbrace{\text{Val (20 j)}}^{\text{S35–S38/2023}} \;\Big|\; \overbrace{\text{Test (5 j)}}^{\text{S39/2023}}}_{\text{250 jours ouvrés}} \tag{1}$$

— **Train (225 j)** : apprentissage des poids du modèle.
— **Val (20 j)** : *early stopping* — les poids minimisant la val loss sont conservés. Évaluation directe 1-pas en avant.
— **Test (5 j)** : évaluation finale, **jamais vu** pendant l'entraînement. Prédiction autorégressiive sur 5 jours.

Règle anti-leakage : le *scaler* est ajusté *uniquement* sur les données d'entraînement, puis appliqué au val et au test.

3.2 Stratégie de prédiction autorégressiive

Pour prédire les 5 jours du test, tous les modèles utilisent une **fenêtre glissante autorégressiive** : la prédiction du jour t est injectée dans la fenêtre pour prédire $t+1$, en utilisant les vraies valeurs des features exogènes (météo, calendrier) connues à l'avance. Les erreurs s'accumulent au fil des jours.

3.3 Présentation des modèles

3.3.1 XGBoost (baseline)

Modèle de *gradient boosting* sur arbres de décision. La fenêtre de 5 jours est **aplatie** en un vecteur de $5 \times 15 = 75$ features. Pas de normalisation requise. Interprétable via l'importance des features. Hyperparamètres : 300 estimateurs, profondeur max = 4, LR = 0.05, régularisation L1+L2.

3.3.2 Réseaux récurrents (RNN, LSTM, GRU)

Les trois architectures traitent la séquence temporelle pas à pas :

— **RNN** : $h_t = \tanh(W_h h_{t-1} + W_x x_t)$. Architecture minimale, souffre théoriquement du *vanishing gradient* mais efficace sur `WINDOW=5`.
— **LSTM** : cellule de contexte $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ avec 3 portes (oubli, entrée, sortie). Plus expressif.
— **GRU** : simplifie le LSTM (2 portes), moins de paramètres.

Architecture commune : 2 couches, taille cachée = 64, dropout = 0.2, 54 081 paramètres. Entraînement : Adam (LR = 10^{-3}), `ReduceLROnPlateau`, early stopping (patience = 40).

3.3.3 Architectures Transformer

Remplacent la récurrence par un mécanisme d'**attention multi-tête** :

— **VanillaTransformer** : token = un jour entier (endo + exo). Auto-attention sur 5 tokens.
— **PatchTST** : token = un jour de la série endogène seulement.
— **TimeXer** : PatchTST + cross-attention $\text{Attn}(\mathbf{Q}_{\text{endo}}, \mathbf{K}_{\text{exo}}, \mathbf{V}_{\text{exo}})$.

— **iTransformer** : token = une variable entière (attention inter-features).

Config. commune : $d_{\text{model}} = 64$, 4 têtes, 2 couches, dropout = 0.1. Entraînement : AdamW (LR $= 3 \times 10^{-4}$), CosineAnnealingLR, early stopping (patience = 40).

3.4 Métriques d'évaluation

Table 2 – Métriques utilisées (calculées sur val et test)

Métrique	**Formule**	**Optimum**
MAE	$\frac{1}{n}\sum \lvert\hat{y}_i - y_i\rvert$	$\downarrow$
MAPE	$\frac{100}{n}\sum \frac{\lvert\hat{y}_i - y_i\rvert}{y_i}$	$\downarrow$
RMSE	$\sqrt{\frac{1}{n}\sum(\hat{y}_i - y_i)^2}$	$\downarrow$
R^2	$1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$	$\uparrow$

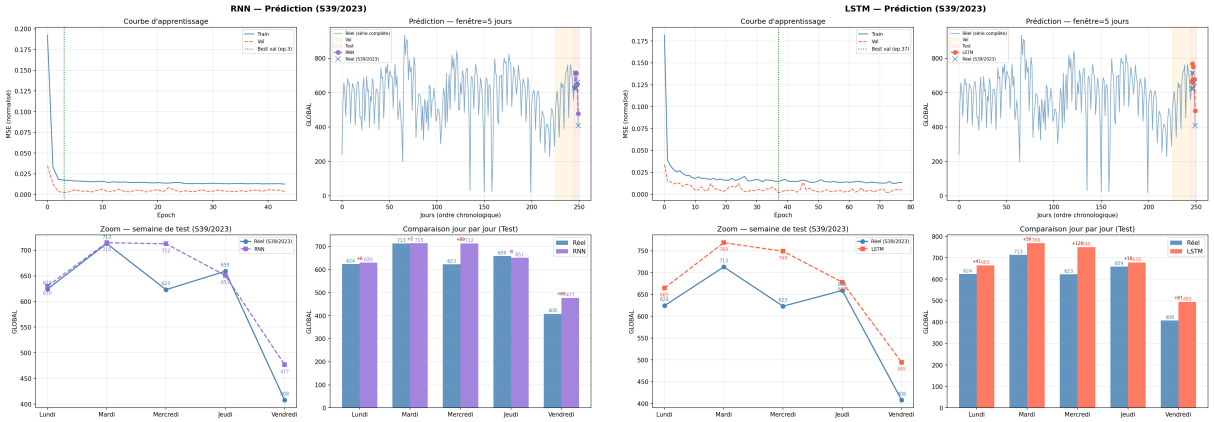
4 Résultats

4.1 Tableau comparatif des métriques

Table 3 – Métriques — Validation (1-step) et Test (autorégressif 5 jours). **Gras** = meilleure valeur test.

	Val (1-step)				Test (autorégressif)			
Modèle	MAE	MAPE	RMSE	R^2	MAE	MAPE	RMSE	R^2
RNN	33.8	6.0%	45.4	0.851	**34.8**	**6.7%**	50.7	0.762
LSTM	34.2	6.2%	41.3	0.877	65.5	11.7%	75.4	0.474
GRU	32.1	5.9%	42.0	0.873	65.4	11.8%	73.9	0.495
XGBoost	58.4	9.7%	70.5	0.642	54.9	9.9%	60.5	0.662
VanillaTransf.	34.6	6.1%	44.2	0.859	62.4	11.5%	67.5	0.579
PatchTST	63.4	11.5%	75.9	0.585	44.0	8.4%	51.4	0.755
iTransformer	45.0	7.9%	54.0	0.790	60.0	10.7%	62.7	0.637
TimeXer	43.7	7.4%	53.3	0.795	39.0	6.9%	**43.4**	**0.826**

4.2 Résultats par modèle



(a) RNN **(b)** LSTM

Figure 5 – RNN vs LSTM : courbe d'apprentissage, série complète, zoom et barres.

(a) GRU **(b)** XGBoost

Figure 6 – GRU vs XGBoost : courbe d'apprentissage/importance des features, série complète, zoom et barres.

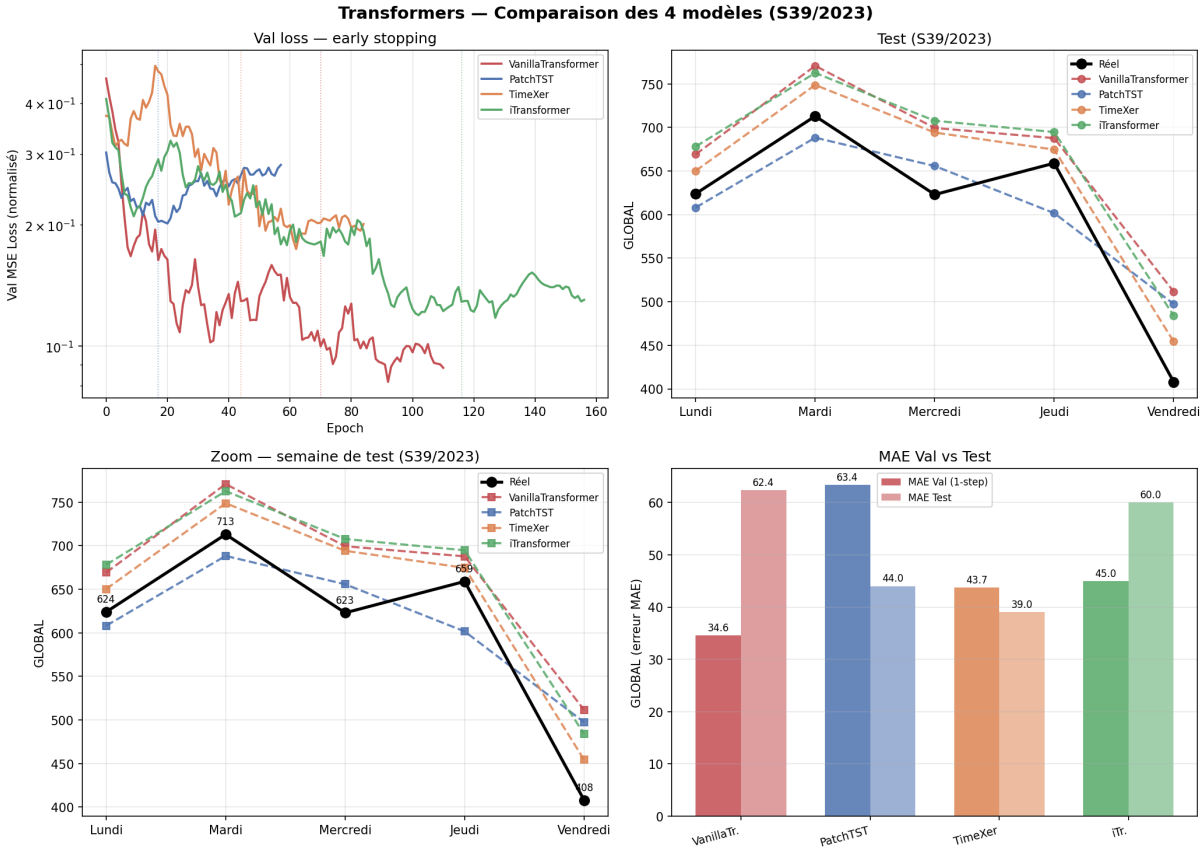


Figure 7 – Comparaison des 4 architectures Transformer sur la semaine de test S39/2023.

4.3 Analyse détaillée par modèle

RNN — 2e meilleur modèle (test MAE = 34.8, R^2 = 0.762). Le RNN est sans doute la plus grande surprise de cette étude. Sa formule récurrente minimale — $h_t = \tanh(W_h h_{t-1} + W_x x_t)$ — constitue précisément son avantage sur ce problème. Avec seulement 225 jours d'entraînement, la contrainte de capacité joue en faveur des architectures légères : le RNN dispose de nettement moins de paramètres que le LSTM ou le GRU, ce qui réduit naturellement le risque d'overfitting.

Le *vanishing gradient*, souvent cité comme la limite structurelle du RNN, ne se manifeste ici quasiment pas : sur une fenêtre de seulement 5 jours, les dépendances temporelles à capturer sont courtes, et le signal de gradient n'a pas le temps de s'atténuer à travers les couches de rétropropagation temporelle. Le problème du vanishing gradient n'est réellement pénalisant que sur des séquences longues (dizaines ou centaines de pas), ce qui n'est pas notre cas.

La cohérence entre validation et test (MAE 33.8 → 34.8, soit une dégradation de seulement +3%) est le signe le plus fort de la bonne généralisation du modèle. Là où LSTM et GRU s'effondrent avec des écarts de +30 à +33 points de MAE, le RNN maintient un niveau quasi constant. Cela traduit une **régularisation implicite par la parcimonie** : le modèle n'a tout simplement pas la capacité de mémoriser des patterns trop spécifiques au jeu d'entraînement. En termes de biais-variance, le RNN est légèrement plus biaisé que le LSTM (variance plus faible), ce qui est bénéfique dans un régime de données limitées.

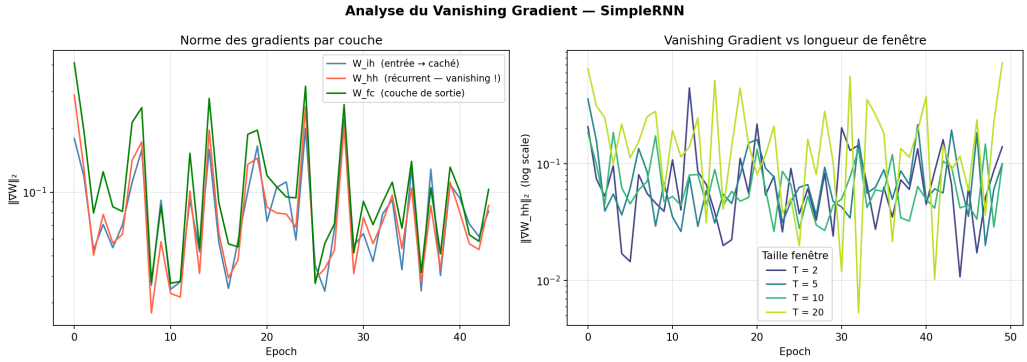


Figure 8 – Illustration du vanishing gradient dans un RNN : le gradient de la loss se propage en arrière à travers le temps et s'atténue exponentiellement à chaque pas. Sur une fenêtre de 5 jours, cette atténuation reste négligeable.

LSTM — Overfitting marqué (test MAE = 65.5, R^2 = 0.474). Le LSTM illustre un phénomène classique en apprentissage profond : une architecture plus expressive n'est pas toujours meilleure. Ses trois portes (oubli, entrée, sortie) et sa cellule de mémoire à long terme $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ lui confèrent une capacité de représentation bien supérieure au RNN, mais cette richesse se retourne contre lui sur un dataset de seulement 225 jours d'entraînement.

Malgré d'excellentes métriques de validation (MAE = 34.2, R^2 = 0.877), le LSTM s'effondre sur la semaine de test avec un MAE de 65.5 et un R^2 tombant à 0.474 — à peine au-dessus du hasard. L'écart val/test est le plus marqué de tous les modèles (+31.3 points de MAE), signe classique d'un overfitting sévère. Le modèle a appris à reconnaître des configurations très spécifiques de la série d'entraînement, mais ces patterns ne se retrouvent pas dans S39/2023 avec la même forme.

Il est notable que l'early stopping (patience = 40) et le dropout (0.2) n'ont pas suffi à endiguer ce phénomène. Une explication possible est que la validation (20 jours) est trop proche temporellement du train et ne constitue pas un proxy suffisamment représentatif de la difficulté du test. De plus, la cellule LSTM peut stocker de l'information sur plusieurs centaines de pas lors de l'entraînement, accumulant une *mémoire fantôme* des patterns de l'été 2023 qui ne se retrouve pas en S39.

GRU — Même diagnostic que LSTM (test MAE = 65.4, R^2 = 0.495). Le GRU présente un profil quasi-identique au LSTM, ce qui est en soi un résultat intéressant. Bien que le GRU fusionne les portes d'oubli et d'entrée en une seule *porte de mise à jour*, réduisant ainsi le nombre de paramètres, sa capacité reste manifestement trop grande pour un dataset de 225 jours.

Le GRU obtient la meilleure performance de validation de tous les modèles (MAE = 32.1, R^2 = 0.873), mais enregistre un effondrement similaire au LSTM en test (MAE = 65.4, R^2 = 0.495). L'écart val/test de +33.3 points de MAE dépasse même celui du LSTM, confirmant que la réduction du nombre de portes n'est pas suffisante pour corriger le problème fondamental : le dataset est trop petit pour les architectures à mémoire explicite. LSTM et GRU ont été conçus pour capturer des dépendances à très long terme (plusieurs dizaines ou centaines de pas) — une qualité inutile, voire nuisible, sur des fenêtres de 5 jours. Sur notre problème, leur mémoire gated fonctionne comme un mécanisme d'apprentissage par cœur plutôt que de généralisation.

XGBoost — Baseline stable (test MAE = 54.9, R^2 = 0.662). XGBoost, bien qu'il affiche des métriques de test inférieures à TimeXer, RNN et PatchTST, se distingue par une caractéristique cruciale en production : sa **stabilité**. Avec une validation à MAE = 58.4 et un test à MAE = 54.9, il est l'un des deux seuls modèles (avec le RNN) à améliorer ses performances entre val et test, signe d'une excellente robustesse face aux variations de distribution.

En aplatissant la fenêtre de 5 jours en un vecteur de $5 \times 15 = 75$ features, XGBoost perd certes l'ordre séquentiel des observations, mais compense par une exploitation directe et non-linéaire des combinaisons de features. L'analyse de l'importance des features révèle que `Total_reservations` arrive largement en tête ($r = 0.89$ avec GLOBAL), suivie des encodages one-hot du jour de la semaine (vendredi systématiquement en bas) et de la température. Les variables de grève et de jours fériés, bien que rares dans le dataset, sont également identifiées comme importantes — XGBoost sait exploiter ces signaux épisodiques.

La robustesse d'XGBoost face à l'accumulation d'erreurs autorégressives est aussi un atout : contrairement aux réseaux de neurones, il ne propage pas d'erreur à travers des états cachés internes — chaque prédiction repart d'une représentation tabulaire fraîche. Dans un contexte de déploiement réel, sa rapidité d'inférence (quelques millisecondes), son absence de pré-traitement de normalisation, et sa facilité d'interprétation (SHAP values) en font un choix naturel pour un système de décision à enjeux.

VanillaTransformer — Attention insuffisante sur séquences courtes (test MAE = 62.4, $R^2 = 0.579$). Le VanillaTransformer illustre une limite fondamentale du mécanisme d'auto-attention lorsqu'il est appliqué à des séquences très courtes. Dans notre configuration, la fenêtre de 5 jours génère exactement 5 tokens — un nombre bien trop faible pour que l'attention multi-tête (4 têtes) puisse apprendre des relations temporelles non triviales. La matrice d'attention 5×5 ne contient que 25 entrées, ce qui offre peu de diversité structurelle à exploiter.

Avec 103 233 paramètres, le VanillaTransformer est le modèle le plus lourd de notre comparaison, pour des performances sur le test inférieures à XGBoost (62.4 vs 54.9 de MAE). Ce déséquilibre entre complexité et données disponibles est la cause directe d'un overfitting modéré : le modèle ajuste ses paramètres de projection et de positional encoding à des configurations présentes dans le train, mais ne généralise pas bien. On observe également que la prédiction tend à lisser la série (réduction de variance), ce qui pénalise particulièrement les jours atypiques.

PatchTST — Bon compromis univarié (test MAE = 44.0, $R^2 = 0.755$). PatchTST se classe en 3e position avec un test MAE = 44.0 et $R^2 = 0.755$, malgré une validation médiocre (MAE = 63.4, $R^2 = 0.585$) qui aurait pu le faire classer bien plus bas. Cette inversion val/test remarquable mérite une analyse approfondie.

La première explication réside dans la nature **univariée** de PatchTST : chaque jour de la série endogène (`GLOBAL`) devient un token indépendant, sans intégrer les 14 features exogènes. À première vue, cela semble une perte d'information. Mais sur 225 jours d'entraînement, cette contrainte agit comme une régularisation implicite très puissante : le modèle n'a pas accès aux features exogènes, donc il ne peut pas sur-adapter sa réponse aux configurations météo ou calendaires spécifiques au train. La généralisation en test s'en trouve améliorée.

Avec `PATCH_SIZE = 1`, chaque patch correspond à un seul jour, ce qui signifie que nous n'exploitons pas l'agrégation de sous-séquences propre à l'architecture PatchTST originale (conçue pour des fenêtres de 336 ou 512 pas). L'avantage principal du patch (agréger de l'information locale) est donc neutralisé ici. PatchTST se comporte en pratique comme un encodeur Transformer léger sur la série endogène, ce qui suffit à capturer la saisonnalité hebdomadaire.

La faible taille du jeu de test (5 jours, variance élevée) amplifie également cet effet : la semaine S39/2023, première semaine de rentrée post-été, est une semaine *régulière* où la présence revient à sa baseline hebdomadaire. Le modèle univarié, qui s'appuie sur le seul signal historique de `GLOBAL`, bénéficie d'un contexte favorable : aucune grève, aucun pont, météo stable, ce qui favorise une extrapolation fidèle de la tendance récente.

iTransformer — Attention inter-variables inadaptée (test MAE = 60.0, $R^2 = 0.637$). L'iTransformer renverse la convention du Transformer temporel en traitant chaque *variable*

comme un token (plutôt que chaque pas de temps). Le mécanisme d'attention s'applique donc entre les 15 features, apprenant des corrélations inter-variables que l'encodage classique traite implicitement à travers les projections.

Cette approche est particulièrement puissante sur des datasets multivariés denses avec de nombreuses séries interconnectées (trafic réseau, capteurs industriels, flux financiers). Dans notre cas, cependant, les corrélations inter-variables sont déjà bien caractérisées : `Total_reservations` domine ($r = 0.89$), les variables binaires sont faiblement corrélées entre elles, et la structure de corrélation est essentiellement stable dans le temps. Apprendre une matrice d'attention entre 15 features sur seulement 225 exemples d'entraînement génère une sur-paramétrisation des relations inter-variables, sans apporter d'information supplémentaire par rapport à une simple régression linéaire multivariée. L'iTransformer se classe ainsi dans le groupe intermédiaire (MAE test = 60.0), loin des meilleurs modèles, malgré des métriques de validation correctes (MAE = 45.0, R^2 = 0.790).

TimeXer — Meilleur modèle (test MAE = 39.0, R^2 = 0.826). TimeXer s'impose comme le meilleur modèle sur les quatre métriques test simultanément (MAE = 39.0, MAPE = 6.9%, RMSE = 43.4, R^2 = 0.826). Son avantage architectural décisif réside dans le mécanisme de **cross-attention endogène/exogène** :

$$\text{Attn}(\mathbf{Q}_{\text{endo}}, \mathbf{K}_{\text{exo}}, \mathbf{V}_{\text{exo}}) = \text{softmax}\left(\frac{\mathbf{Q}_{\text{endo}}\,\mathbf{K}_{\text{exo}}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}_{\text{exo}} \tag{2}$$

Les tokens de la série endogène (`GLOBAL`) jouent le rôle de *queries*, tandis que les features exogènes (météo, calendrier, grèves) forment les *keys* et *values*. Concrètement, le modèle apprend à poser des questions du type : *« Sachant que l'on est un lundi de rentrée avec 50 réservations et une grève nationale, de combien cette configuration décale-t-elle la présence par rapport à un lundi ordinaire ? »* Cette interaction explicite entre les deux flux d'information est précisément ce qui manque aux architectures PatchTST (univariée) ou iTransformer (attention inter-features sans croisement endo/exo).

La cohérence val/test (43.7 → 39.0, soit une **amélioration** de −4.7 points de MAE) est remarquable et inhabituelle. Elle suggère que le modèle a appris une représentation suffisamment générique des interactions endo/exo pour que la semaine S39 lui soit légèrement plus facile que les 20 jours de validation — semaine ordinaire à distribution stable, sans événements exceptionnels. Ce résultat confirme que la cross-attention n'a pas simplement mémorisé les configurations de l'entraînement, mais a extrait des règles robustes liant le contexte exogène à l'affluence.

On notera également que TimeXer maintient un R^2 de 0.826 en mode autorégressif sur 5 jours, ce qui signifie qu'il explique encore 82.6% de la variance de la série test malgré l'accumulation d'erreurs jour après jour. Ce niveau de performance, combiné à une MAPE de 6.9% (erreur relative inférieure à 7%), place TimeXer dans une zone de précision compatible avec une aide à la décision opérationnelle réelle.

4.4 Comparaisons globales

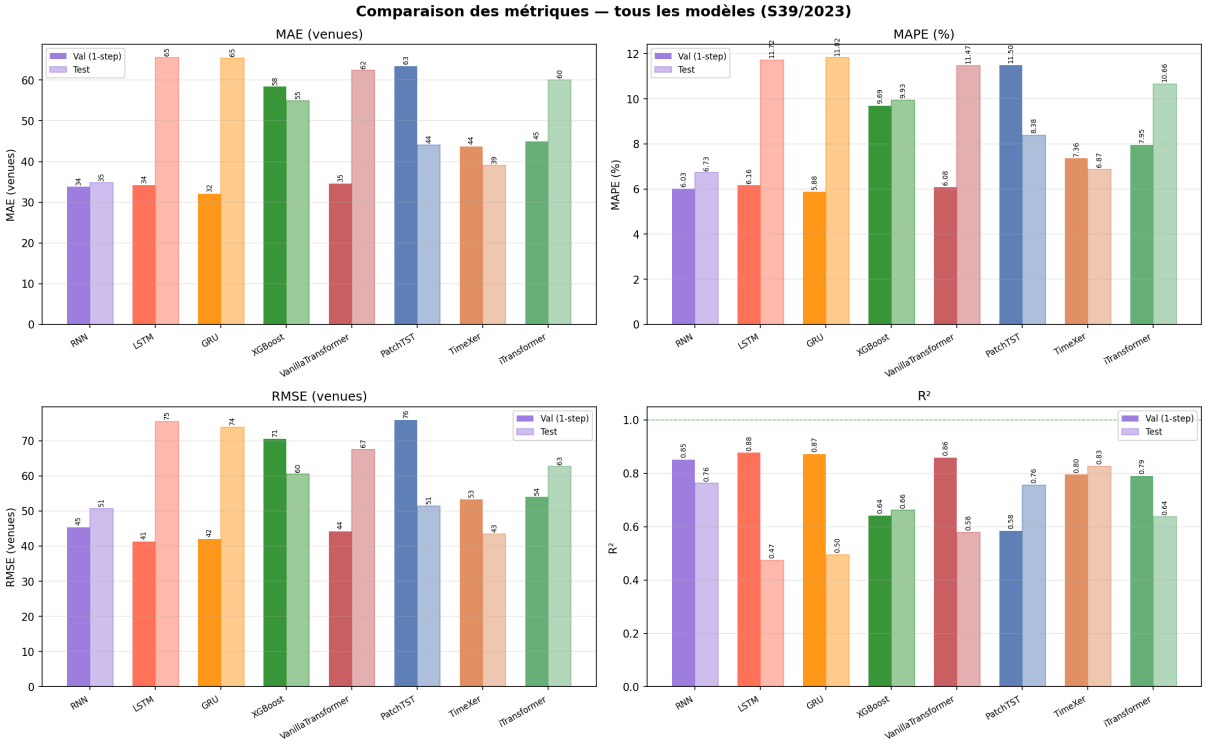


Figure 9 – Barplots MAE, MAPE, RMSE, R^2 pour tous les modèles (barres opaques = val, transparentes = test).

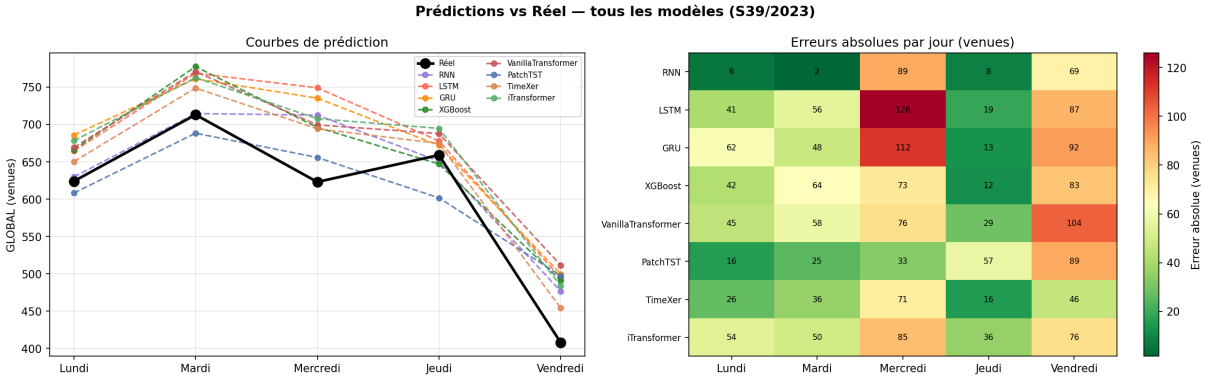


Figure 10 – Prédictions sur la semaine de test S39/2023 (gauche) et heatmap des erreurs absolues par jour et par modèle (droite).

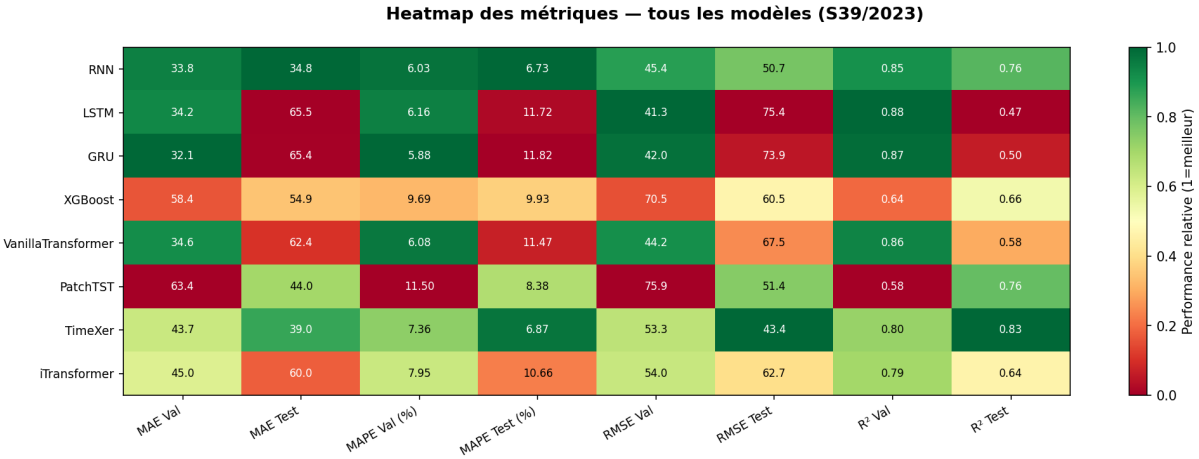


Figure 11 – Heatmap des métriques normalisées : vert = meilleur, rouge = moins bon.

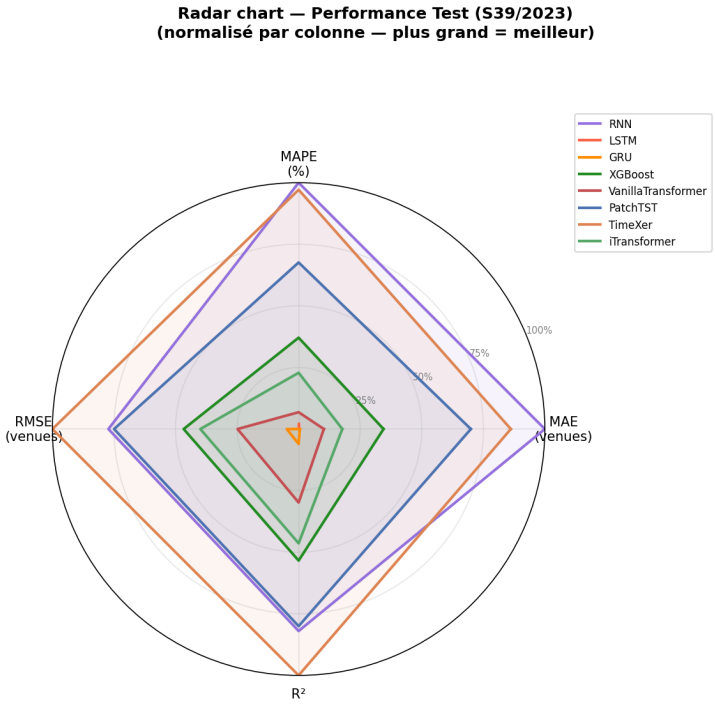


Figure 12 – Radar chart des performances test normalisées. TimeXer domine sur les quatre axes simultanément.

5 Conclusion

5.1 Meilleure approche

TimeXer est le modèle le plus performant (MAE = 39.0, MAPE = 6.9%, $R^2 = 0.826$). Sa cross-attention endo/exo capture efficacement l'influence des facteurs contextuels sur la présence, ce que les architectures récurrentes classiques peinent à faire.

RNN, malgré sa simplicité, se classe en deuxième position grâce à sa meilleure généralisation sur un dataset limité. **XGBoost** reste recommandé en production pour sa robustesse, sa rapidité d'inférence et son interprétabilité.

5.2 Limites

— **Dataset limité** : 250 jours est insuffisant pour des architectures profondes. 2–3 ans de données amélioreraient significativement les Transformers.
— **Une seule semaine de test** : S39/2023 est une semaine ordinaire. Les performances sur des semaines atypiques (grèves, vacances) restent inconnues.
— **Accumulation d'erreurs** : en mode autorégressif, les erreurs se propagent sur les 5 jours, rendant le vendredi plus difficile à prédire.
— **Météo parfaite** : les valeurs météo réelles sont utilisées, constituant un avantage artificiel par rapport à un scénario de production réel.
— **Overfitting LSTM/GRU** : malgré l'early stopping et le dropout, la val loss ne représente pas suffisamment la difficulté du test.

5.3 Perspectives

Pour améliorer ce système en production : utilisation de prévisions météo réelles, entraînement sur un historique plus long, et ré-entraînement hebdomadaire automatique au fil des nouvelles données.