Projet Catégorie 3 : Prédiction de Séries Temporelles avec RNNs

Objectif Spécifique

L'objectif de ce projet est de résoudre un problème de **prédiction de séries temporelles** (régression) en utilisant des **Réseaux de Neurones Récurrents (RNNs)**. Vous travaillerez sur un **jeu de données temporel qui vous sera fourni** (parmi une sélection ci-dessous), par exemple pour prédire des températures futures, la consommation d'énergie, ou d'autres valeurs séquentielles. Votre tâche principale sera de prétraiter la série temporelle, de construire et de comparer différentes architectures RNN (`SimpleRNN`, `LSTM`, `GRU`), et d'analyser leurs performances pour la tâche de prédiction donnée.

*(Rappel : Référez-vous aux **Consignes Générales** pour les modalités de soumission, de documentation des expérimentations, d'auto-apprentissage, les contraintes techniques et les modalités de l'examen oral.)*

Jeux de Données Possibles

Votre groupe travaillera sur **un seul** des jeux de données suivants. Vous devrez utiliser les outils appropriés (`tensorflow_datasets`, API Kaggle, `pandas.read_csv` depuis un lien direct) pour charger les données dans votre notebook.

1. **Daily Temperature Time Series (Prédiction Météo Simple)**
   - **Tâche** : Prédire la température minimale quotidienne moyenne pour le jour suivant, en se basant sur les températures passées.
   - **Type** : Régression (Prédiction Univariée).
   - **Source** : Disponible sur Kaggle (ex: https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data - se concentrer sur une seule colonne comme `mean_temperature`).
   - **Considérations** : Un bon point de départ pour comprendre les bases de la prédiction temporelle.
2. **Air Quality (Prédiction de Pollution Simple)**
   - **Tâche** : Prédire la concentration d'un polluant spécifique (ex: PM2.5) pour l'heure suivante, en se basant sur les mesures passées (éventuellement incluant d'autres variables comme la température, le vent, etc.).
   - **Type** : Régression (Prédiction Univariée ou Multivariée simple).
   - **Source** : Disponible sur le dépôt UCI ML ou Kaggle (ex: "Beijing PM2.5 Data" - https://archive.ics.uci.edu/dataset/381/beijing+pm2+5+data). Nécessite potentiellement un nettoyage et une sélection de features.
   - **Considérations** : Plus complexe que la température, peut nécessiter la gestion de valeurs manquantes et la sélection de variables pertinentes si multivariée.
3. **Household Energy Consumption (Subset - Consommation Énergétique)**
   - **Tâche** : Prédire la puissance active globale ('Global_active_power') pour la prochaine minute (ou un autre intervalle), en se basant sur les minutes précédentes.

- **Type** : Régression (Prédiction Univariée).
- **Source** : Disponible sur le dépôt UCI ML (“Individual household electric power consumption Data Set” - https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption). **Attention : Dataset très volumineux.** Nécessite de ne sélectionner qu’une **période limitée** et de gérer les valeurs manquantes pour rester dans les contraintes du projet.
- **Considérations** : Données haute fréquence, nécessite une gestion attentive du volume de données et du prétraitement.

(L’énoncé final de votre projet confirmera le jeu de données exact que votre groupe devra utiliser.) N’oubliez pas de préciser dans le tabbleur le sujet que vous avez choisi.

Tâches Détaillées

1. **Chargement et Exploration des Données Fournies**
   - Chargez le jeu de données temporel assigné.
   - Explorez-le : structure, fréquence des données, tendances, saisonnalités (visualisation graphique essentielle), présence de valeurs manquantes.
2. **Prétraitement Spécifique aux Séries Temporelles (Étapes Clés)**
   - **Nettoyage** : Gérez les valeurs manquantes (imputation simple ou interpolation, *justifiez*).
   - **Normalisation/Standardisation** : Mettez à l’échelle les valeurs numériques. *Discutez de la stratégie : normaliser sur tout le set d’entraînement ou par fenêtre ?*
   - **Windowing (Fenêtrage)** : Transformez la série temporelle en un jeu de données supervisé en créant des fenêtres glissantes. Chaque fenêtre contiendra une séquence d’entrée (les `N` valeurs passées) et une cible de sortie (la valeur `M` pas de temps plus tard - souvent `M=1` pour prédire la prochaine valeur). Utilisez `tf.data.Dataset.window` ou des techniques NumPy/Pandas. *Justifiez la taille de la fenêtre choisie.*
   - **Préparation Finale** : Séparez vos données fenêtrées en ensembles d’entraînement, de validation et de test (attention à ne pas mélanger le temps lors de la séparation ! La validation et le test doivent venir *après* l’entraînement dans le temps).
3. **Expérimentation et Comparaison d’Architectures RNN (Obligatoire)**
   - **Modèles à Comparer (minimum)** : Construisez, entraînez et comparez rigoureusement les performances des modèles suivants :
     - Modèle 1 : Basé sur `tf.keras.layers.SimpleRNN`.
     - Modèle 2 : Basé sur `tf.keras.layers.LSTM`.
     - Modèle 3 : Basé sur `tf.keras.layers.GRU`.
   - **Exploration Architecturale (pour chaque type de RNN)** : Testez et documentez l’impact de :
     - Nombre d’unités RNN (`units`).
     - Empilement de couches RNN (utilisation de `return_sequences=True`).
     - Ajout de couches `Dense` et `Dropout` après la/les couche(s) RNN.
     - *(Optionnel)* Utilisation de `tf.keras.layers.Bidirectional`.

- **Couche de Sortie** : Une couche `Dense` avec 1 neurone et *pas* d'activation (régression linéaire sur la sortie du RNN).

4. **Entraînement et Compilation**
   - Compilez chaque modèle testé avec un optimiseur (`Adam` recommandé), une fonction de perte adaptée à la **régression** (`mse` ou `mae`), et des métriques de régression (`mae`, `rmse`).
   - Entraînez chaque modèle en utilisant les ensembles d'entraînement et de validation fenêtrés, avec des callbacks (`EarlyStopping`, `ModelCheckpoint`).
5. **Évaluation Comparative et Analyse**
   - Évaluez les *meilleures variantes* de chaque type de RNN (`SimpleRNN`, `LSTM`, `GRU`) sur l'ensemble de **test** fenêtré.
   - Comparez leurs performances (MSE, MAE, RMSE).
   - **Visualisez les prédictions** : Tracez les valeurs réelles et les prédictions de vos meilleurs modèles sur une portion de l'ensemble de test pour comparer visuellement leur qualité.
   - Analysez les résultats :
     - Quelle architecture RNN (`SimpleRNN`, `LSTM`, `GRU`) a le mieux fonctionné pour cette série temporelle ?
     - Pourquoi observez-vous ces différences (théorie vs pratique) ?
     - Quelles ont été les limites observées ?
     - Analyse des courbes d'apprentissage.

Exploration Guidée Attendue (Auto-Apprentissage)

Prétraitement Séries Temporelles :
- **Normalisation** : Quand et comment normaliser (avant/après windowing, sur quelles données) ?
- **Windowing** : Comprendre comment créer des fenêtres d'entrée/sortie. Explorer l'utilisation de `tf.data.Dataset.window`, `.flat_map`, `.batch` ou des alternatives NumPy/Pandas. Impact de la taille de la fenêtre.

Architectures RNN :
- Limitations `SimpleRNN` pour dépendances longues.
- Rôle des portes `LSTM`/`GRU`.
- Intérêt de `Bidirectional` (est-ce pertinent pour la *prédiction* future ? Discutez-en).

Évaluation Régression : Comprendre les métriques MSE, MAE, RMSE et leur interprétation. Savoir visualiser et analyser les prédictions vs les valeurs réelles.

Livrable Attendu (dans le Notebook Jupyter)

- Exploration du dataset temporel assigné (visualisations, statistiques).
- Code et justifications détaillées pour toutes les étapes de prétraitement (nettoyage, normalisation, **windowing**).
- Code et documentation rigoureuse des expérimentations comparant `SimpleRNN`, `LSTM`, `GRU` et l'exploration architecturale.
- Tableaux comparatifs clairs des performances finales (MSE, MAE, RMSE) sur

l’ensemble de test.
- Visualisations des prédictions des meilleurs modèles vs les valeurs réelles.
- Analyse critique comparative des architectures RNN pour *cette* tâche de prédiction.
- Section de conclusion (travail, apprentissages sur les RNNs et séries temporelles, difficultés).
- Liste des dépendances.