3 Modèle de prédiction

3.1 Réseaux de neurones récurrents

L'objectif de ce projet est de créer un modèle de prédiction capable de générer des données de consommation électrique à partir des données des périodes précédentes. Pour atteindre cet objectif, nous allons utiliser le langage de programmation Python, avec les bibliothèques Keras et TensorFlow (Ann.6). Les données utilisées pour entraîner ce modèle comprennent des séries temporelles de courbes de charge électrique fournies par le Costic. À partir de l'état de l'art, nous avons pu conclure que les réseaux de neurones récurrents sont les algorithmes les plus adaptés pour l'apprentissage sur des séries temporelles en raison de leur capacité à mémoriser des informations à partir des longues séquences.

Les séries temporelles, particulièrement prévalentes dans le domaine de l'énergie, se caractérisent par des dépendances temporelles où les valeurs actuelles sont souvent influencées par les valeurs passées. Un aspect important de ces séries, surtout dans le contexte de la consommation énergétique, est la saisonnalité. Celle-ci se manifeste par des variations régulières sur des intervalles inférieurs à un an, comme des fluctuations hebdomadaires, mensuelles ou trimestrielles. Pour traiter efficacement ces données séquentielles, les réseaux de neurones récurrents (RNN) sont particulièrement adaptés. En examinant le schéma des RNN 3.1, Le mécanisme des RNN repose sur une interaction dynamique entre l'entrée courante $X(t)$ et l'état caché précédent $h(t-1)$. Ainsi, lorsqu'un nouvel instant t est considéré, le RNN combine l'entrée $X(t)$ avec l'état caché $h(t-1)$ pour générer un nouvel état caché $h(t)$. Cette structure confère aux RNN une forme de "mémoire" qui est cruciale pour comprendre et prédire les séries temporelles, en tenant compte des influences et des tendances passées. Cela les rend idéaux pour analyser et prédire des modèles de consommation énergétique, qui sont non seulement séquentiels mais aussi fortement influencés par des facteurs saisonniers. [4]

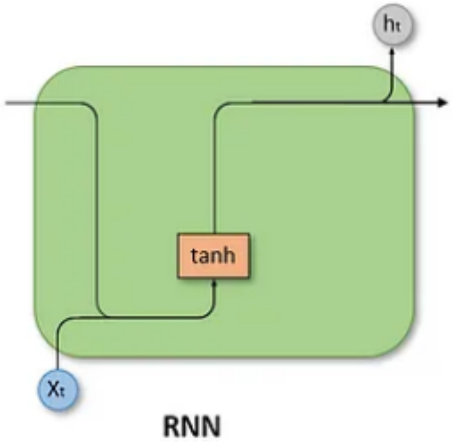


Figure 3.1 – Illustration de RNN

Cependant, les RNN classiques rencontrent des difficultés lorsqu'il s'agit de capturer des dépendances à long terme. En raison du problème de la disparition et de l'explosion du gradient(dérivée des poids), ils ont du mal à relier des informations après plusieurs pas de temps. Dans des domaines tels que la consommation énergetique, où les événements passés peuvent avoir des répercussions longtemps après, cette limitation est particulièrement problématique.

Pour surmonter les défis associés aux RNN traditionnels, les LSTM (Long Short-Term Memory) ont été développés. Ces réseaux sont une variante des RNN qui sont conçus pour capturer et conserver des dépendances à long terme, ce qui les rend particulièrement adaptés aux séries temporelles dans des domaines comme l'énergie où les tendances peuvent persister sur de longues périodes.

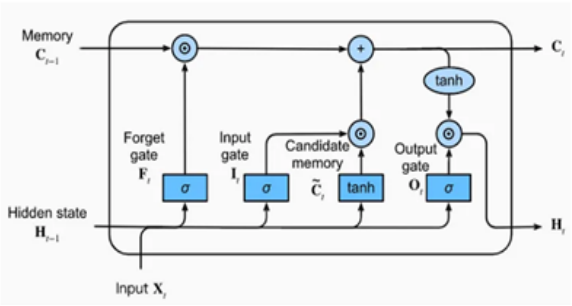


Figure 3.2 – Illustration de LSTM

Contrairement aux RNN traditionnels, les LSTM sont dotés de mécanismes (portes) qui leur permettent de décider de manière sélective quelles informations stocker, oublier ou mettre à jour. Ces mécanismes les aident à surmonter les problèmes de disparition et d'explosion du gradient. [5]

3.1.1 Porte d'oubli (Forget Gate) :

La porte d'oubli sert à éliminer de l'état de la cellule les informations qui ne sont plus utiles. Deux entrées, x(t) (entrée à un moment donné) et h(t-1) (sortie de la cellule précédente), sont alimentées à la porte. Ces entrées sont multipliées par des matrices de poids, puis une constante de biais(erreur de la cellule) est ajoutée. Le résultat est ensuite passé à travers une fonction d'activation(sigmoïde) qui donne une sortie binaire. Si pour un état de cellule particulier la sortie est 0, l'information est oubliée. Si la sortie est 1, l'information est conservée pour une utilisation future. [6]

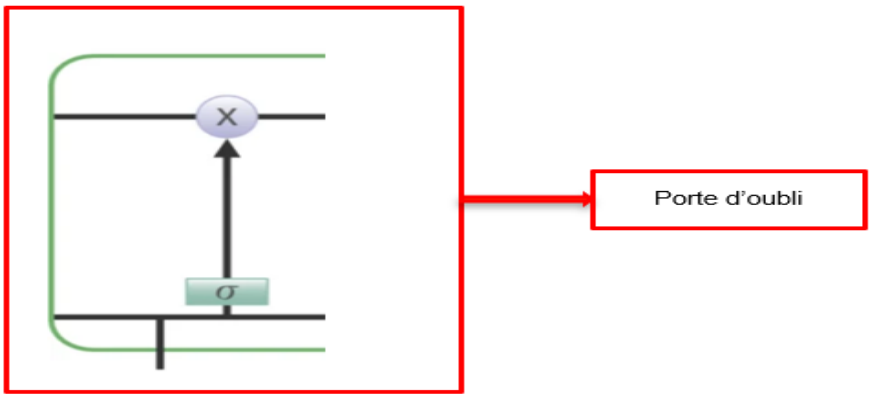


Figure 3.3 – Porte d'oubli

Afin d'accomplir cela, l'équation suivante est employée :

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f) \tag{3.1}$$

avec :
f_t : Sortie de la porte d'oubli pour le temps t.
σ : Fonction d'activation sigmoïde.
W_f : Matrice de poids associée à la porte d'oubli.
h_{t-1} : État caché (ou sortie) du LTSM au temps (t-1).
X_t : Entrée au temps t.
b_f : Biais associé à la porte d'oubli.

3.1.2 Porte d'entrée (Input gate) :

La porte d'entrée ajoute des informations pertinentes à l'état de la cellule. Ce processus se déroule en plusieurs étapes. Premièrement, un vecteur d'informations potentiellement utiles est généré via une fonction d'activation tanh, qui produit une sortie dans l'intervalle de -1 à +1. Parallèlement, un autre vecteur est créé en utilisant la fonction sigmoïde, qui régule les informations en déterminant leur degré d'importance (entre 0 et 1). Ensuite, une opération de multiplication élément par élément est effectuée entre ces deux vecteurs : le vecteur d'informations potentielles et le vecteur régulateur. Ce processus permet de filtrer et de conserver uniquement les informations jugées utiles. Enfin, ce vecteur résultant est ajouté à l'état de la cellule, qui a déjà intégré les informations retenues par la porte d'oubli.[6]

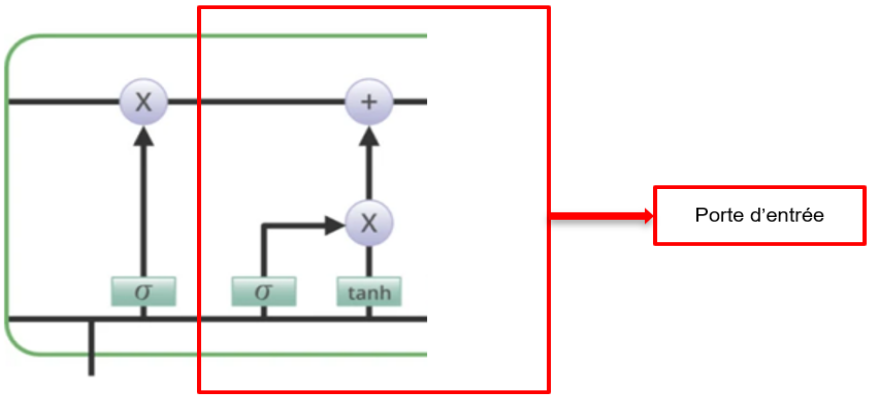


Figure 3.4 – Porte d'entrée

Pour réaliser les opérations mentionnées précédemment, le réseau se sert des équations suivantes :

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i) C'_t = \tanh(W_c \cdot [h_{t-1}, X_t] + b_c) C_t = f_t \cdot C_{t-1} + i_t \cdot C'_t \tag{3.2}$$

avec :
i_t : Fonction d'activation sigmoïde.
W_i : Matrice de poids associée à la porte d'entrée.
h_{t-1} : État caché (ou sortie) du LTSM au temps (t-1).
X_t : Entrée au temps t.
b_i : Biais associé à la porte d'entrée.
C'_t : Valeur candidate pour le nouvel état de la cellule au temps t.
W_c : Matrice de poids pour la valeur candidate de l'état de la cellule.
b_c : Biais pour la valeur candidate de l'état de la cellule.
C_t : État de la cellule au temps t.

3.1.3 Porte de sortie (Output gate) :

La porte de sortie extrait l'information utile de l'état actuel de la cellule pour la présenter comme sortie. Un vecteur est d'abord généré en appliquant la fonction tanh à la cellule. Ensuite, l'information est régulée à l'aide de la fonction sigmoïde. Enfin, les valeurs du vecteur et les valeurs régulées sont multipliées pour être envoyées en sortie et en entrée à la cellule suivante.[6]

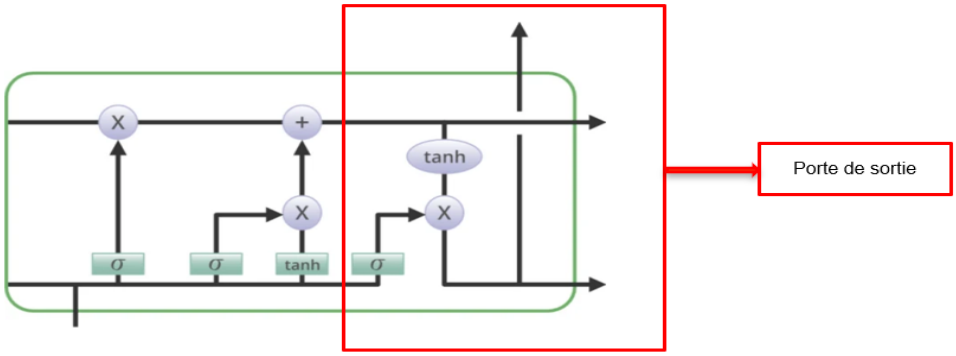


Figure 3.5 – porte de sortie

Pour ce faire, cette partie de la LSTM, appelée ”porte de sortie”, utilise l'équation présentée ci-dessous :

$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o) \tag{3.3}$$

avec :
o_t : Sortie de la porte de sortie pour le temps t.
σ : Fonction d'activation sigmoïde.
W_o : Matrice de poids associée à la porte de sortie.
h_{t-1} : État caché (ou sortie) du LTSM au temps (t-1).
X_t : Entrée au temps t.
b_o : Biais associé à la porte de sortie.

3.2 Convergence du modèle

La convergence d'un modèle d'apprentissage profond est un processus crucial qui détermine la capacité du modèle à fournir des prédictions précises. Elle se base sur plusieurs étapes interdépendantes et des concepts essentiels.

3.2.1 La Fonction de Perte

La première étape dans l'évaluation de la convergence d'un modèle en apprentissage automatique est la mesure de sa performance. Cette évaluation est réalisée à l'aide de la fonction de perte, qui quantifie l'écart entre les prédictions du modèle et les valeurs réelles attendues.

Un exemple courant est la Mean Squared Error (MSE), souvent utilisée dans les tâches de régression. La MSE est définie comme la moyenne des carrés des différences entre les prédictions du modèle et les véritables valeurs. Elle est donnée par la formule :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.4}$$

où y_i est la valeur réelle et $\hat{y}_i$ est la valeur prédite par le modèle, pour chaque échantillon i dans un ensemble de n échantillons. Une MSE faible indique que les prédictions du modèle sont proches des valeurs réelles. [7]

3.2.2 Rétropropagation : Corriger les erreurs

Une fois l'erreur évaluée, l'étape suivante consiste à ajuster le modèle pour minimiser cette erreur. La rétropropagation est une méthode essentielle dans l'apprentissage des réseaux de neurones. Elle permet de mettre à jour les poids du modèle en fonction de l'erreur mesurée par la fonction de perte. Cette méthode calcule le gradient de l'erreur par rapport à chaque poids en utilisant la règle de la chaîne, puis ajuste les poids dans la direction qui minimise l'erreur.

La mise à jour du poids w se fait selon la formule :

$$w_{new} = w_{old} - \eta \frac{\partial E}{\partial w} \tag{3.5}$$

où E est l'erreur calculée par la fonction de perte, $\frac{\partial E}{\partial w}$ est le gradient de l'erreur par rapport au poids w, et η est le taux d'apprentissage. [8]

3.2.3 Optimiseurs : La mise à jour efficace des poids

Après que la rétropropagation a calculé les gradients, l'optimiseur intervient pour mettre à jour les poids du réseau. Il détermine la manière et l'ampleur de l'ajustement des poids, avec pour objectif de minimiser l'erreur, évaluée par la fonction de perte.

Dans ce travail, nous utilisons l'optimiseur ADAM pour minimiser l'erreur. Cet algorithme se distingue par son suivi efficace des gradients pour optimiser les paramètres du modèle. Voici le processus détaillé :

— **Initialisation** : ADAM initialise les estimations des gradients du premier ordre (moyennes mobiles des gradients) et du second ordre (moyennes mobiles des carrés des gradients) à zéro. Mathématiquement, cela s'exprime par $m_0 = 0$ et $v_0 = 0$.
— **Mise à jour des estimations** : À chaque étape t, ADAM calcule le gradient actuel g_t, puis met à jour ses estimations m_t et v_t en utilisant les équations :

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{3.6}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{3.7}$$

où β_1 et β_2 sont des hyperparamètres de décomposition.
— **Correction de biais** : Les estimations initiales peuvent être biaisées vers zéro, en particulier lorsque β_1 et β_2 sont proches de 1. ADAM corrige ce biais en ajustant les estimations avec les équations :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.9}$$

— **Mise à jour des paramètres** : Les paramètres du modèle sont ajustés en utilisant les estimations corrigées :

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{3.10}$$

où α est le taux d'apprentissage et ϵ est un petit nombre pour éviter la division par zéro.
— **Répétition** : Ces étapes sont répétées jusqu'à convergence du modèle ou atteinte d'un nombre prédéfini d'itérations. [9]

3.2.4 Processus de convergence

À chaque itération, le modèle utilise d'abord la fonction de perte pour évaluer l'écart entre ses prédictions et les valeurs réelles. Ensuite, la technique de rétropropagation est employée pour calculer le gradient de l'erreur par rapport à chaque poids du réseau. Cette étape est cruciale car elle

détermine comment et dans quelle mesure les poids doivent être ajustés pour réduire l'erreur. Enfin, l'optimiseur, tel que ADAM dans notre cas, est utilisé pour effectuer ces ajustements de façon efficace. ADAM adapte le taux d'apprentissage pour chaque poids en fonction des informations précédemment calculées, permettant ainsi des mises à jour plus précises et adaptatives.

Au fil des itérations, et sous réserve d'une conception adéquate du modèle et d'une sélection appropriée des données, la fonction de perte devrait progressivement diminuer. Cette diminution indique que le modèle converge vers une solution plus optimale. Ce cycle d'évaluation, d'ajustement et de mise à jour se poursuit jusqu'à ce que le modèle atteigne un niveau de précision considéré comme satisfaisant, marquant ainsi l'achèvement du processus de convergence.[9]

3.3 Développement et utilisation d'un programme basé sur les RNN

Notre modèle, centré sur l'intelligence artificielle et exploitant les réseaux de neurones, notamment les réseaux de neurones récurrents (RNN), vise à prédire la consommation électrique d'un bâtiment en se basant sur ses données de consommation passées. Pour construire efficacement ce modèle, nous avons opté pour l'utilisation du langage de programmation Python. Dans le but de faciliter son implémentation, nous nous appuierons sur les bibliothèques Keras et TensorFlow. Les données nécessaires à l'entraînement du modèle sont mises à notre disposition par le COSTIC, sous la forme de courbes de charge électrique pour plusieurs bâtiments. Ces données sont directement extraites des enregistrements d'ENEDIS, assurant ainsi une base fiable et pertinente pour l'apprentissage du modèle. Dans les sections à venir, nous exposerons les données dont nous disposons et détaillerons comment nous avons exploité ces données dans le développement de notre modèle.

3.3.1 Exploration des données

Nous disposons de deux ensembles de données qui enregistrent la consommation électrique :

Le premier ensemble concerne un bureau pour la période du 31 décembre 2021 au 30 décembre 2022 et le second ensemble concerne un hôpital pour la période du 1er janvier 2022 au 29 Avril 2023 , La consommation est enregistrée en watts toutes les 10 minutes pour les deux établissements.

3.3.1.1 Données de l'hôpital

Les données de l'hôpital comprennent deux colonnes : une colonne "Horodaté" pour le temps et une colonne "Valeur" qui indique la consommation énergétique en watt-heure. Nous présenterons dans le tableau ci-dessus un résumé des informations disponibles sur la consommation électrique de l'hôpital, avec un intervalle de temps égal à 10 minutes.

Catégorie	**Valeur**
Date de début	2022-01-01 00 :00 :00 UTC
Date de fin	2023-04-29 21 :50 :00 UTC
consommation électrique Min (Wh par intervalle de temps)	96000
consommation électrique Max (Wh par intervalle de temps)	890000
consommation électrique Moyenne (Wh par intervalle de temps)	460594

Table 3.1 – Résumé des informations sur la consommation électrique de l'hôpital.

Suite à l'analyse des données de consommation électrique de l'hôpital, il a été identifié une absence complète d'enregistrements entre le 29 avril 2022 et le 13 mai 2022 (3.6). Cette période de 15 jours suggère soit un arrêt de l'hôpital, soit un problème dans la collecte des données.

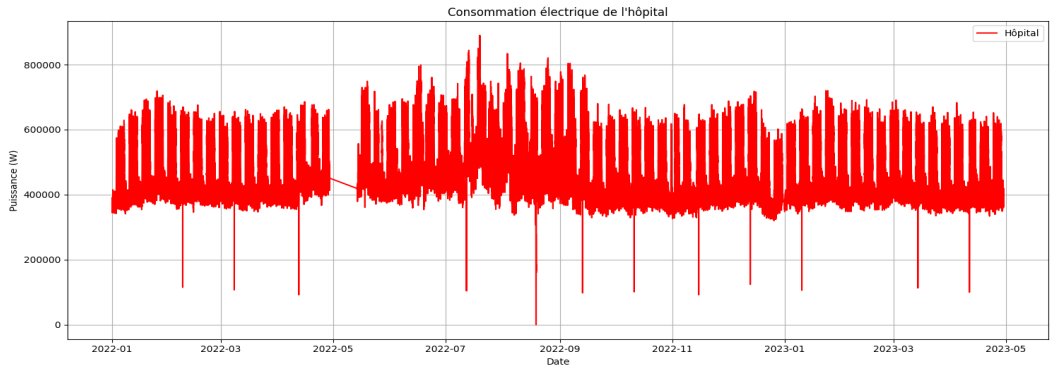


Figure 3.6 – La courbe électrique de l'hôpital

3.3.1.2 Données du bureau

Dans les informations relatives au bureau, on trouve deux colonnes distinctes : la première, intitulée "Horodaté", dédiée à la gestion temporelle, tandis que la seconde, sous le nom de "Valeur", enregistre la consommation énergétique exprimée en watt-heure pour chaque pas de temps.

Il est important de noter que le nombre de lignes dans les données du bureau est significatif en raison de l'enregistrement fréquent toutes les 5 minutes pendant certaines périodes. Cependant, dans le cadre de notre analyse, nous ne prendrons en compte que les intervalles de 10 minutes. Cette démarche sera abordée plus en détail lors de la phase de nettoyage des données.

Le tableau présent synthétise les données relatives à la consommation électrique du bureau.

Catégorie	**Valeur**
Date de début	2021-12-31 23 :20 :00 UTC
Date de fin	2022-12-30 22 :50 :00 UTC
consommation électrique Min (Wh par intervalle de temps)	14000
consommation électrique Max (Wh par intervalle de temps)	103000
consommation électrique Moyenne (Wh par intervalle de temps)	27171.35

Table 3.2 – Résumé des informations sur la consommation électrique de l'hôpital.

La courbe ci-dessous(3.7) illustre la consommation électrique du bureau tout au long de l'année. Les variations dans la consommation du bureau sont plus marquées, probablement en raison des heures d'ouverture et de fermeture habituelles, ainsi que des week-ends et des jours fériés.

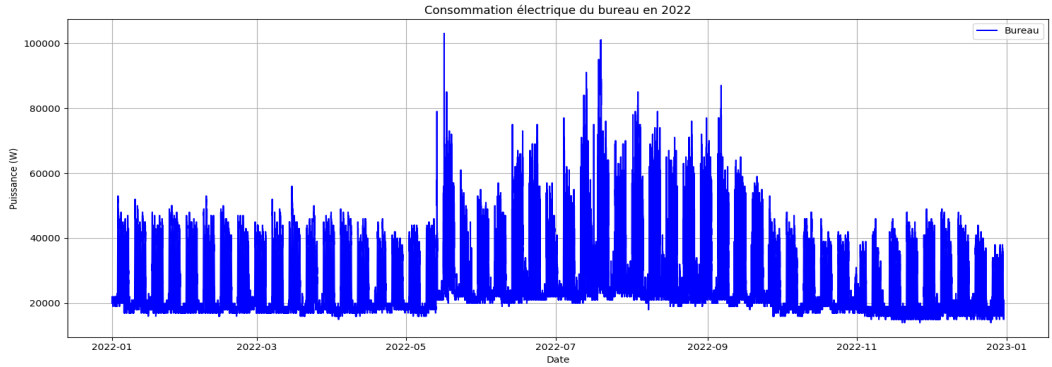


Figure 3.7 – La courbe électrique du bureau

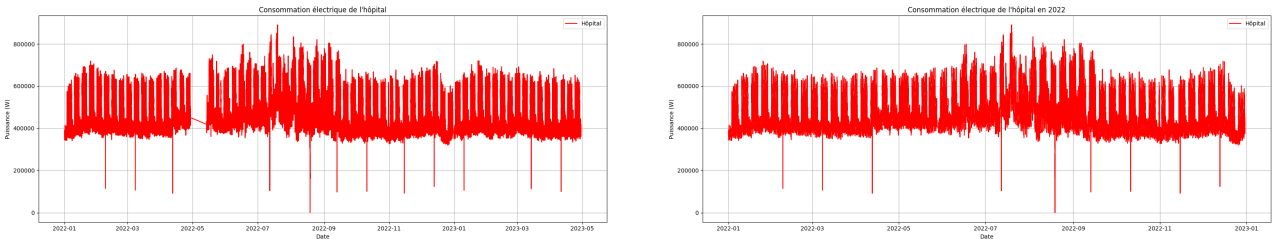
3.3.2 Traitement des données

3.3.2.1 Nettoyage des données - Data cleaning

Données de l'hôpital

Dans notre analyse dans la partie précédente, nous avons identifié des périodes où les données étaient manquantes dans les données de l'hôpital. Pour remédier à cette absence d'informations et maintenir l'intégrité de la série temporelle, nous avons adopté une approche basée sur les données historiques. Cette méthode consistait à combler les périodes manquantes en prenant en compte les données des périodes équivalentes avant et après l'intervalle de temps absent.

Cette stratégie a été choisie pour sa capacité à refléter les tendances saisonnières et les comportements habituels observés dans les données. Elle permet de préserver la structure temporelle et les caractéristiques intrinsèques de la série, garantissant ainsi que le modèle de prévision soit alimenté par des séquences complètes et représentatives des cycles réels.



(a) Avant le nettoyage des données (b) Aprés le nettoyage des données

Figure 3.8 – Nettoyage des données

Données du Bureau

Comme mentionné précédemment dans la section 3.3.1.2, les données enregistrées pour le bureau étaient collectées à des intervalles de 5 minutes pour certaines périodes. Afin de normaliser ces données, un traitement a été appliqué. Ce traitement consistait à éliminer les observations prises à des intervalles de 5 minutes et à ne conserver que celles enregistrées à des pas de temps de 10 minutes (reste de division par 10). Cette démarche vise à uniformiser les données pour une analyse plus cohérente. Le nombre de colonnes est passé de 73 297 à 52 416. Il convient de noter que ce traitement a été réalisé sur Python.

3.3.3 Mise en forme des données

3.3.3.1 Mise à l'echelle des données - Data scaling

Le scaling, ou mise à l'échelle, est une pratique standard dans le traitement des séries temporelles pour les modèles d'apprentissage automatique, tels que les LSTM. Cette pratique consiste à transformer les données afin qu'elles soient sur une échelle uniforme, généralement entre 0 et 1. Une telle normalisation est essentielle car elle permet aux algorithmes d'apprentissage profond de converger de manière plus rapide et stable, en évitant les déséquilibres dans les plages de valeurs des différentes caractéristiques (Ann.7).

Pour effectuer ce scaling, le MinMax Scaler a été utilisé. Ce scaler ajuste les données selon la formule :

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.11}$$

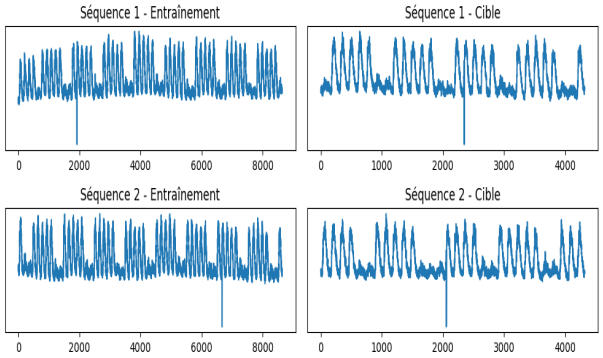
où x représente la valeur originale, x_{min} et x_{max} sont respectivement les valeurs minimale et maximale de la série de données. Le résultat x_{scaled} est la valeur mise à l'échelle, située dans la nouvelle plage entre 0 et 1. [10]

3.3.3.2 Répartition des données

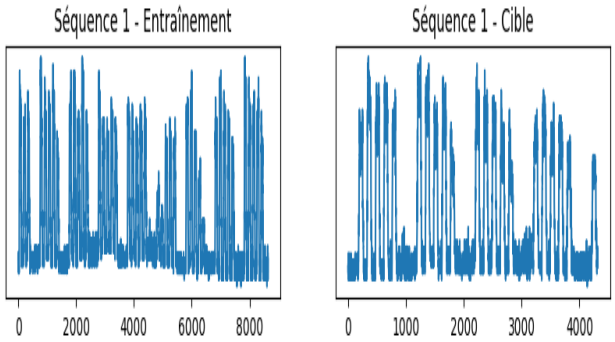
L'objectif principal de cette étude est de développer un modèle capable de prédire la consommation énergétique pour le mois à venir, en se basant sur une séquence de données de deux mois avec des relevés effectués toutes les 10 minutes. Pour ce faire, il est nécessaire de préparer les données de manière adéquate afin de les rendre exploitables par le modèle. Dans cette optique, nous structurons les données en séquences de trois mois : deux mois serviront de données d'entraînement et le mois suivant comme cible de prédiction. Étant donné que nous disposons de 16 mois de données provenant de l'hôpital, nous pouvons générer 14 séquences distinctes. Parmi celles-ci, 12 seront utilisées pour l'entraînement du modèle, et les deux séquences restantes serviront à tester sa fiabilité. Parallèlement, nous avons également 12 mois de données issues du bureau, permettant la création de 10 séquences, dont 9 pour l'entraînement et une pour les tests.

En raison de la faible quantité de données à notre disposition, et parce que les modèles d'apprentissage profond nécessitent une grande quantité de données, et surtout qu'il nous fallait des données pour l'entrainement et d'autres pour le test, on a opté pour cette répartition de séquence de 3 mois afin de générer un nombre acceptable de séquences et garantir une évaluation fiable de la performance du modèle. . On note que ce modèle et avec de légères modifications peut être utilisé sur des séquences plus longues.

Séquences des données utilisées pour la validation du modéle



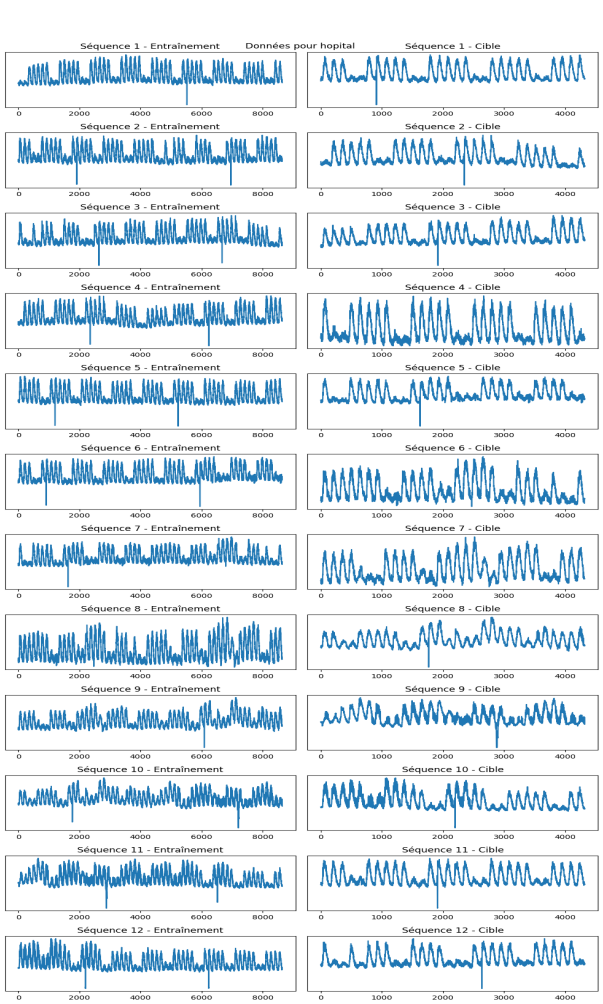
(a) séquences des données de l'hôpital utilisées pour la validation



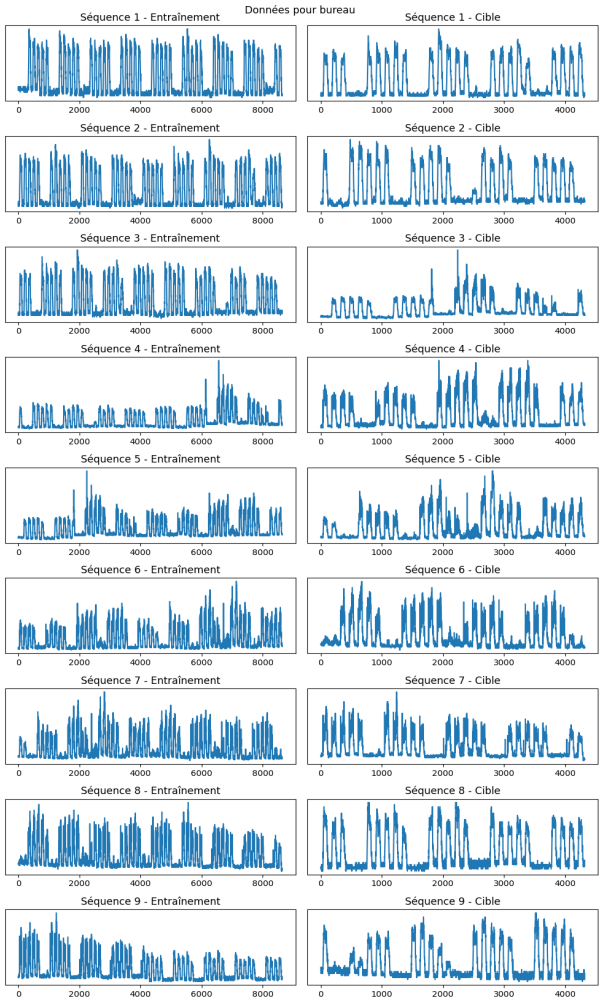
(b) séquences des données du bureau utilisées pour la validation

Figure 3.9 – séquences des données utilisées pour la validation du modéle

Séquences des données utilisées pour l'entraînement



(a) séquences des données de l'hôpital utilisées pour l'entrainement



(b) séquences des données du bureau utilisées pour l'entrainement

Figure 3.10 – séquences des données utilisées pour l'entrainement du modéle

Cette méthodologie vise à garantir que le modèle est non seulement capable de prédire précisément la consommation énergétique sur la base des données d'entraînement, mais également qu'il soit fiable et performant lorsqu'il est confronté à de nouvelles données, reflétant ainsi sa capacité à généraliser à partir de tendances variées.

3.4 Description du Modèle

Le modèle LSTM conçu est un réseau de neurones pour le traitement de séquences temporelles. Le modèle est construit à l'aide de la bibliothèque Keras et se compose des éléments suivants :

— **Couche LSTM (512 unités) :** Cette première couche LSTM traite la séquence d'entrée avec 512 unités. En d'autres termes, il y a 512 "cellules" LSTM fonctionnant simultanément dans cette couche. Chacune de ces unités effectue des opérations pour apprendre des motifs dans les séquences temporelles. Elle est configurée pour retourner la séquence complète, facilitant ainsi la connexion avec les couches suivantes.
— **Dropout (0.2) :** Une couche de dropout avec un taux de 20% est utilisée pour réduire le surajustement. Elle fonctionne en désactivant aléatoirement un pourcentage des connexions neuronales pendant l'entraînement, ce qui aide à généraliser le modèle.
— **Couche LSTM (256 unités) :** La deuxième couche LSTM, avec 256 unités, reçoit la séquence de la première couche LSTM et retourne uniquement la dernière sortie de la séquence.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 8640, 512)         1052672

 dropout (Dropout)           (None, 8640, 512)         0

 lstm_1 (LSTM)               (None, 256)               787456

 dropout_1 (Dropout)         (None, 256)               0

 dense (Dense)               (None, 4320)              1110240

=================================================================
Total params: 2,950,368
Trainable params: 2,950,368
Non-trainable params: 0
```

Figure 3.11 – Les composants du modéle

— **Dropout (0.1) :** Suit la même logique que la première couche de dropout mais avec un taux réduit à 10%, appliqué après la deuxième couche LSTM.
— **Couche Dense :** La couche dense avec une activation linéaire est utilisée pour la sortie du modèle. Le nombre d'unités correspond à la longueur de la séquence de sortie souhaitée. Cette couche consolide les informations apprises et les transforme en prédictions finales de la séquence.

Le modèle est compilé avec une fonction de perte 'mean squared error' (MSE) et utilise l'optimiseur Adam pour l'ajustement des poids.

3.5 Entraînement du Modèle

Pour optimiser l'efficacité de l'entraînement de notre modèle , nous avons intégré des techniques, garantissant des résultats précis et fiables :

— **EarlyStopping** : Cette méthode est essentielle pour éviter le surajustement, où le modèle s'adapte trop précisément aux données d'entraînement au détriment de sa capacité à généraliser. L'entraînement s'arrête lorsque la perte ('loss') ne diminue plus, ce qui signifie que le modèle cesse de s'améliorer. Le paramètre 'patience', fixé à 300 époques, détermine le nombre d'itérations sans amélioration avant l'arrêt. Dans ce contexte, la notion"époque" fait référence à une itération complète à travers l'ensemble des données d'entraînement pendant le processus d'apprentissage du modèle. [11]
— **CustomCallback** : Cette fonctionnalité enregistre automatiquement le meilleur modèle en fonction de la perte minimale à chaque fin d'époque, et sauvegarde le modèle final. Ce système de rappel (callback) est une procédure ou fonction qui est appelée à un certain point du processus d'entraînement pour effectuer des actions spécifiques, telles que la sauvegarde de modèles ou le suivi des métriques. [12]
— **Entraînement du Modèle** : Notre modèle a été entraîné sur 2000 époques, en utilisant un taux d'apprentissage de 0.001 pour un apprentissage efficace. L'élément clé de notre méthode d'entraînement est l'utilisation de l'entraînement par lot (batch). Cette technique divise les données en petits groupes pour un traitement parallèle efficace, optimisant ainsi l'utilisation des ressources de calcul.
Cette approche est particulièrement avantageuse sur notre machine équipée d'un GPU Nvidia RTX 3050 Ti. Ce GPU, adapté à l'entraînement par batch, permet une accélération notable du processus grâce à sa capacité de calcul parallèle. De plus, l'entraînement par batch facilite une

meilleure gestion de la mémoire, chaque batch étant chargé et traité séparément, ce qui réduit la charge sur la mémoire et accélère l'entraînement

3.5.1 Entraînement du Modèle sur les données de l'hôpital

Comme mentionné précédemment dans la section 3.3.3.2, notre modèle a été entraîné sur un ensemble de 12 séquences de données de la consommation énergétique d'un hôpital. Ce modèle a été entraîné avec une taille de lot (batch size) de 12, ce qui signifie que dans chaque itération, le modèle apprend sur la totalité des séquences disponibles. Comme on peut le voir sur la figure 3.13, l'entraînement s'est arrêté à l'époque 1890 grâce à la fonctionnalité de l'EarlyStopping mentionnée précédemment.

Concernant la performance en termes de temps, la durée totale de l'entraînement s'est élevée à environ 3 heures. Il est important de souligner que cette efficacité est en partie due à l'utilisation d'une machine dotée d'une unité de traitement graphique (GPU) avec une capacité de 4 Go. L'emploi d'un tel équipement a permis d'accélérer considérablement le processus d'entraînement, démontrant l'importance d'une infrastructure matérielle adaptée dans le déploiement efficace des modèles d'apprentissage profond.

La figure 3.13 illustre l'évolution de la perte au fil des époques [3.5]. On observe que le modèle converge rapidement, atteignant un état stable après environ une trentaine d'époques. Cette convergence rapide est suivie de légères fluctuations, témoignant d'un ajustement fin du modèle aux nuances des données. Cette rapidité de convergence peut s'expliquer par la quantité relativement limitée de données disponibles pour l'entraînement. Dans les cas où l'ensemble de données est restreint, le modèle a tendance à apprendre et à s'adapter plus rapidement, car il a moins de variations et de complexités à intégrer.

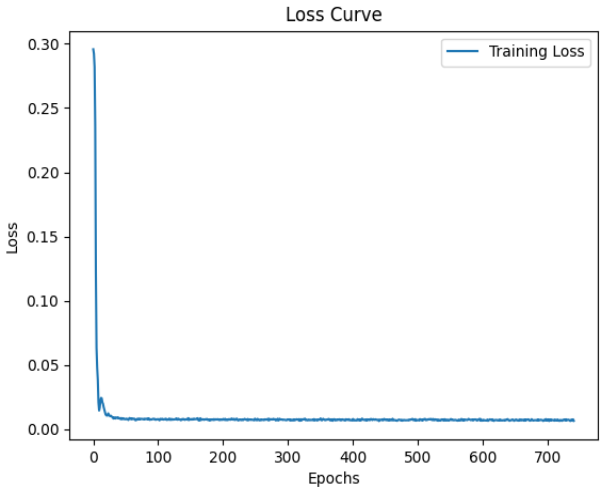


Figure 3.12 – L'evolution de la perte au fil des époques du modéle de l'hopital

3.5.2 Entraînement du Modèle sur les données du bureau

Pour le jeu de données du bureau, le modèle a été entraîné sur 9 séquences, avec un lot de taille 9, avec une convergence plus lente, comparable à celle constatée pour les données de l'hôpital. Néanmoins, la durée totale de l'entraînement a été considérablement augmentée, s'établissant à environ 5 heures et 15 minutes. Cette augmentation est attribuable au fait que le modèle a été entraîné sur plus de 1800 époques, car la perte a continué de diminuer jusqu'à l'époque 1500.

Pour ce modèle appliqué aux données du bureau, la courbe de perte montre une stabilisation lente accompagnée de légères fluctuations. Une observation notable est que la perte minimale obtenue est largement inférieure à celle enregistrée pour le modèle entraîné sur les données hospitalières, ce qui suggère une convergence efficace. Cette différence peut s'expliquer de plusieurs manières : il est

possible que les données de l'hôpital présentent une complexité supérieure, rendant l'apprentissage des tendances plus ardu pour le modèle, surtout que dans la partie du nettoyage des données, on a procédé au remplissage des données manquantes en utilisant des données historiques, ce qui peut engendrer un biais influençant les performances des modèles. De plus, cela peut être attribuable à l'application de l'EarlyStopping. Si le critère d'EarlyStopping (patience de 300 époques [3.5]) a été défini de manière trop conservatrice, le modèle entraîné sur les données de l'hôpital aurait pu bénéficier de plus d'époques pour affiner son apprentissage et réduire davantage la perte. Cela souligne l'importance du choix des paramètres utilisés pour le processus d'entraînement.

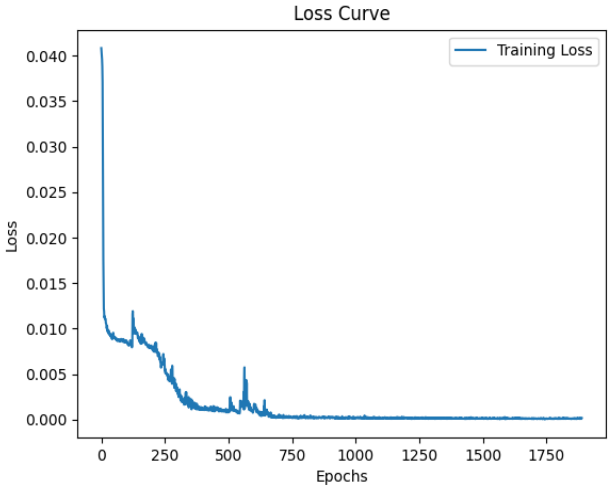


Figure 3.13 – La variation de la perte à travers les époques du modéle du bureau.

3.5.3 Entraînement du Modèle sur la totalité des ensembles

Dans ce troisième scénario, le modèle a été entraîné sur l'intégralité des séquences des deux ensembles de données : hôpital et bureau. Cette approche visait à évaluer la capacité du modèle à apprendre à partir de données présentant des tendances et des niveaux de consommation variés.

Au total, 21 séquences ont été utilisées pour l'entraînement, avec un lot de taille 21, comme expliqué précédemment, ce qui signifie que le modèle apprend sur la totalité des 21 séquences simultanément dans chaque époque [3.5]. Dans ce contexte, la convergence a été plus lente par rapport aux deux cas précédents, un résultat attendu non seulement en raison du nombre élevé de séquences, mais aussi de la complexité et de l'hétérogénéité des données. Concernant le temps d'exécution, il a doublé par rapport à l'entraînement sur les données hospitalières, s'élevant à 6 heures pour un entraînement sur un peu moins de 1000 époques. Cette augmentation est logique compte tenu du nombre plus élevé de séquences intégrées à chaque époque [3.5].

Observant l'évolution de la perte lors de ce troisième scénario d'entraînement (Fig. 3.14), on note que le modèle présente une trajectoire d'apprentissage distincte par rapport aux scénarios précédents. Il manifeste d'abord une convergence vers un état stable qui se prolonge jusqu'à l'époque 300, indiquant une adaptation plus lente due à la diversité des tendances des données combinées. La présence de deux paliers de stabilité suggère que le modèle a expérimenté des difficultés à trouver une solution optimale qui englobe efficacement les variations inhérentes aux ensembles de données hétérogènes.

Cependant, plusieurs pics marqués de perte sont observés au cours des époques [3.5] même après la stabilité. Ces variations soudaines dans la perte peuvent être attribuées à l'utilisation du dropout pendant l'entraînement. Le dropout fonctionne en désactivant aléatoirement un pourcentage de neurones à chaque itération, ce qui aide à prévenir le surajustement en forçant le réseau à ne pas compter excessivement sur un neurone en particulier. Cependant, cette désactivation aléatoire peut parfois entraîner une perte d'information importante pour la prédiction durant certaines itérations, ce qui se traduit par des pics dans la courbe de perte. Ces pics reflètent des itérations spécifiques où

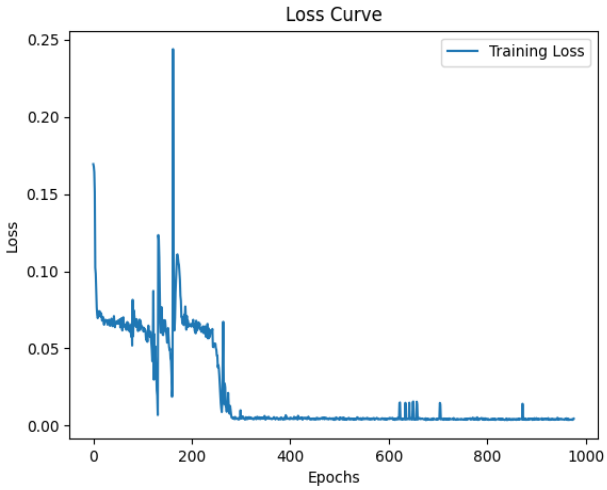


Figure 3.14 – L'évolution de la perte au cours les époques du modéle compiné

le masquage des neurones a probablement empêché le réseau de faire des prédictions précises, révélant ainsi l'impact fluctuant du dropout sur le processus d'apprentissage.

Malgré ces irrégularités, la perte globale diminue au cours des époques, ce qui indique une amélioration continue des capacités prédictives du modèle. Cela démontre la capacité d'adaptation du modèle, qui parvient progressivement à intégrer et à généraliser à partir d'une variété de tendances et de comportements de consommation présents dans les ensembles de données.

3.6 Validation du modèle

Après avoir observé une convergence satisfaisante des modèles sur les données d'entraînement, il est impératif de reconnaître que cette réussite ne garantit pas une performance globale optimale. La validation de ces modèles sur des données nouvelles et jamais vues est essentielle pour évaluer leur capacité à généraliser.

Chacun des trois modèles sera testé sur les séquences de test de l'hôpital et du bureau, cette démarche vise à déterminer avec précision l'efficacité et la fiabilité de chaque modèle lorsqu'ils sont confrontés à des données qu'ils n'ont pas encore rencontrées, ainsi leur évaluation sur des données qui n'ont pas les mêmes tendances que les données d'entrainement.

Pour évaluer la performance des modèles, nous utiliserons l'erreur moyenne absolue (MAE) et le coefficient de variation de l'erreur moyenne absolue (CV(MAE)). Le MAE est une mesure standard de précision qui quantifie l'écart moyen entre nos prédictions et les valeurs réelles définie par :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|prediction_i - target_i| \tag{3.12}$$

où $prediction_i$ est la valeur prédite par le modèle pour l'élément i, et $target_i$ est la valeur réelle pour cet élément.

Le CV(MAE) nous permettra de mettre cette erreur en perspective par rapport à la variabilité des données observées, ce qui est particulièrement utile pour comparer la performance du modèle sur des séries de données avec différentes échelles, ce coefficient est définie par :

$$CV(MAE) = \left(\frac{MAE}{\bar{target}}\right) \times 100\% \tag{3.13}$$

où $\bar{target}$ est la moyenne des valeurs observées dans la séquence cible.

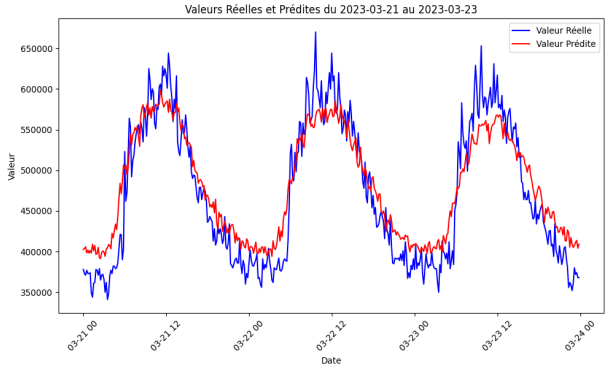
3.6.1 Modèle entrainé sur les données de l'hopital

Le modèle, après avoir été entraîné sur les données de l'hôpital, présente un coefficient de variation de l'erreur moyenne absolue (CV(MAE)) compris entre 9% et 12.5% lorsqu'il est testé sur de nouvelles séquences de l'hôpital (voir Figure 3.15). Cela indique une généralisation réussie à des données inconnues jusqu'alors. Cependant, l'erreur augmente significativement lorsque ce modèle est testé sur les données du bureau, ce qui suggère une capacité de généralisation moins efficace à des données présentant des tendances et des niveaux de consommation différents de ceux sur lesquels il a été formé.

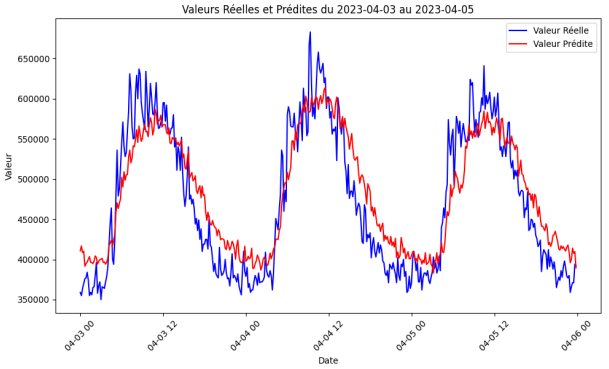
	Modèle	Séquence	MAE (Wh)	Moyenne des données réelles (Wh)	CV_MAE (%)
0	Modèle_hôpital	sequence_test_bureau.csv	458851.943171	24135.879630	1901.119620
1	Modèle_hôpital	sequence_test_hopital_1.csv	45125.012652	456614.583333	9.882517
2	Modèle_hôpital	sequence_test_hopital_2.csv	54842.378320	439253.240741	12.485367

Figure 3.15 – L'érreur moyenne absolue sur le modéle de l'hopital

La Figure 3.16 présente les résultats détaillés du test du modèle sur deux extraits de 3 jours des séquences de test des données de l'hôpital sélectionnées aléatoirement. Ces résultats montrent que le modèle a bien appris sur les données de l'hôpital, parvenant à suivre les tendances même sur des données qu'il n'a jamais vues auparavant. Cependant, il semble avoir plus de difficultés à suivre les pics dans la courbe de charge électrique, ce qui suggère une meilleure compréhension des tendances globales par rapport aux variations locales.



(a) Application sur la période 21/03/2023 au 23/03/2023



(b) Application sur la période 03/04/2023 au 05/04/2023

Figure 3.16 – Comparaison des résultats prédictifs et des valeurs observées du modèle d'hopital sur une séquence hospitalière

3.6.2 Modèle entrainé sur les données du bureau

Lorsqu'il est entraîné avec des données provenant du bureau, le modèle présente un coefficient de variation de l'erreur moyenne absolue (CV(MAE)) atteignant 22% lorsqu'il est testé avec une nouvelle séquence du bureau qu'il n'a jamais vue auparavant. Cette augmentation de l'erreur par rapport aux données de test précédentes s'explique par le fait que le modèle n'a jamais été exposé à des données de la même période de l'année, conduisant à une lacune dans la compréhension des tendances spécifiques du bureau pendant cette période (décembre 2022). Cependant, sur une séquence déjà vue lors de l'entraînement (séquence 9), l'erreur diminue à environ 1%, ce qui soulève la question de l'absence de données de test valides correspondant à une période déjà observée par le modèle, ce qui rend difficile la conclusion sur l'éventuel surajustement du modèle.

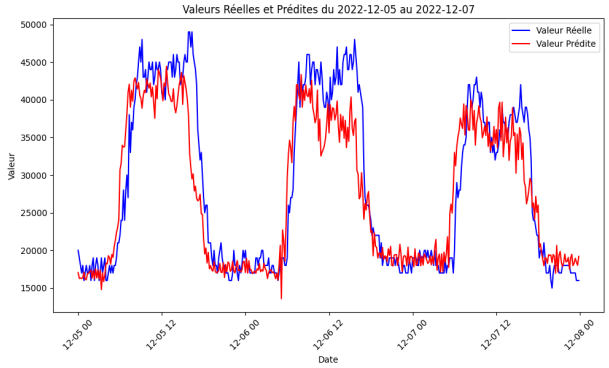
Les résultats du modèle sur les données de l'hôpital montrent une augmentation significative de l'erreur, suggérant une capacité moindre du modèle à généraliser ses connaissances à des données

présentant des tendances et des niveaux de consommation différents de ceux sur lesquels il a été initialement formé, en partie en raison des différences d'utilisation entre les deux types de bâtiments.

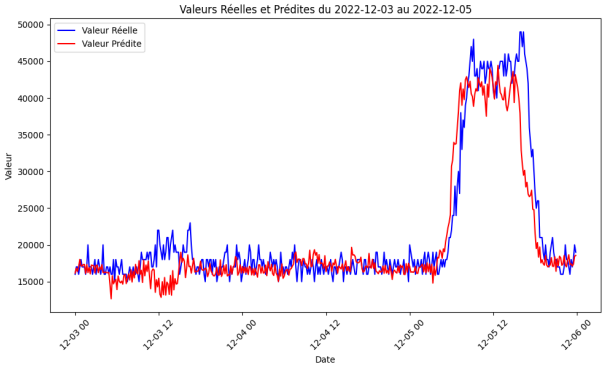
	Modèle	Séquence	MAE (Wh)	Moyenne des données réelles (Wh)	CV_MAE (%)
0	Modèle_bureau	sequence_9_train_bureau.csv	259.257585	22999.768519	1.127218
1	Modèle_bureau	sequence_test_bureau.csv	5378.125355	24135.879630	22.282699
2	Modèle_bureau	sequence_test_hopital_1.csv	401090.304355	456614.583333	87.840012
3	Modèle_bureau	sequence_test_hopital_2.csv	383648.389446	439253.240741	87.341049

Figure 3.17 – L'érreur moyenne absolue sur le modéle du bureau

La Figure 3.18 présente les résultats détaillés du test du modèle sur deux extraits de 3 jours des séquences de test des données du bureau sélectionnées aléatoirement. Les résultats du modèle sont satisfaisants, et il est remarquable qu'il ait appris les tendances hebdomadaires. Par exemple, sur la figure 3.18a, la courbe de charge du 03/12 au 05/12 correspond au premier week-end de décembre, où la consommation au bureau est minimale en raison de la fermeture. Le modèle a appris cette tendance et prédit correctement la consommation pendant les week-ends, tout en suivant la montée de la consommation en début de semaine.



(a) Application sur la période 05/12/2022 au 07/12/2022



(b) Application sur la période 03/12/2022 au 05/12/2022

Figure 3.18 – Comparaison des résultats prédictifs et des valeurs observées du modèle du bureau sur une séquence de bureau

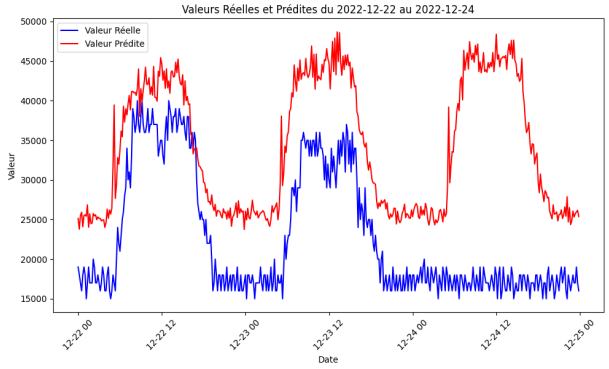
3.6.3 Modèle entrainé sur les données combinées

Après son entraînement sur les données de l'hôpital et du bureau, le modèle présente un coefficient de variation de l'erreur moyenne absolue (CV(MAE)) d'environ 10% lorsqu'il est évalué sur de nouvelles séquences hospitalières. Cela indique une adaptation réussie à de nouvelles données de l'hôpital. Cependant, une augmentation significative de l'erreur se produit lors des tests sur les données du bureau, à la fois pour les données déjà vues lors de l'entraînement (séquence 9) et les données de test. Une des raisons de ce biais envers les données du bureau est le déséquilibre des données, avec 12 séquences d'hôpital par rapport à seulement 9 séquences de bureau, ce qui se traduit par de meilleures performances sur la classe majoritaire.

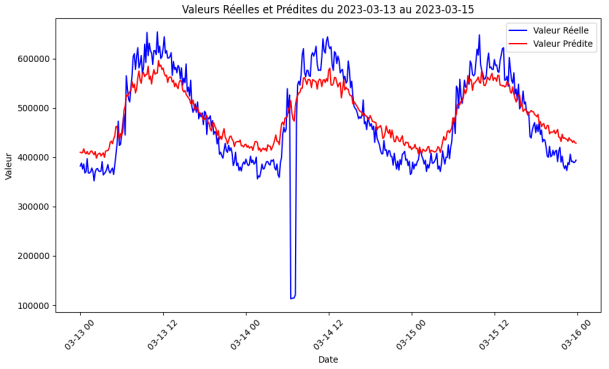
	Modèle	Séquence	MAE (Wh)	Moyenne des données réelles (Wh)	CV_MAE (%)
0	Modèle_combiné	sequence_9_train_bureau.csv	10595.380286	22999.768519	46.067334
1	Modèle_combiné	sequence_test_bureau.csv	10058.807284	24135.879630	41.675743
2	Modèle_combiné	sequence_test_hopital_1.csv	43172.765466	456614.583333	9.454969
3	Modèle_combiné	sequence_test_hopital_2.csv	44376.845841	439253.240741	10.102793

Figure 3.19 – L'érreur moyenne absolue sur le modéle combiné

La Figure 3.20 confirme cette observation. Le modèle entraîné sur les données combinées suit les tendances des données de l'hôpital, où la consommation pendant les week-ends ou en milieu de semaine est presque la même. Cependant, il semble avoir du mal à s'adapter aux variations spécifiques au bureau, comme le montre la courbe de charge du 24/12/22 (Fig. 3.20a), qui correspond à un samedi avec une consommation minimale au bureau. Le modèle prédit une augmentation de la consommation en suivant les tendances de l'hôpital, ce qui suggère une difficulté à généraliser efficacement aux spécificités des données du bureau.



(a) Application sur la période 22/12/2022 au 24/12/2022 pour une séquence de bureau



(b) Application sur la période 13/03/2023 au 15/03/2023 pour une séquence d'hopital

Figure 3.20 – Comparaison des résultats prédictifs et des valeurs observées du modèle combiné sur une séquence de bureau et de l'hopital

En résumé, l'analyse des modèles montre que chacun présente des performances différentes en fonction des données sur lesquelles il a été formé, mettant en évidence l'importance de la qualité et de la quantité des données d'entraînement, ainsi que des différences d'utilisation entre les bâtiments.

4 Conclusion

En conclusion, ce projet de recherche a permis de développer et de tester avec succès des modèles de prédiction de la consommation électrique basés sur les Réseaux de Neurones Récursifs (RNN). Notre objectif principal était de créer des modèles capables de prédire avec précision les courbes de charges électriques, en tenant compte des dépendances temporelles complexes présentes dans les données.

Au cours de ce projet, nous avons exploré en profondeur les principes fondamentaux des RNN, en mettant en évidence leur capacité à capturer les motifs récurrents et à modéliser les séquences de données temporelles. Nous avons également examiné les défis et les limitations de l'apprentissage profond, notamment la nécessité de données de haute qualité, les risques de surapprentissage et les exigences en matière de puissance de calcul.

Le développement du modèle de prédiction a impliqué la mise en œuvre de différents aspects, tels que la sélection et le traitement des données d'entraînement, l'exploration des caractéristiques des données de l'hôpital et du bureau, ainsi que la mise en forme et la préparation des données pour l'entraînement. Nous avons également décrit en détail le modèle lui-même, en mettant en évidence les portes de contrôle clés des RNN qui permettent de gérer les informations séquentielles.

L'entraînement des modèles a été réalisé sur des ensembles de données distincts provenant de l'hôpital et du bureau, ainsi que sur un ensemble combiné. Les résultats de la validation ont montré que les modèles étaient capables de généraliser efficacement à de nouvelles données, en particulier lorsqu'ils étaient entraînés sur des données de l'hôpital. Cependant, il est important de noter que la faible quantité de données disponibles ainsi que les contraintes en matière de ressources matérielles ont limité la complexité et la performance des modèles.

Malgré les succès obtenus, des défis subsistent, notamment la gestion des déséquilibres potentiels entre les ensembles de données provenant de différents types de bâtiments. En ce sens, une piste d'amélioration importante serait d'intégrer une deuxième entrée dans le modèle, prenant en compte les scénarios d'occupation des bâtiments. Cette approche pourrait potentiellement résoudre le problème de suivi des tendances de l'hôpital au détriment des données du bureau dans le modèle combiné. De plus, des recherches futures pourraient explorer d'autres techniques d'apprentissage automatique et d'autres modèles pour la prédiction de la consommation électrique, en tenant compte de la disponibilité de données plus riches et de ressources matérielles accrues.

En fin de compte, ce projet ouvre la voie à une utilisation plus intelligente et plus efficace de l'électricité dans divers contextes, contribuant ainsi à une gestion plus durable des ressources énergétiques. Les modèles prédictifs basés sur les RNN constituent un outil précieux pour anticiper les besoins en énergie électrique, avec des applications potentielles dans de nombreux secteurs, de la gestion des bâtiments à l'optimisation de la production d'énergie.