Ecole Polytechnique

Cas Technique
Prédiciton de la panne d'une turbine - NASA

Auteur :
Paul VARDON

4 mars 2018

Table des matières

Chapitre 1

Présentation du projet

1.1 Contexte

La NASA est l'agence gouvernementale américaine en charge de l'exploration spatiale et de la recherche aéronautique. A la pointe de l'innovation, ses programmes sont régulièrement confrontés à des problèmes de conception et de financement. Récemment, la fiabilité de ses turboréacteurs et les coûts de maintenance que cela implique ont été pointés du doigt. Une panne de moteur en plein vol devant absolument être évitée, l'agence inspecte fréquemment ses moteurs en suivant un programme uniforme de maintenance planifiée. Pourtant, les réacteurs ne vieillissent pas au même rythme : certaines réparations s'avèrent ainsi inutiles alors que d'autres opérations peuvent arriver trop tard, immobilisant l'appareil au sol et entraînant un manque à gagner considérable. Vous êtes mandaté par la NASA pour redéfinir le programme de maintenance. L'ambition est de mettre en place une solution de maintenance prédictive, qui consiste à anticiper le niveau de dégradation et déterminer la date de réparation la plus judicieuse. L'agence pourra éviter un maximum de pannes tout en minimisant le nombre d'interventions, et donc le coût de la maintenance.

1.2 Données disponibles

Pour définir un programme de maintenance prédictive, vous disposez d'un ensemble de données générées par les capteurs qui analysent les paramètres du moteur pendant le vol. Ces données sont des séries temporelles multivariées générées par des capteurs à chaque cycle d'opérations. Elles se décomposent en deux échantillons :

- L'échantillon d'entraînement (fichier "TrainSet") est constitué de données de 100 turbines du même type. Au début de la série temporelle, les turbines fonctionnent normalement. Elles commencent ensuite à présenter des comportements anormaux qui s'amplifient jusqu'à ce qu'elles tombent en panne. Ainsi, pour une machine donnée, la fin de la série temporelle correspond au dernier cycle avant l'occurrence de la panne.
- L'échantillon de test (fichier "TestSet"), est également constitué de 100 turbines du même type, mais la série temporelle se termine aléatoirement avant l'occurrence de la panne. Vous êtes ainsi dans les conditions réelles de fonctionnement de votre solution : le nombre de cycles restant avant l'incident est cette fois-ci à déterminer.

Dans chaque fichier, les colonnes renvoient aux éléments suivants :

- Colonne 1 : numéro de la turbine
- Colonne 2 : n-ème cycle opérationnel de la turbine
- Colonne 3 à 5 : configurations opérationnelles
- Autres colonnes : mesures des capteurs

1.3 Cahier des charges

Avant de mettre en œuvre votre solution dans les conditions réelles, la NASA veut s'assurer de votre capacité à prédire l'occurrence d'une panne. Pour la convaincre d'aller plus loin dans la collaboration, votre objectif est de rapprocher votre prédiction de la réalité constatée par ailleurs par la NASA (Fichier "TestSet_RUL"). L'indicateur de performance à minimiser pour votre client est donc l'erreur moyenne au carré (MSE – Mean Squared Error) entre le nombre prédit par votre modèle et la réalité opérationnelle. En plus de la performance du modèle, votre client sera particulièrement attentif à la qualité de votre code (lisibilité et documentation), à votre créativité et à l'interprétation que vous ferez des résultats.

1.4 Format de restitution

Vous avez été invité à envoyer les premiers résultats de vos travaux à votre client. Ses attentes sont les suivantes :

- Qualité des données mises à votre disposition (statistiques descriptives)
- Présentation de votre modèle : choix de l'algorithme, performances
- Proposition de prochaines étapes : pistes d'amélioration du modèle, évaluation des potentiels risques et dérives liés à la mise en place effective d'une telle solution

Chapitre 2

Analyse statistique des données

Dans cette partie, nous allons conduire une analyse statistique des données fournies par la NASA.

Répartition des pannes Tout d'abord, nous nous intéressons à la répartition des pannes (i.e. temps avant la première panne) pour chaque turbine, dans l'échantillon d'entraînement.

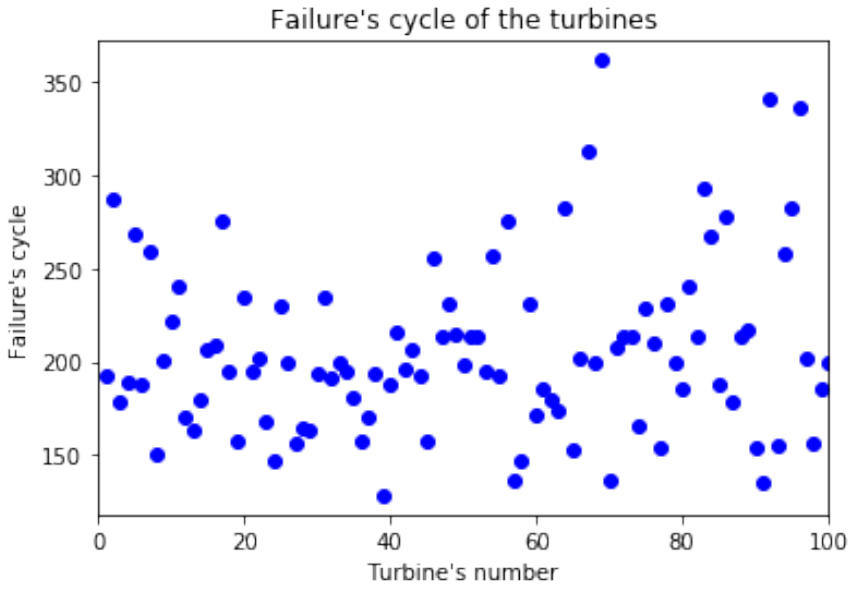

FIGURE 2.1 – Temps avant la première panne (train set)

On observe une répartition des données étendue et sans hiérarchie apparente.

Grandeur	Valeur
Moyenne	206.31
Ecart-type	46.34
Minimum	128
1e quartile	177
Médiane	199
3e quartile	229
Maximum	362

FIGURE 2.2 – Analyse statistique de l'occurrence des pannes

Données manquantes Il n'y a pas de données manquantes.

Comportement des paramètres Dans cette partie, on s'intéresse plus spécifiquement au comportement d'un paramètre au cours des cycles pour une turbine donnée.

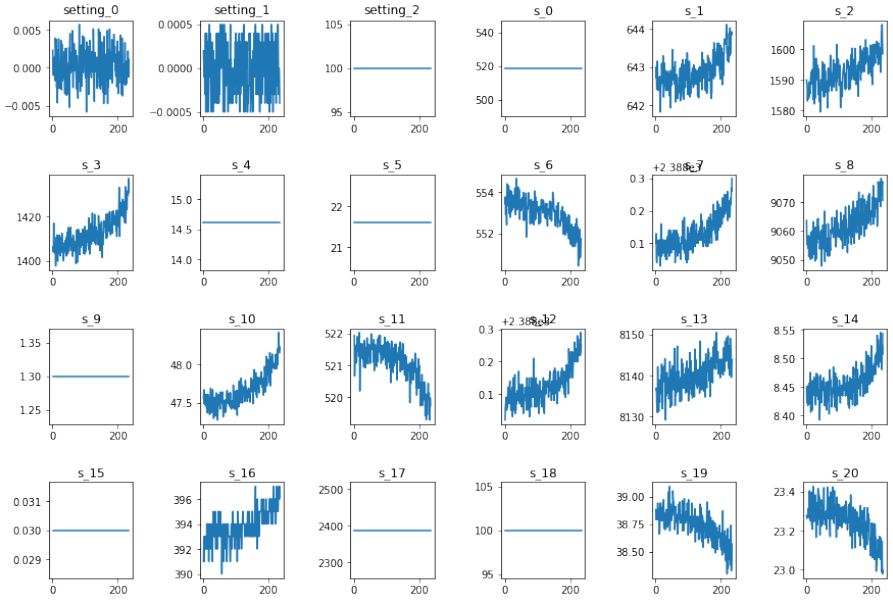

FIGURE 2.3 – Évolution des paramètres de la turbine 20 du set d'entraînement

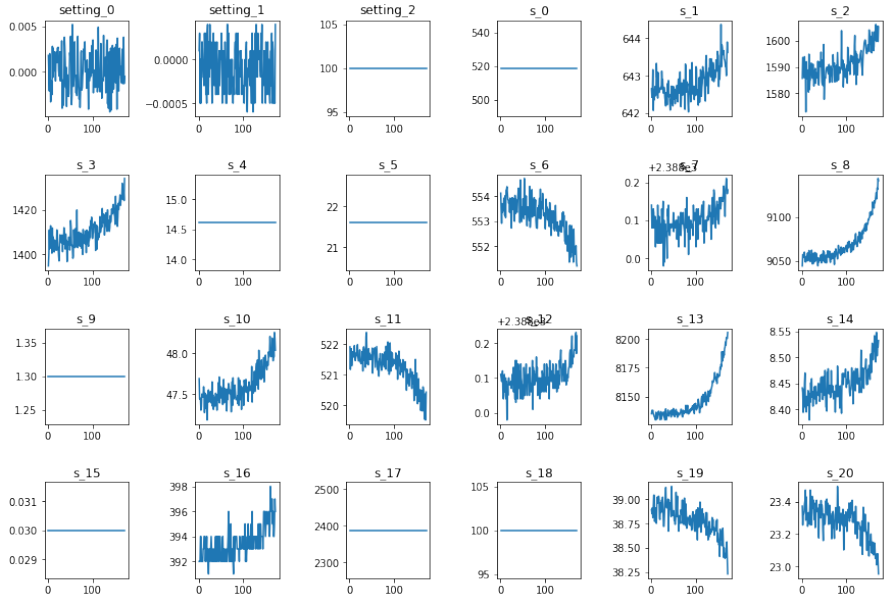

FIGURE 2.4 – Évolution des paramètres de la turbine 60 du set d'entraînement

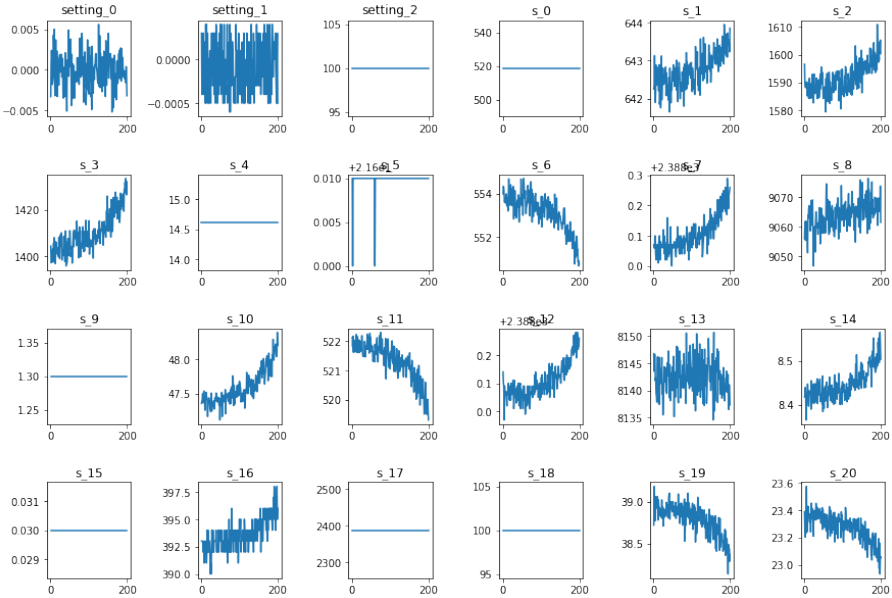

FIGURE 2.5 – Évolution des paramètres de la turbine 100 du set d'entraînement

Analyse statistique des paramètres Nous allons nous intéresser plus spécifiquement à la répartition de chaque paramètre au sein des deux sets.

Paramètre	Moyenne	Écart-type	Minimum	1e quartile	Médiane	3e quartile	Maximum
setting_0	-8.870e-06	0.002	-0.0087	-0.0015	0.0	0.0015	0.0087
setting_1	2.35e-06	0.00029	-0.0006	-0.00019	0.0	0.0003	0.0006
setting_2	100	0	100	100	100	100	100
s_0	518.669	0.0	518.669	518.669	518.669	518.669	518.669
s_1	642.68	0.5	641.21	642.325	642.64	643.0	644.53
s_2	1590.52	6.13	1571.04	1586.26	1590.1	1594.38	1616.91
s_3	1408.93	9	1382.25	1402.36	1408.04	1414.55	1441.49
s_4	14.62	0.0	14.62	14.62	14.62	14.62	14.62
s_5	21.60	0.0013	21.6	21.61	21.61	21.61	21.61
s_6	553.36	0.885	549.84	552.8	553.44	554.01	556.05
s_7	2388.09	0.07	2387.89	2388.05	2388.09	2388.13	2388.56
s_8	9065.242	22.082	9021.73	9053.099	9060.66	9069.41	9244.58
s_9	1.3	0.0	1.3	1.3	1.3	1.3	1.3
s_10	47.54	0.267	46.84	47.34	47.50	47.7	48.52
s_11	521.41	0.73	518.69	520.96	521.47	521.95	523.38
s_12	2388.09	0.071	2387.87	2388.04	2388.09	2388.13	2388.56
s_13	8143.75	19.07	8099.9	8133.24	8140.54	8148.31	8293.71
s_14	8.44	0.0375	8.32	8.41	8.43	8.46	8.58
s_15	0.03	0.0	0.03	0.03	0.03	0.03	0.023
s_16	393.21	1.54	388.0	392.0	393.0	394.0	400.0
s_17	2388.0	0.0	2388.0	2388.0	2388.0	2388.0	2388.0
s_18	100.0	0.0	100.0	100.0	100.0	100.0	100.0
s_19	38.81	0.18	38.13	38.7	38.83	38.95	39.43
s_20	23.28	0.108	22.89	23.22	23.29	23.36	23.61

FIGURE 2.6 – Analyse statistique de l'occurrence des pannes dans le set d'entraînement

Paramètre	Moyenne	Écart-type	Minimum	1e quartile	Médiane	3e quartile	Maximum
setting_0	-1.11e-05	0.00	-0.01	-0.00	0.0	0.00	0.01
setting_1	4.23e-06	0.0003	-0.0006	-0.0002	0.0	0.0003	0.0007
setting_2	100.0	0.0	100.0	100.0	100.0	100.0	100.0
s_0	518.66	0.0	518.66	518.66	518.66	518.66	518.66
s_1	642.47	0.40	641.13	642.19	642.46	642.73	644.29
s_2	1588.09	5.00	1569.04	1584.59	1587.98	1591.36	1607.55
s_3	1404.73	6.68	1384.39	1399.94	1404.43	1409.05	1433.35
s_4	14.61	0.0	14.61	14.61	14.61	14.61	14.61
s_5	21.60	0.0017	21.6	21.61	21.61	21.61	21.61
s_6	553.7	0.68	550.88	553.30	553.79	554.23	555.84
s_7	2388.07	0.0574	2387.88	2388.03	2388.07	2388.11	2388.3
s_8	9058.40	11.43	9024.53	9051.01	9057.32	9064.11	9155.03
s_9	1.29	0.0	1.29	1.29	1.29	1.29	1.29
s_10	47.41	0.19	46.79	47.27	47.40	47.54	48.25
s_11	521.74	0.55	519.38	521.38	521.78	522.15	523.76
s_12	2388.07	0.056	2387.88	2388.03	2388.07	2388.11	2388.32
s_13	8138.94	10.18	8108.5	8132.31	8138.39	8144.35	8220.48
s_14	8.42	0.029	8.33	8.40	8.42	8.44	8.54
s_15	0.029	0.0	0.029	0.029	0.029	0.029	0.029
s_16	392.57	1.23	389.0	392.0	393.0	393.0	397.0
s_17	2388.0	0.0	2388.0	2388.0	2388.0	2388.0	2388.0
s_18	100.0	0.0	100.0	100.0	100.0	100.0	100.0
s_19	38.89	0.14	38.31	38.79	38.90	38.99	39.40
s_20	23.33	0.084	22.93	23.28	23.33	23.39	23.64

FIGURE 2.7 – Analyse statistique de l'occurrence des pannes dans le set de test

Chapitre 3

Instructions pour lancer le projet

Dans cette partie, nous allons présenter comment générer les résultats à partir des notebook python.

Les notebooks sont codés en python et sont au nombre de trois :
— preprocess.ipynb : ce fichier réalise le pré-traitement des échantillons d'entraînement et de test. Il retourne les fichiers train.csv et test.csv.
— trainNN.ipynb : ce fichier crée et adapte le réseau de neurones à l'échantillon d'entraînement. Il nécessite la présence du fichier train.csv. Il retourne des fichiers d'images et le fichier nom_de_la_modélisation.h5.
— main.ipynb : ce fichier réalise la prédiction par le réseau de neurones de l'échantillon de test et la compare avec les relevés réels. Il nécessite un fichier de modélisation (fichier en .h5, dont le nom est modifiable au début du fichier main.ipynb) et le fichier test.csv. Il retourne un fichier submit_test.csv qui contient la prédiction finale.

Lancement du programme

Afin de lancer les fichiers, trois configurations sont possibles, présentées ci-dessous.

Pas de nouvelles données Il s'agit là de générer les mêmes résultats que ceux déjà disponibles dans le dossier output. Il faut dans ce cas lancer, à la suite et dans cet ordre, les fichiers preprocess.ipynb, trainNN.ipynb et main.ipynb.

Nouvelle prédiction de panne Une correction des prédictions des pannes est disponible. Il faut la placer, en remplacement de l'ancien fichier, dans le dossier input, puis relancer à la suite et dans cet ordre les fichiers preprocess.ipynb et main.ipynb.

Nouvel échantillon de test Un nouvel échantillon de test est disponible. Il faut le placer, en remplacement de l'ancien, dans le dossier input, puis relancer à la suite et dans cet ordre les fichiers preprocess.ipynb et main.ipynb.

Nouvel échantillon d'entraînement Un nouvel échantillon d'entraînement est disponible. Il faut le placer, en remplacement de l'ancien, dans le dossier input, puis relancer, à la suite et dans cet ordre, les fichiers preprocess.ipynb, trainNN.ipynb et main.ipynb.

Chapitre 4

Algorithme de prédiction

Dans cette partie, nous présenterons en détail les choix algorithmiques faits pour obtenir une prédiction des cycles de panne des turbines. Pour cela, nous prendrons, fichier par fichier, les opérations réalisées.

4.1 Preprocessing des données

Cette étape consiste à nettoyer et modifier les données avant leur intégration dans l'algorithme de prédiction. Elle concerne à la fois les sets d'entraînement et de test. Elle est réalisée par le fichier preprocess.ipynb.

Chargement des données La première étape consiste à récupérer, dans le dossier input, les fichiers contenant les données sur lesquelles nous allons travailler. Ces fichiers sont au format texte, chaque ligne étant délimité par un saut de ligne et chaque colonne par un espace. Les noms donnés aux colonnes dépendent de la nature des paramètres spécifiés dans le cahier des charges.

Transformation du set d'entraînement Cette étape s'effectue sur le set d'entraînement. Trois nouvelles grandeurs sont construites :

- RUL (Remaining Useful Life) : ce paramètre correspond aux nombres de cycles restants avant la panne de la machine
- label1 : il s'agit d'un paramètre qui n'est pas utilisé en tant que tel ici. Il représente l'état de la turbine au cycle donné par rapport à l'hypothèse "la turbine va tomber en panne d'ici n cycles". Il peut être utile pour l'implémentation (non réalisée ici) d'une classification binaire.
- label2 : similaire au précédent, ce paramètre n'est pas non plus utilisé.

Une fois cette étape passé, on normalise les colonnes, sauf le numéro de la turbine. Pour le numéro de cycle, on ajoute un paramètre de cycle normalisé, tout en gardant le précédent pour des raisons d'indexage. L'étape de normalisation (ou feature scaling en anglais) est nécessaire au vu des valeurs prises par les différents paramètres : en effet, un paramètre dont la moyenne serait très élevée par rapport à celle d'un autre paramètre pourrait compter pour plus significatif dans l'algorithme d'apprentissage, ce qui ne reflète pas nécessairement la réalité de ces paramètres. Enfin, on exporte ce nouveau jeu de données dans un fichier .csv qui sera facilement exploitable par la suite.

Transformation du set de test La transformation du set d'entraînement se déroule de la même manière que précédemment. La seule différence réside dans l'origine des informations, puisque la prévision des pannes n'est disponible que dans le fichier de prédiction, et non directement dans le fichier de test.

Une fois ces étapes passées, on peut s'intéresser à l'entraînement du réseau de neurones.

4.2 Entraînement du réseau de neurones

Cette étape constitue le cœur du problème. Il s'agit de générer, depuis le set d'entraînement, un modèle prédictif pour le set de test.

Formatage du set d'entraînement S'il ne s'agit plus à proprement parlé de preprocessing, il demeure nécessaire d'adapter les données d'entraînement à l'algorithme choisi. Pour cela, on importe le set d'entraînement préalablement généré, puis on le formate pour correspondre à la taille imposée (nombre de turbines, nombre de cycle (fixé), nombre de paramètres). La fonction reshapeFeatures permet de transformer, pour une turbine

donnée, l'ensemble de ses cycles en 142 (i.e. 192 - 50) paquets de 50 cycles (de 1 à 51, de 2 à 52, etc..). La fonction reshapeLabel attribue aux paquets précédents le bon label, i.e. la valeur de RUL du dernier cycle du paquet (paquet 1-51 => RUL 51, paquet 1-52 => RUL 52, etc...).

Création du réseau de neurones Le réseau de neurones choisi est basé sur la technologie de réseaux de neurones récurrents, et plus précisément sur les réseaux LSTMs (i.e. Long Short Term Memory). Ces réseaux permettent de gérer la dépendance des paramètres au cours des cycles, et de doter l'algorithme d'apprentissage d'une mémoire à court et moyen terme. Le réseau de neurones est donc construit comme suit :

- Première couche : LSTM(100, 50, 25)
- Deuxième couche : Dropout. Cette couche vise à réduire l'overfitting.
- Troisième couche : LSTM(50). Cette couche intermédiaire comporte 50 nœuds.
- Quatrième couche : Dropout
- Cinquième couche : LSTM(25). Cette couche intermédiaire comporte 25 nœuds.
- Sixième couche : Dropout
- Septième couche : Dense(1) & Activation(linear) : cette couche est la couche finale, avec une seule sortie (la valeur de RUL) et un activation linéaire puisqu'il s'agit d'un problème de regression.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_0 (LSTM)                (None, 50, 100)           50400
_________________________________________________________________
dropout_0 (Dropout)          (None, 50, 100)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 50, 50)            30200
_________________________________________________________________
dropout_1 (Dropout)          (None, 50, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 25)                7600
_________________________________________________________________
dropout_2 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_0 (Dense)              (None, 1)                 26
_________________________________________________________________
activation_0 (Activation)    (None, 1)                 0
=================================================================
Total params: 88,226
Trainable params: 88,226
Non-trainable params: 0
_________________________________________________________________
None
```

FIGURE 4.1 – Le réseau de neurones récurrents (Long Short Term Memory)

L'étape suivante consiste à entrainer le réseau de neurones (cette étape est longue et coûteuse en calcul). Le réseau de neurones est ensuite enregistré dans le fichier output.

4.3 Prédiction du test par l'algorithme

Cette étape, que l'on retrouve dans le fichier main.py, se concentre sur la prédiction, à partir du réseau de neurones entraîné, du fichier de test.

Pour chaque turbine, on récupère les 50 derniers cycles de chaque turbine (les turbines qui ont moins de 50 cycles sont éliminées du processus). On peut alors prédire le label RUL à partir des données du set de test.

Chapitre 5

Résultats

5.1 Entraînement du réseau de neurones

Tout d'abord, on peut estimer l'efficacité du réseau de neurones en séparant le set d'entraînement entre un set train et un set de validation. On peut alors évaluer l'efficacité du réseau de neurones sur ces deux sets. L'entraînement du réseau de neurones est réalisé, ainsi que demandé, avec une fonction de coût en MSE (Mean Square Error). Pour évaluer, nous prenons, les métriques RMSE (Root Mean Square Error) et MAE (Mean Absolute Error), qui offrent des informations intéressantes et comparables sur l'erreur du modèle.

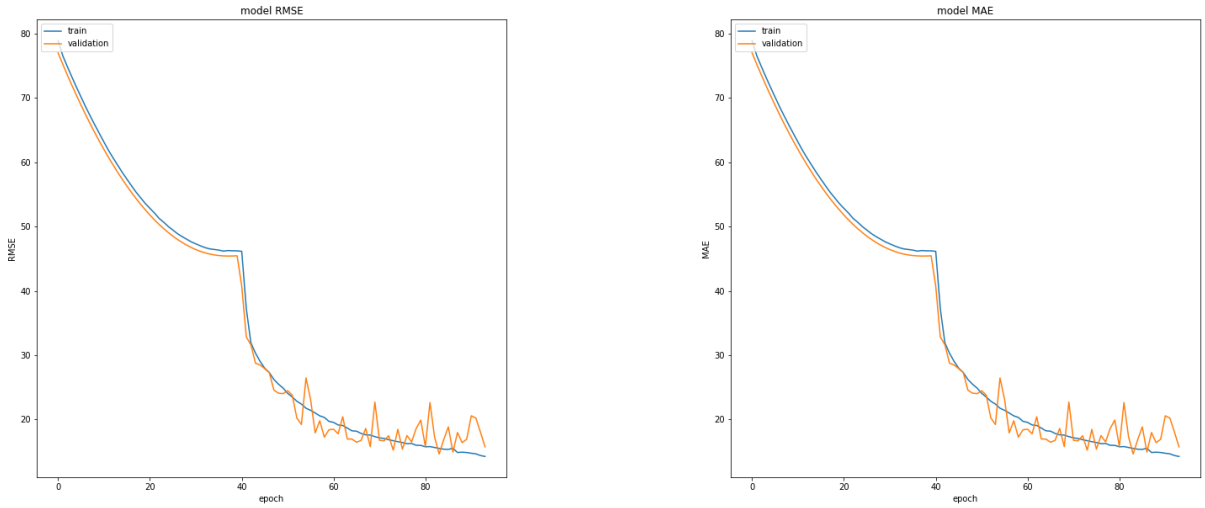

FIGURE 5.1 – Évolution de la RMSE (à gauche) et la MAE (à droite)

On observe une décroissance forte de la RMSE avec la croissance des epochs. Il est donc judicieux de construire un modèle avec un nombre relativement grand de passage du réseau neuronal. Dans ce cas précis, la valeur de l'epoch était fixée à 100, pour une taille de batch à 200, ce qui résultait en un temps de calcul d'approximativement 50 minutes.

5.2 Résultats finaux

L'algorithme sur le set de test donne le résultat suivant :
On peut s'intéresser aux prédictions effectives de l'algorithme par rapport à la réalité.

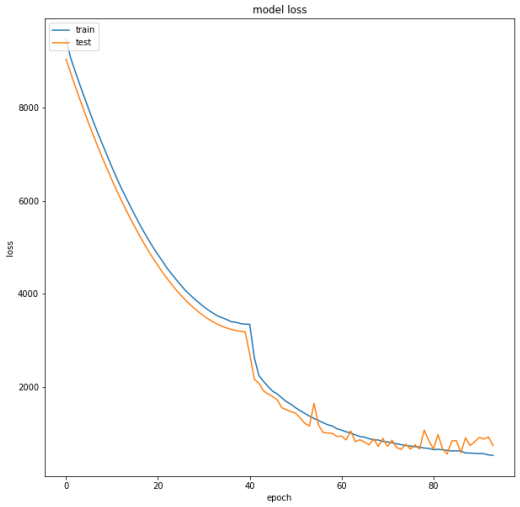

FIGURE 5.2 – Évolution de la fonction de coût (MSE)

Métrique	Résultat
Mean Square Error	303.49
Root Mean Square Error	11.65
Mean Absolute Error	11.65
Efficiency	85.79%

FIGURE 5.3 – Résultats du réseau de neurones

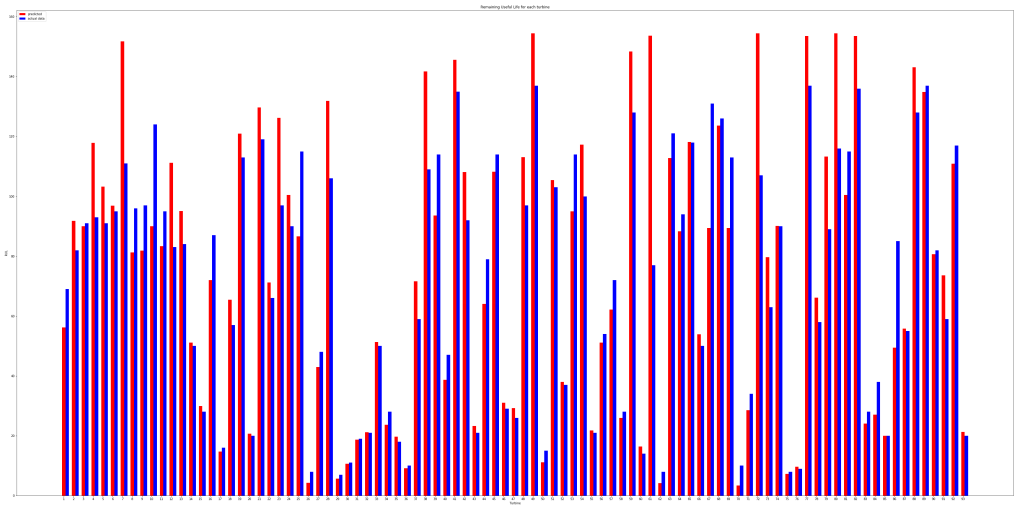

FIGURE 5.4 – RUL par turbine prédite (en bleu) et réelle (en rouge)

Chapitre 6

Conclusion

6.1 Bilan des résultats

Le réseau de neurones semble être une solution adaptée au problème. Assez flexible, cette technologie permet d'obtenir des résultats satisfaisants, avec une erreur à la moyenne de ? ?. Il serait intéressant de comparer ce chiffre à d'autres modèles de prédiction, ou encore à la situation actuelle de révision des machines. Afin d'avoir une idée de l'efficacité intrinsèque de la méthode, on peut calculer le efficience de la prédiction, dont la formule est :

$$EF = 1 - \frac{\sum(Ypred_i - Ytest_i)}{\sum(Ypred_i - \widehat{Ypred})}$$

Dans le cas présent, l'efficience est de 85.79%.

On peut donc conclure que l'approche prédictive par un réseau de neurones est une approche satisfaisante, qui permet une prédiction utile aux pannes des turbines et qui favorise une prise en charge efficace de l'entretien des machines.

6.2 Risques de la méthode

Cette méthode prédictive n'est cependant pas sans risque. En effet, elle ne permet pas de prédire totalement, et avec certitude l'occurrence d'une panne. Elle offre une expertise supplémentaire quant aux réparations à effectuer, mais ne peut en l'état ni systématiser un processus, ni constituer l'unique estimateur du calendrier de maintenance. Il convient de familiariser les équipes de maintenance à cet outil prévisionnel afin de cibler plus efficacement leurs réparations sur des turbines qui pourraient potentiellement tomber en panne.

6.3 Améliorations possibles

A ce stade, trois axes d'amélioration sont à envisager :

— Prise en compte de toutes les turbines. Ce défaut, structurel à l'algorithme employé, provoque un rejet des turbines (d'entraînement, ce qui est moindre mal, mais aussi de test) dont le nombre de cycles est inférieur à 50. Il est donc nécessaire de laisser les turbines tourner un minimum de 50 cycles si l'on veut obtenir une prédiction quant à leur défaillance. Ce seuil peut être baisser, mais cette opération se fait au détriment de la précision de la prédiction.

— Optimisation des paramètres. Le réseau de neurones utilisé dans l'algorithme est assez standard, et n'a pas été, à quelques exceptions près, affiné pour la situation en jeu ici. Il serait donc judicieux d'ajuster ces hyper-paramètres afin de s'adapter aux données disponibles. En outre, l'ajout de couches internes pourraient aller aussi dans ce sens. Il convient cependant de garder à l'esprit que ces opérations sont extrêmement coûteuses en ressources informatiques et la génération du réseau de neurones pourraient devenir handicapante en cas de complexité trop importante du réseau de neurones.

— Nouveau critère d'évaluation. Jusqu'ici, il était question de prédire le cycle auquel une turbine risquait de tomber en panne. Observer d'autres critères, comme la probabilité qu'une machine a de tomber en panne sur un nombre de cycle donné, pourrait faciliter les calculs et apporter une précision supplémentaire. L'algorithme est en partie orienté vers cette étape à travers les grandeurs crées label1 et label2. Il faudrait adapter par la suite le réseau de neurones lui-même pour implémenter cette détection. Un tel choix d'évaluation de la maintenance permettrait peut-être un systématisation plus simple de la solution

développée, puisqu'un seuil de tolérance serait établi, au-delà duquel une réparation aurait toujours lieu, prévenant ainsi de la situation très coûteuse d'une turbine qui tombe effectivement en panne.