N° 2025

École Nationale des Sciences Appliquées

DÉPARTEMENT GÉNIE MÉCANIQUE

UNIVERSITÉ SIDI BEN ABDELLAH

- Fes -

Option : **Génie Mécanique**

Application de L'IA en Mécanique : Maintenance Prédictive

Réalisé par | ARKHIS M'HAMMED
RADOUANE ELARFAOUI
FARID ELABISISI
AMYNE ED-DARIF
SABANE MOHAMMED
KHALID LOULIJET

Encadré par | PR. AZIZ ELMRABET

Année Universitaire : **2025 / 2026**

Chapitre 1

Introduction

Ce projet explore l'application des techniques d'Intelligence Artificielle, spécifiquement les réseaux de neurones, au domaine de l'ingénierie mécanique. L'objectif principal est de mettre en place un système de maintenance prédictive capable d'anticiper les pannes de machines industrielles.

Nous utilisons l'environnement MATLAB et ses boîtes à outils spécialisées (Neural Network Toolbox) pour développer et entraîner nos modèles.

Chapitre 2

Contexte et Objectifs

2.1 Problématique

Dans l'industrie moderne (Industrie 4.0), la fiabilité des équipements est cruciale. Les pannes inattendues entraînent des arrêts de production coûteux, des pertes de matières premières et parfois des risques pour la sécurité des opérateurs.

Les approches traditionnelles de maintenance présentent des limites importantes :

— **Maintenance corrective :** Elle consiste à intervenir uniquement après la panne. Cette approche est réactive et engendre des coûts d'arrêt de production non maîtrisés.

— **Maintenance préventive :** Elle repose sur des interventions planifiées à intervalles fixes (ex : toutes les 1000 heures). Bien que plus sûre, elle conduit souvent à remplacer des composants encore sains, ce qui constitue un gaspillage économique.

La problématique centrale de ce projet est donc la suivante : *Comment utiliser les données collectées par les capteurs industriels pour anticiper les défaillances avec précision et n'intervenir qu'au moment nécessaire ?*

2.2 Objectif du Projet

L'objectif est de développer un système de maintenance prédictive intelligent. Il s'agit de concevoir un modèle de réseau de neurones capable de classifier l'état de la machine en temps réel (0 = Fonctionnel, 1 = Panne) à partir des paramètres opérationnels.

Chapitre 3

Méthodologie

3.1 Description des Données

Le jeu de données utilisé pour ce projet est un enregistrement détaillé des conditions de fonctionnement d'une machine-outil. Chaque ligne du fichier `machine failure.csv` représente un cycle de production et contient les informations collectées par divers capteurs.

3.1.1 Variables Explicatives (Features)

Les variables d'entrée, utilisées pour prédire l'état de la machine, sont les suivantes. Elles simulent les données qu'on obtiendrait de capteurs industriels.

- **Type de produit (colonne 'Type')** : Indique la variante de qualité du produit fabriqué (L = Faible, M = Moyenne, H = Haute). Cette variable catégorielle a un impact sur les contraintes (vitesse, couple) appliquées à la machine.
- **Température de l'air [K] (colonne 'Air temperature [K]')** : Mesure la température ambiante autour de la machine.
  - **Capteur associé :** Un **thermomètre** ou une **sonde de température** placé à proximité de la machine.
- **Température du processus [K] (colonne 'Process temperature [K]')** : Représente la température de la pièce en cours d'usinage ou de l'outil. C'est un indicateur critique de la santé du processus.
  - **Capteur associé :** Un **thermocouple** en contact avec la pièce ou un **pyromètre infrarouge** qui mesure la température à distance.
- **Vitesse de rotation [rpm] (colonne 'Rotational speed [rpm]')** : Indique la vitesse de rotation de la broche principale de la machine, en tours par minute.
  - **Capteur associé :** Un **tachymètre** ou un **codeur rotatif** monté sur l'axe de la broche.
- **Couple [Nm] (colonne 'Torque [Nm]')** : Mesure la force de torsion appliquée par la broche sur l'outil. Une augmentation anormale du couple peut signaler une usure d'outil ou un problème de coupe.
  - **Capteur associé :** Un **capteur de couple** intégré à la broche, ou une estimation calculée à partir du courant consommé par le moteur de la broche.
- **Usure de l'outil [min] (colonne 'Tool wear [min]')** : Représente le temps d'utilisation cumulé de l'outil de coupe. Ce n'est pas une mesure directe de capteur,

mais une valeur suivie par le système de contrôle de la machine (automate programmable) pour estimer l'état d'usure.

3.1.2 Variable Cible (Target)

La variable que nous cherchons à prédire est :

— **Machine Failure (colonne 'Machine failure')** : Une variable binaire qui indique si une panne est survenue durant le cycle. **0** signifie un fonctionnement normal, et **1** signifie qu'une défaillance a eu lieu. C'est la variable cible principale de notre modèle.

3.1.3 Autres Informations Contextuelles

Le jeu de données contient également des colonnes qui ne sont pas utilisées pour l'entraînement du modèle, mais qui donnent un contexte sur le type de panne.

— **UDI** et **Product ID** : Des identifiants uniques pour chaque enregistrement et produit.
— **TWF (Tool Wear Failure)** : Indicateur binaire signalant une panne due à l'usure de l'outil.
— **HDF (Heat Dissipation Failure)** : Indicateur de panne liée à une surchauffe.
— **PWF (Power Failure)** : Indicateur de panne due à un problème d'alimentation électrique.
— **OSF (Overstrain Failure)** : Indicateur de panne causée par une contrainte excessive sur l'outil.
— **RNF (Random Failure)** : Indicateur pour les pannes aléatoires.

Dans notre projet, la colonne `MachineFailure` est une agrégation de ces différents types de pannes.

3.2 Outils Utilisés

Nous avons utilisé l'environnement MATLAB pour l'ensemble du projet, de la préparation des données à l'entraînement du réseau de neurones.

— **Langage :** MATLAB
— **Manipulation de données :** Matrices et tableaux MATLAB
— **Modélisation :** Neural Network Toolbox ('fitnet', 'nntool')
— **Visualisation :** Fonctions graphiques natives ('plot', 'confusionchart')

3.3 Implémentation sous MATLAB

Cette section détaille les étapes clés de l'implémentation du projet en utilisant le langage MATLAB.

3.3.1 Chargement et Préparation des Données

La première étape consiste à charger les données depuis le fichier CSV et à les préparer pour le réseau de neurones. Contrairement à Python où les données sont souvent

structurées en (Samples x Features), la *Neural Network Toolbox* de MATLAB attend généralement des données en (Features x Samples).

```
% Chargement des données
filename = 'machine failure.csv';
opts = detectImportOptions(filename);
opts.SelectedVariableNames = {'Type', 'AirTemperature_K_', ...
    'ProcessTemperature_K_', 'RotationalSpeed_rpm_', ...
    'Torque_Nm_', 'ToolWear_min_', 'MachineFailure'};
data = readtable(filename, opts);

% Conversion de la variable catégorielle 'Type' en numérique
data.Type = grp2idx(data.Type);

% Séparation Features (X) et Target (Y) et Transposition
X = [data.Type, data.AirTemperature_K_, data.ProcessTemperature_K_, ...
     data.RotationalSpeed_rpm_, data.Torque_Nm_, data.ToolWear_min_]';
Y = data.MachineFailure';
```

Listing 3.1 – Chargement et prétraitement des données

Nous utilisons `readtable` pour lire le fichier et `grp2idx` pour convertir la colonne catégorielle "Type" (L, M, H) en indices numériques (1, 2, 3), ce qui est nécessaire pour le traitement par le réseau de neurones.

3.3.2 Architecture du Réseau de Neurones

Nous utilisons la fonction `patternnet` qui est spécifiquement conçue pour les problèmes de reconnaissance de formes (classification). L'architecture choisie comporte deux couches cachées de 100 et 50 neurones respectivement.

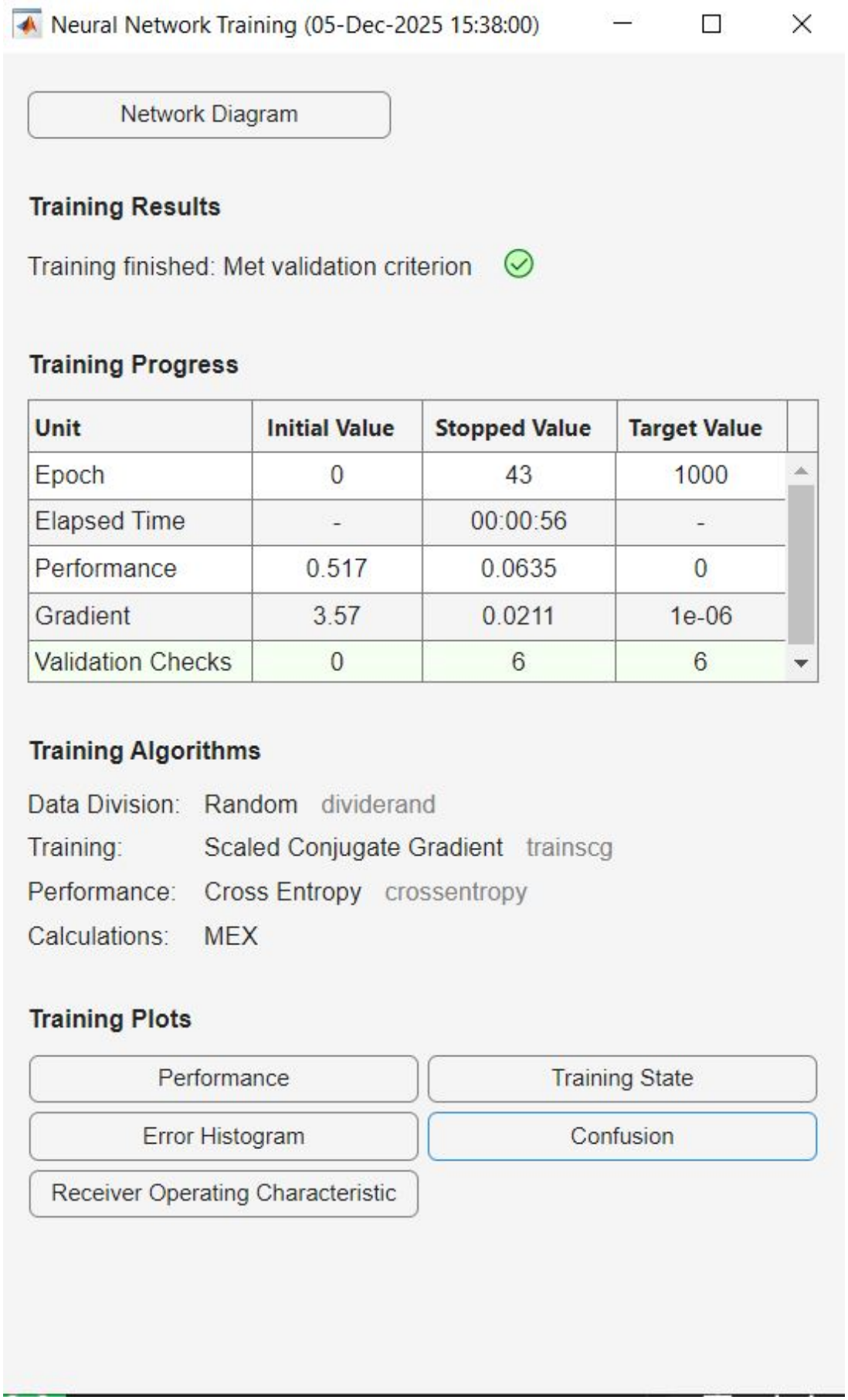

FIGURE 3.1 – Résumé de l'outil d'entraînement (MATLAB)

```
% Création d'un réseau de reconnaissance de motifs
hiddenLayerSize = [100, 50];
net = patternnet(hiddenLayerSize);
```

Listing 3.2 – Création du réseau de neurones

3.3.3 Entraînement

Avant l'entraînement, nous définissons la répartition des données : 70% pour l'entraînement, 15% pour la validation et 15% pour le test. La fonction `train` se charge ensuite d'optimiser les poids du réseau.

```
% Division des données
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Entraînement
[net, tr] = train(net, X, Y);
```

Listing 3.3 – Configuration et entraînement

3.3.4 Évaluation

Une fois le modèle entraîné, nous évaluons ses performances sur le jeu de test (données non vues durant l'entraînement).

```
% Prédiction sur les données de test
XTest = X(:, tr.testInd);
YTest = Y(:, tr.testInd);
YPred = net(XTest);

% Conversion des probabilités en classes (0 ou 1)
YPredClass = round(YPred);

% Calcul de la précision (Accuracy)
accuracy = sum(YPredClass == YTest) / length(YTest);
fprintf('Précision sur le jeu de test : %.2f%%\n', accuracy * 100);
```

Listing 3.4 – Évaluation du modèle

Nous utilisons la fonction `plotconfusion` pour visualiser les performances détaillées.

Chapitre 4

Résultats et Analyse

4.1 Performance du Modèle

L'évaluation du modèle est réalisée sur l'ensemble de test, qui représente 15% des données et n'a pas été utilisé durant l'entraînement. Les performances sur cet ensemble sont donc le meilleur indicateur de la capacité du modèle à généraliser sur de nouvelles données.

Le modèle atteint une précision globale (*Accuracy*) de **97.5%** sur l'ensemble de test, ce qui indique un très bon taux de classification générale.

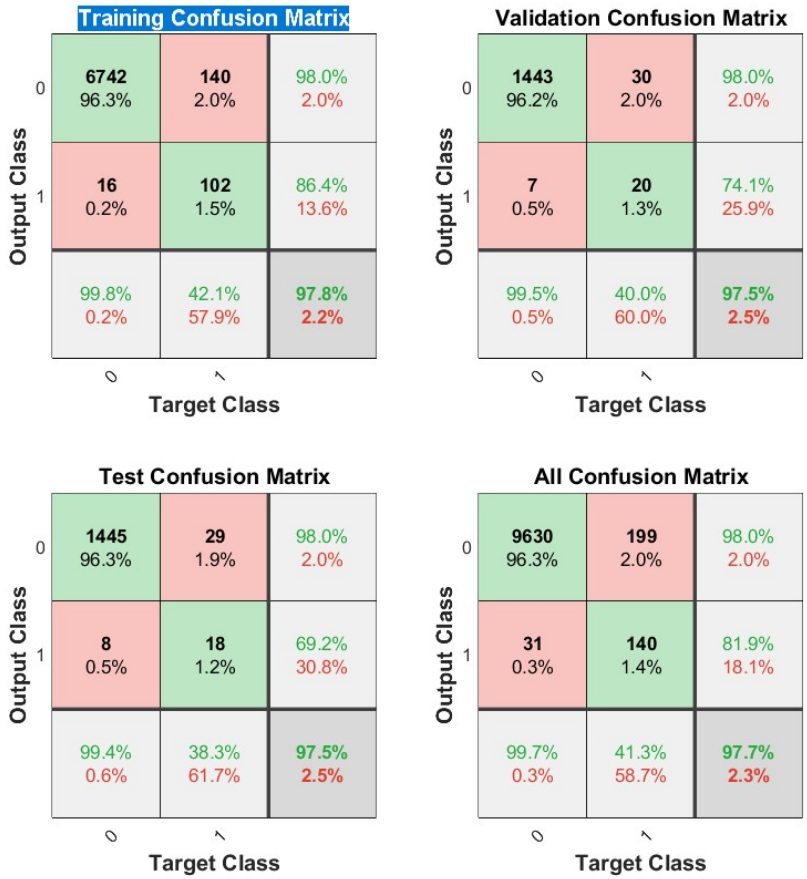

FIGURE 4.1 – Matrices de Confusion pour les ensembles d'entraînement, de validation, de test et pour la totalité des données.

Pour une analyse plus fine, nous devons examiner les performances pour chaque classe :

- **Classe 0 (Fonctionnement normal) :** Le modèle est extrêmement fiable pour identifier les situations normales. Il atteint une **précision de 98.0%** (les prédictions de normalité sont correctes à 98%) et un **rappel de 99.4%** (il identifie 99.4% de tous les cas normaux).
- **Classe 1 (Panne) :** La détection des pannes est une tâche plus difficile, notamment en raison du déséquilibre des classes (les pannes sont rares).
  - La **précision est de 69.2%**. Cela signifie que lorsque le modèle prédit une panne, cette prédiction est correcte dans 69.2% des cas.
  - Le **rappel (ou sensibilité) est de 38.3%**. Cela indique que le modèle parvient à détecter 38.3% de l'ensemble des pannes réelles.

Le rappel pour la classe "Panne" est un axe d'amélioration potentiel. Bien que le modèle soit globalement précis, il ne détecte pas encore une majorité des pannes (un peu plus d'une sur trois). Selon le contexte industriel, il pourrait être préférable d'améliorer ce rappel, même au risque de créer plus de fausses alertes (baisse de la précision), afin de ne manquer aucune défaillance critique.

4.2 Analyse Approfondie des Courbes

Les graphiques générés par MATLAB nous offrent une vision plus détaillée du comportement du modèle.

4.2.1 Courbe ROC (Receiver Operating Characteristic)

La courbe ROC illustre le compromis entre le taux de vrais positifs (rappel) et le taux de faux positifs. Une courbe proche du coin supérieur gauche indique une excellente performance.

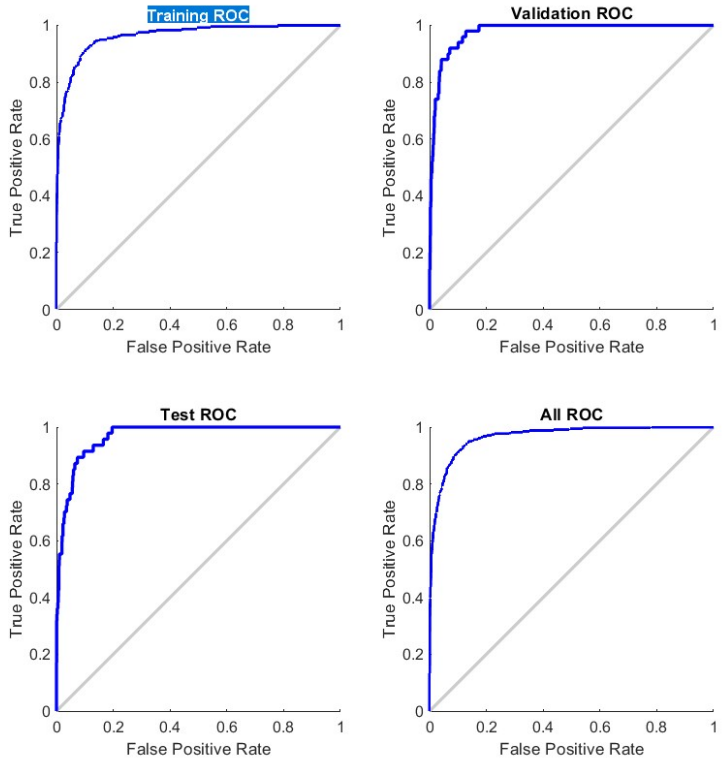

FIGURE 4.2 – Courbes ROC pour les différents ensembles de données.

Les courbes pour les ensembles de test et de validation sont très bonnes, confirmant la capacité du modèle à discriminer les classes.

4.2.2 Performance de Validation

L'évolution de l'erreur (Cross-Entropy) au fil des époques montre que le modèle a convergé efficacement. La meilleure performance de validation a été atteinte à l'époque 37, après quoi l'entraînement s'est arrêté pour éviter le sur-apprentissage.

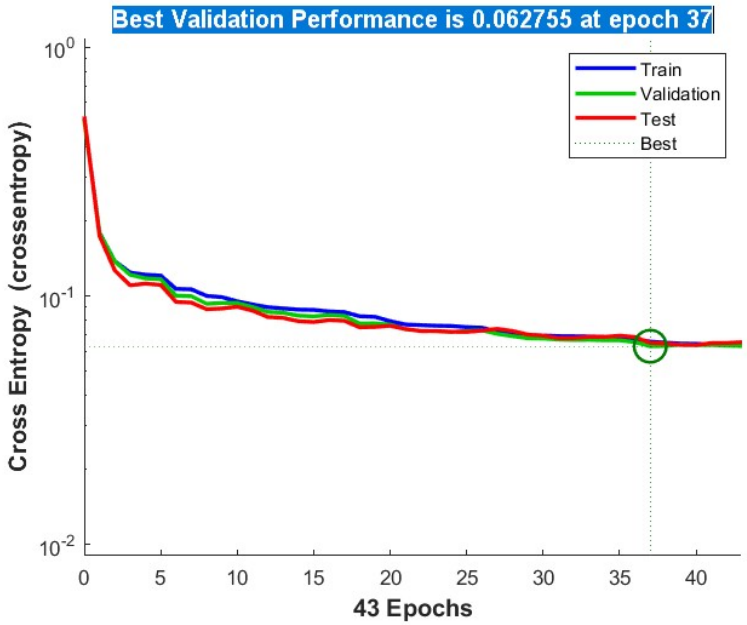

FIGURE 4.3 – Performance de validation (Cross-Entropy) en fonction des époques.

4.2.3 Histogramme d'Erreur

L'histogramme des erreurs montre que la majorité des erreurs de prédiction sont très proches de zéro, ce qui est un signe de bonne performance.

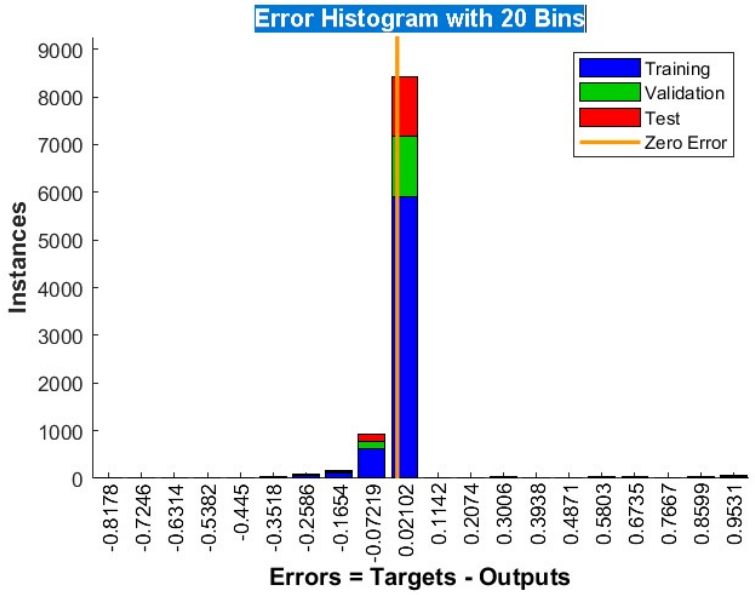

FIGURE 4.4 – Histogramme des erreurs de prédiction.

4.2.4 État de l'Entraînement

Ce graphique confirme que l'entraînement s'est arrêté après 6 échecs consécutifs d'amélioration sur l'ensemble de validation, ce qui est une bonne pratique pour éviter le sur-apprentissage.

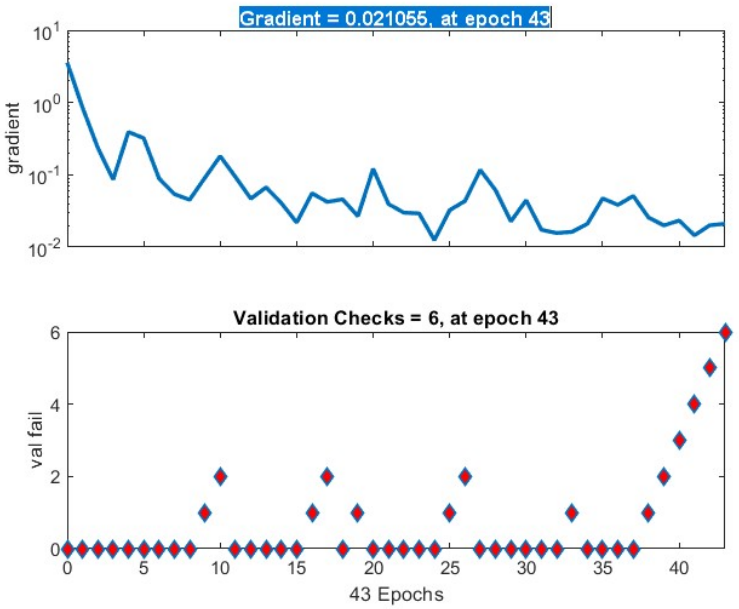

FIGURE 4.5 – État de l'entraînement (gradient et contrôle de validation).

Chapitre 5

Conclusion

Ce projet a permis de démontrer l'efficacité de l'IA pour la maintenance prédictive. L'utilisation de MATLAB et de sa Neural Network Toolbox a facilité la mise en œuvre rapide et efficace d'un réseau de neurones performant pour la détection de pannes.