Projet : Conception et Mise en Œuvre de Réseaux de Neurones Artificiels pour le Contrôle d'un Robot Autonome

Othmane EL HOUDAIGUI
Mohamed ES-SALHY

Arts et Métiers ParisTech (ENSAM campus de Metz)

Janvier 2026

Abstract

Ce rapport présente la conception, l'implémentation et l'analyse de deux réseaux de neurones artificiels destinés au contrôle d'un robot mobile autonome. Le premier réseau (Modèle A) effectue une régression pour réduire les données de 24 capteurs ultrason vers 4 valeurs agrégées. Le second réseau (Modèle B) réalise une classification pour déterminer la commande de pilotage du robot. L'intégration des deux réseaux en série est ensuite évaluée pour valider le système complet. Les résultats montrent un taux de réussite global de **84%** pour le système intégré, avec une excellente qualité de régression (R^2 moyen de 0.954) et une accuracy de classification de 98.1% sur données exactes.

Contents

1 Introduction **3**

2 Description des Données **3**
2.1 Jeux de données disponibles 3
2.2 Prétraitement des données 4
2.3 Répartition des données 4

3 Modèle A : Réseau de Régression **4**
3.1 Objectif 4
3.2 Étude préliminaire : Détermination des hyperparamètres 5
3.3 Architecture et plan d'expériences 6
3.4 Résultats du plan d'expériences 6
3.5 Analyse de Pareto : Performance vs Complexité 6
3.6 Configuration retenue : [40, 10] 7
3.6.1 Qualité de la régression : Graphiques R^2 7
3.7 Résumé du Modèle A 9

4 Modèle B : Réseau de Classification **9**
4.1 Objectif . . . 9
4.2 Préparation des données . . . 9
4.2.1 Division avec stratification . . . 9
4.2.2 Encodage des sorties . . . 9
4.3 Plan d'expériences . . . 9
4.4 Configuration retenue : [20, 15] . . . 10
4.5 Analyse SHAP . . . 11
4.6 Résumé du Modèle B . . . 11

5 Système Complet : Test d'Intégration **11**
5.1 Architecture et résultats . . . 11
5.2 Analyse de la dégradation . . . 11
5.3 Analyse des erreurs . . . 12
5.4 Métriques par classe . . . 12

6 Discussion et Perspectives **13**
6.1 Synthèse . . . 13
6.2 Pistes d'amélioration . . . 13

7 Conclusion **13**

A Annexe : Code Python **13**
A.1 Construction du Modèle A . . . 13
A.2 Construction du Modèle B . . . 14
A.3 Test du système complet . . . 15

1 Introduction

Dans le cadre du module dédié aux réseaux de neurones artificiels, ce projet vise à concevoir et analyser des architectures de réseaux capables de traiter des données issues de capteurs ultrason embarqués sur un robot mobile autonome.

L'objectif global est de permettre au robot de longer un mur dans le sens des aiguilles d'une montre de manière fiable, en exploitant les informations fournies par ses 24 capteurs ultrason, tout en respectant des contraintes de performance, de robustesse et de complexité du modèle.

Le projet est structuré autour de deux réseaux de neurones distincts :

- Un premier réseau de **régression** (Modèle A), chargé de réduire l'information issue de 24 capteurs vers une représentation plus compacte à 4 capteurs agrégés (avant, arrière, droit, gauche);
- Un second réseau de **classification** (Modèle B), chargé de décider la commande de pilotage du robot à partir de ces 4 valeurs agrégées.

Enfin, le système complet comprenant les deux réseaux placés en série est évalué pour vérifier son bon fonctionnement opérationnel.

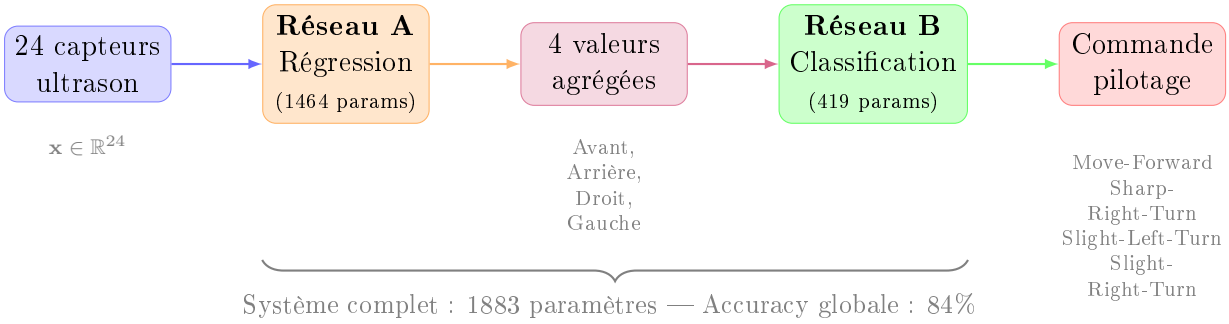

Figure 1: Architecture du système de contrôle robotique : le Réseau A réduit les 24 capteurs en 4 valeurs agrégées, puis le Réseau B détermine la commande de pilotage

Les quatre commandes de sortie possibles sont :

- `Move-Forward` : avancer tout droit
- `Sharp-Right-Turn` : virage serré à droite
- `Slight-Left-Turn` : léger virage à gauche
- `Slight-Right-Turn` : léger virage à droite

Intérêt de l'approche modulaire : La décomposition en deux réseaux distincts présente plusieurs avantages. D'une part, elle permet une interprétabilité accrue grâce aux valeurs intermédiaires physiquement significatives. D'autre part, elle facilite le diagnostic et la maintenance du système en isolant les sources d'erreur potentielles.

2 Description des Données

2.1 Jeux de données disponibles

Les données ont été collectées au cours d'environ 5456 expériences de navigation du robot et sont réparties dans deux fichiers CSV :

- **sensor_readings_24.csv** : contient les lectures des 24 capteurs ultrason disposés circulairement autour du robot, ainsi que la commande associée (25 colonnes);
- **sensor_readings_4.csv** : contient les données agrégées sous forme de 4 capteurs (avant, arrière, droit, gauche) plus la commande (5 colonnes).

2.2 Prétraitement des données

Le prétraitement a consisté en plusieurs étapes :

1. **Séparation des entrées et sorties** : extraction des 24 (resp. 4) capteurs comme variables d'entrée et de la commande comme variable cible;
2. **Division du jeu de données** : répartition différente selon le modèle (cf. Tableau 1);
3. **Normalisation** : application d'un `StandardScaler` sur les entrées pour centrer-réduire les données, essentiel pour la convergence des réseaux de neurones.

2.3 Répartition des données

Modèle	Ensemble	Proportion	Nb échantillons	Stratification
Modèle A (Régression)	Entraînement	70%	3819	Non
	Validation	15%	818	Non
	Test	15%	819	Non
Modèle B (Classification)	Entraînement	80%	4365	Oui
	Test	20%	1091	Oui
Total		100%	5456	—

Table 1: Répartition des données pour chaque modèle

Justification des choix :

- **Modèle A (70-15-15)** : Le split en trois parties permet d'avoir un ensemble de validation séparé pour le suivi de l'entraînement (early stopping, sélection de modèle) et un ensemble de test indépendant pour l'évaluation finale non biaisée.
- **Modèle B (80-20 avec stratification)** : La stratification garantit que la distribution des 4 classes est conservée entre Train et Test, ce qui est crucial car les classes sont déséquilibrées (cf. Section 4.2).

3 Modèle A : Réseau de Régression

3.1 Objectif

Le Modèle A a pour but de réduire la dimensionnalité des données en passant de 24 capteurs à 4 valeurs agrégées. Il s'agit d'un problème de **régression multivariée** où le réseau doit apprendre la relation non-linéaire entre les 24 lectures brutes et les 4 mesures agrégées correspondantes.

Cette réduction dimensionnelle permet de :

1. Diminuer la complexité du problème de classification en aval;
2. Produire des représentations interprétables (avant, arrière, droit, gauche);
3. Réduire le bruit inhérent aux capteurs individuels.

3.2 Étude préliminaire : Détermination des hyperparamètres

Avant de lancer le plan d'expériences complet, une étude préliminaire a été réalisée avec un réseau simple pour déterminer les hyperparamètres d'entraînement optimaux (nombre d'époques, learning rate, etc.).

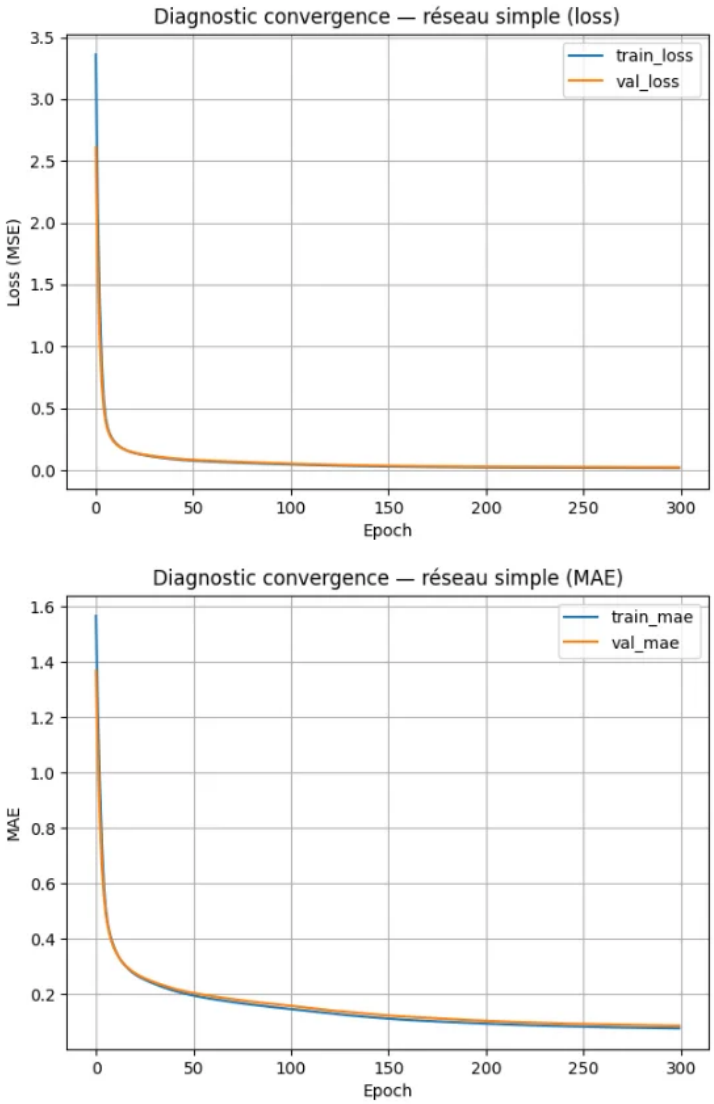

Figure 2: Diagnostic de convergence sur un réseau simple : cette étude préliminaire permet de calibrer les hyperparamètres (nombre d'époques, learning rate) pour le plan d'expériences.

Conclusions de l'étude préliminaire :

- **Convergence rapide** : La majorité de l'apprentissage se fait dans les 50 premières époques.
- **Stabilisation** : Les métriques se stabilisent autour de l'époque 150-200.

- **Absence de sur-apprentissage** : Les courbes Train et Validation restent très proches.
- **Learning rate adapté** : Pas d'oscillations, confirmant que $lr = 10^{-3}$ est approprié.

Hyperparamètres retenus : 200 époques (avec early stopping, patience = 20), $lr = 10^{-3}$, batch size = 128.

3.3 Architecture et plan d'expériences

Conformément aux consignes du projet, un plan d'expériences a été mis en place :

- **Nombre de couches cachées** : 1 ou 2 couches
- **Nombre de neurones par couche** : 10, 15, 20, 25, 30, 40
- **Distribution triangulaire** : nombre décroissant de neurones au fil des couches

Hyperparamètres fixes : Activation ReLU (couches cachées) + linéaire (sortie), optimiseur Adam ($lr = 10^{-3}$), loss MSE, 200 époques avec early stopping (patience = 20), batch size = 128, 3 répétitions par configuration (seeds différentes).

3.4 Résultats du plan d'expériences

La figure 3 présente la comparaison des configurations testées sous forme de boxplots (3 seeds par configuration).

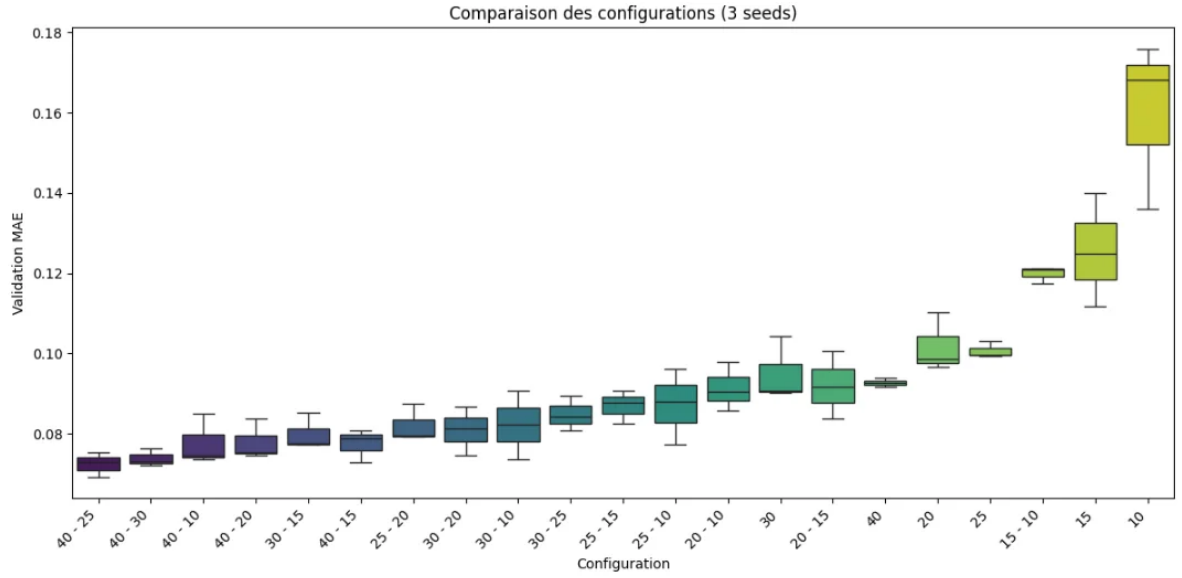

Figure 3: Comparaison des configurations du Modèle A : MAE de validation pour chaque architecture (3 seeds)

Observations : Les configurations à 2 couches (40-25, 40-30, 40-10) surpassent celles à 1 couche. La variabilité diminue avec la complexité du réseau. Les meilleures configurations atteignent une MAE d'environ 0.07-0.08.

3.5 Analyse de Pareto : Performance vs Complexité

Le front de Pareto (courbe rouge) identifie les configurations *non-dominées*. La configuration **[40, 10]** se situe au "coude" de la courbe, offrant une MAE d'environ 0.08 pour ∼1500

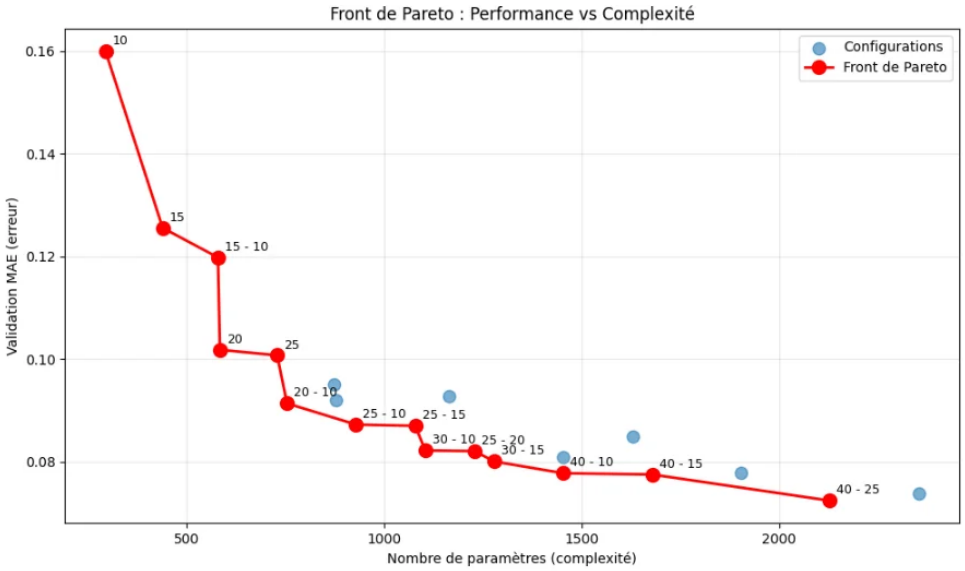

Figure 4: Front de Pareto : compromis entre performance (MAE) et complexité (nombre de paramètres)

paramètres — soit 95% de la performance maximale avec seulement 60% des paramètres. Au-delà de 1500 paramètres, les gains deviennent marginaux.

3.6 Configuration retenue : [40, 10]

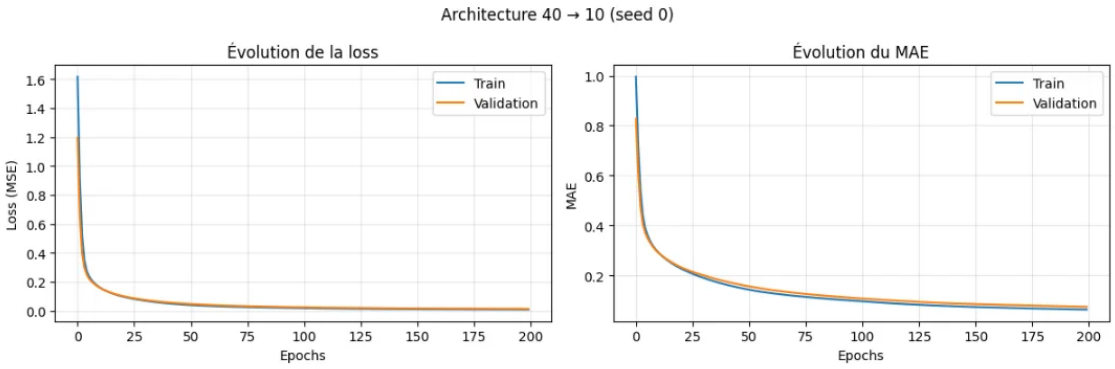

Figure 5: Évolution de la Loss (MSE) et de la MAE pendant l'entraînement pour l'architecture [40, 10]

Diagnostic : Convergence rapide (50 premières époques), absence de sur-apprentissage (courbes Train/Validation proches, écart $< 5\%$), stabilisation à MAE ≈ 0.08.

3.6.1 Qualité de la régression : Graphiques R^2

Analyse détaillée par capteur :

Interprétation physique : Le capteur **Gauche** ($R^2 = 0.9953$) est le mieux prédit car le robot longe le mur à droite (mesures stables côté libre). Le capteur **Arrière** ($R^2 = 0.9132$) est le moins bien prédit car il capture des informations moins structurées.

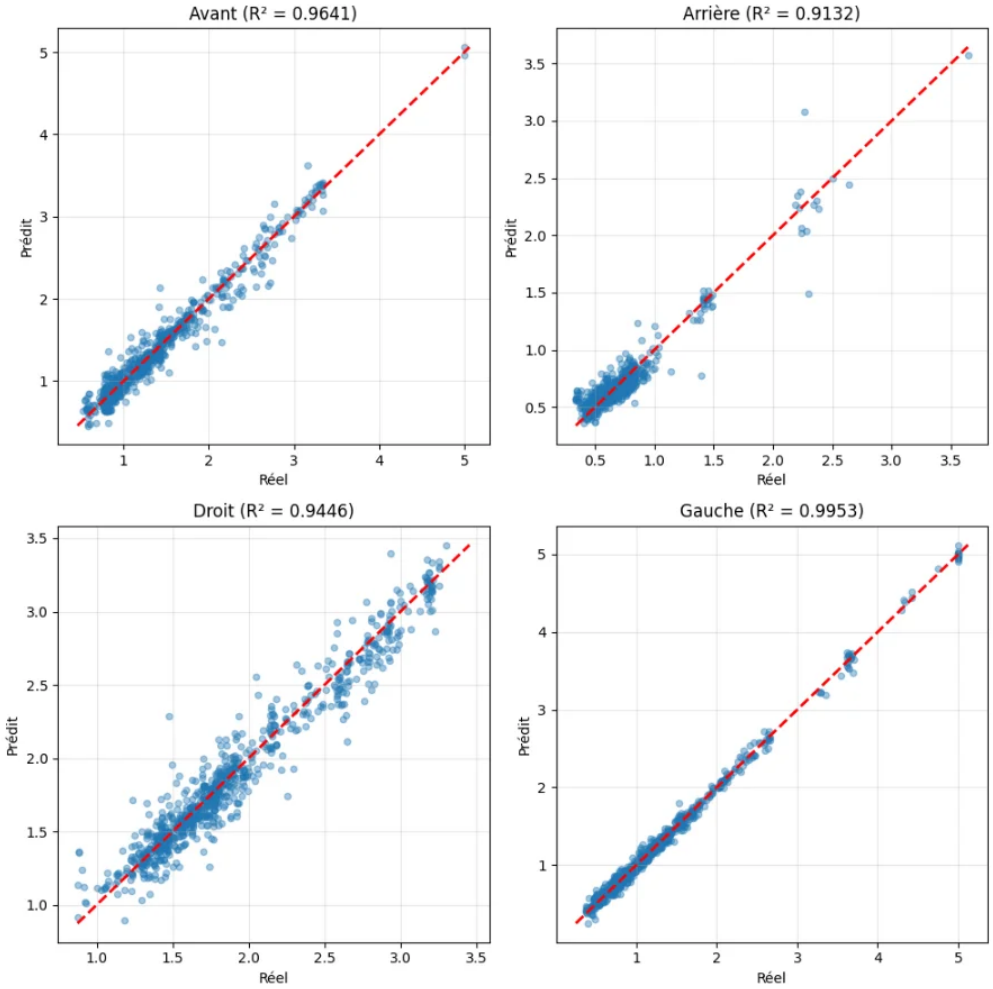

Figure 6: Qualité de la régression : valeurs prédites vs réelles avec coefficients R^2

Capteur	R^2	**Qualité**	**Interprétation**
Gauche	0.9953	Excellente	Relation quasi-linéaire, peu de dispersion
Avant	0.9641	Très bonne	Légère dispersion aux valeurs élevées
Droit	0.9446	Très bonne	Dispersion modérée, plus de variabilité
Arrière	0.9132	Bonne	Plus grande variabilité, outliers présents
Moyenne	**0.9543**	**Très bonne**	

Table 2: Synthèse des coefficients de détermination par capteur

Paramètre	**Valeur**
Architecture	$24 \to 40$ (ReLU) $\to 10$ (ReLU) $\to 4$ (linéaire)
Nombre de paramètres	1 464
MAE validation	0.08
MSE validation	0.02
R^2 moyen	0.954
Temps d'entraînement	$\sim$30 secondes (GPU)

Table 3: Caractéristiques du Modèle A retenu

3.7 Résumé du Modèle A

4 Modèle B : Réseau de Classification

4.1 Objectif

Le Modèle B est un réseau de classification qui détermine la commande de pilotage du robot à partir des 4 valeurs agrégées. Il s'agit d'une **classification multiclasse** avec 4 classes mutuellement exclusives.

4.2 Préparation des données

4.2.1 Division avec stratification

Pour le Modèle B, une attention particulière est portée à la **stratification** afin de conserver la distribution des classes dans les ensembles d'entraînement et de test. Ceci est crucial car les classes sont déséquilibrées.

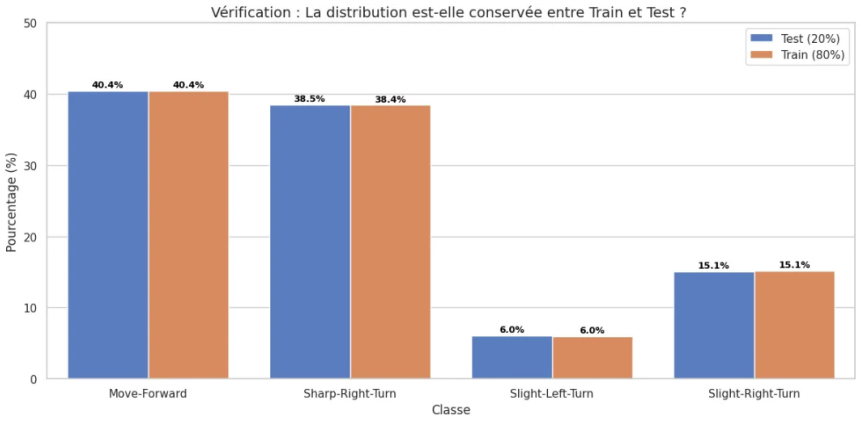

Figure 7: Vérification de la conservation de la distribution des classes entre Train (80%) et Test (20%) pour le Modèle B

Analyse de la distribution des classes :

- `Move-Forward` : 40.4% — commande majoritaire (navigation en ligne droite)
- `Sharp-Right-Turn` : 38.4-38.5% — virages dans les coins du parcours
- `Slight-Right-Turn` : 15.1% — corrections fines de trajectoire
- `Slight-Left-Turn` : 6.0% — classe minoritaire (éloignement du mur)

Impact du déséquilibre : La classe `Slight-Left-Turn` ne représente que 6% des données, ce qui pourrait impacter les performances sur cette classe minoritaire.

4.2.2 Encodage des sorties

Les commandes textuelles ont été transformées en encodage one-hot pour permettre l'utilisation de la fonction softmax et de la cross-entropy :

4.3 Plan d'expériences

- **Couches cachées** : 1, 2, 3 ou 4 couches
- **Neurones par couche** : 1, 2, 4, 6, 8, 10, 15, 20

Commande	Classe 0	Classe 1	Classe 2	Classe 3
Move-Forward	1	0	0	0
Sharp-Right-Turn	0	1	0	0
Slight-Left-Turn	0	0	1	0
Slight-Right-Turn	0	0	0	1

Table 4: Encodage one-hot des commandes

- **Hyperparamètres** : ReLU + Softmax, Adam ($lr = 10^{-3}$), Cross-Entropy, 200 époques (patience = 30), 3 seeds

4.4 Configuration retenue : [20, 15]

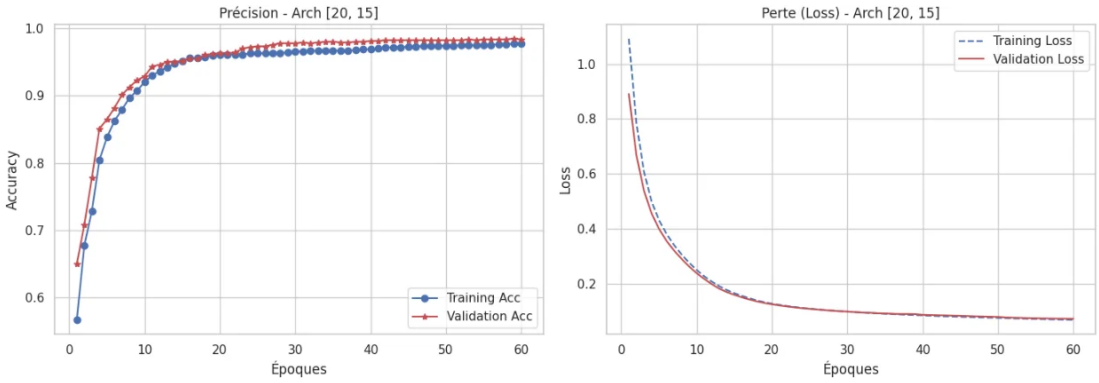

Figure 8: Évolution de l'Accuracy et de la Loss pour l'architecture [20, 15]

Analyse : Convergence très rapide (90% dès l'époque 10, 98% après 30 époques), excellente généralisation (écart Train/Val $< 1\%$), loss stable autour de 0.05.

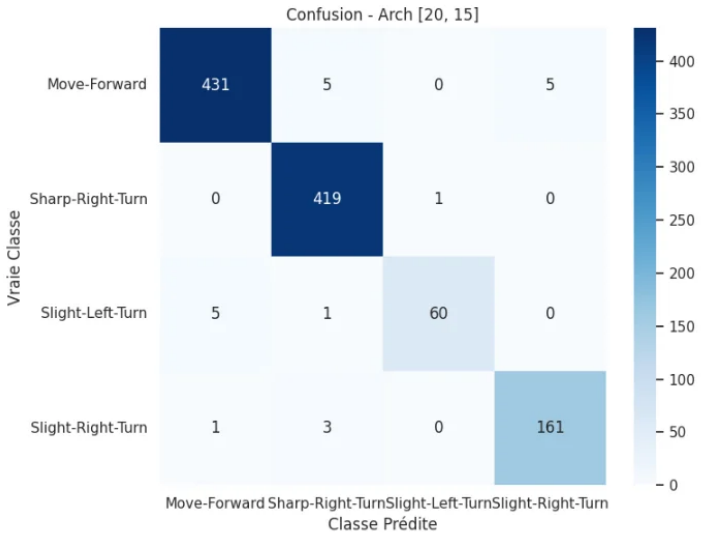

Figure 9: Matrice de confusion du Modèle B (architecture [20, 15])

Performance par classe : Move-Forward 97.7%, Sharp-Right-Turn 99.8%, Slight-Left-Turn 90.9%, Slight-Right-Turn 97.6%. **Accuracy globale : 98.1%**. La classe Slight-Left-Turn a le taux le plus bas en raison de sa faible représentation (6%).

4.5 Analyse SHAP

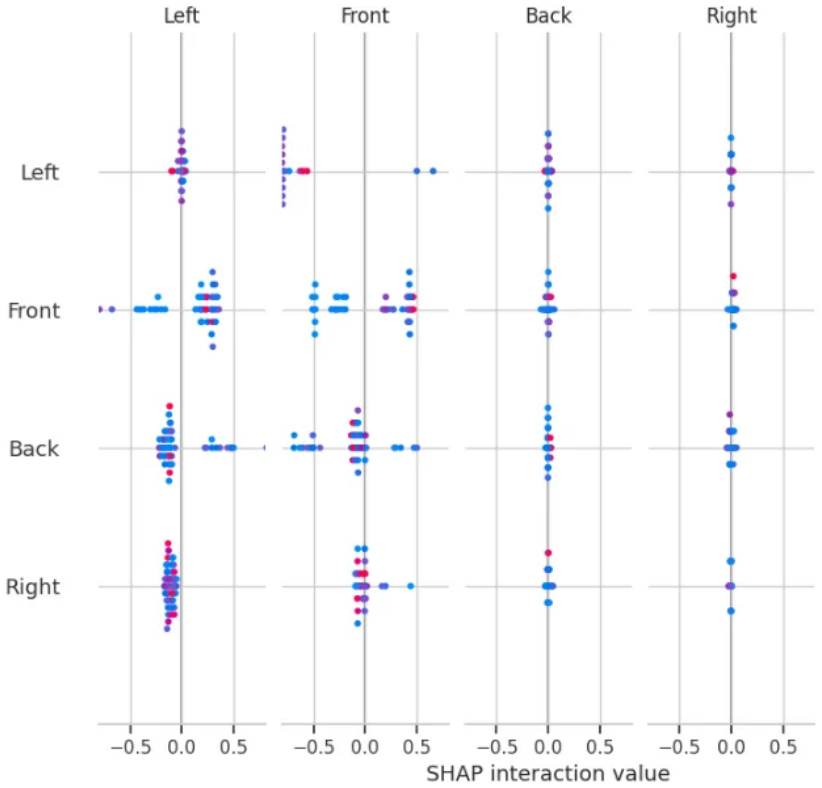

Figure 10: Matrice des interactions SHAP entre les 4 capteurs agrégés

Interprétation : Le capteur **Right** présente la plus grande dispersion SHAP — c'est le plus influent, cohérent avec la mission de longer un mur à droite. Le capteur **Front** est second en importance (détection d'obstacles). L'interaction **Right-Front** est cruciale pour les décisions de virage. Les capteurs Left et Back ont des effets plus faibles. Cette analyse confirme que le modèle a appris une représentation **physiquement cohérente**.

4.6 Résumé du Modèle B

Architecture : $4 \to 20 \to 15 \to 4$ | Paramètres : 419 | Accuracy test : **98.1%**

5 Système Complet : Test d'Intégration

5.1 Architecture et résultats

Le système enchaîne : Normalisation → Modèle A (régression) → Modèle B (classification) → argmax. Le Modèle B reçoit les valeurs **prédites** par A, donc toute erreur de A se propage.

5.2 Analyse de la dégradation

La perte de 14.1 points s'explique par la **propagation des erreurs** : la MAE de 0.08 du Modèle A décale les entrées du Modèle B dans des zones de décision incorrectes, particulièrement pour les classes proches.

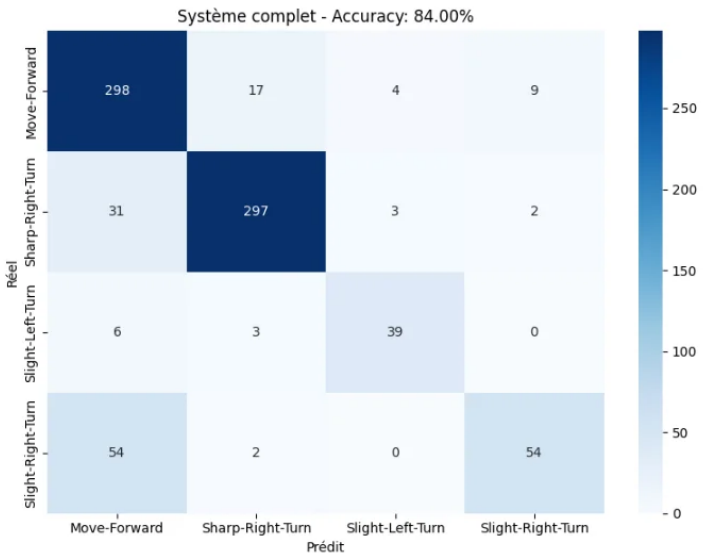

Figure 11: Matrice de confusion du système complet — Accuracy : **84.00%**

Configuration	**Accuracy**	**Δ**
Modèle B seul (données exactes)	98.1%	—
Système complet (A + B)	84.0%	-14.1%

Table 5: Impact de la cascade sur les performances

5.3 Analyse des erreurs

Erreurs principales : Slight-Right → Move-Forward (54 cas, 33%), Sharp-Right → Move-Forward (31 cas, 19%). Ces erreurs (53% du total) correspondent à des virages manqués — gravité modérée, corrigeable au cycle suivant. Les confusions critiques (directions opposées) représentent moins de 5%.

5.4 Métriques par classe

Classe	**Précision**	**Rappel**	**F1-Score**	**Diagnostic**
Move-Forward	0.77	0.91	0.83	Sur-prédit (biais)
Sharp-Right-Turn	0.93	0.89	0.91	Équilibré
Slight-Left-Turn	0.85	0.81	0.83	Équilibré
Slight-Right-Turn	0.83	**0.49**	0.62	**Sous-détecté**
Moyenne pondérée	**0.84**	**0.84**	**0.82**	

Table 6: Métriques de classification par classe pour le système complet

Point critique : La classe `Slight-Right-Turn` a un rappel de seulement **0.49**, signifiant que plus de la moitié des virages légers à droite sont manqués. Ce problème mérite une attention particulière dans les améliorations futures.

6 Discussion et Perspectives

6.1 Synthèse

Points forts : Excellente réduction dimensionnelle ($R^2 = 0.954$), classification quasi-parfaite sur données exactes (98.1%), architecture légère (1883 paramètres), interprétabilité des valeurs intermédiaires, cohérence SHAP.

Points d'amélioration : Propagation d'erreur A→B, classe Slight-Right-Turn sous-détectée (rappel 49%), capteur Arrière moins bien prédit ($R^2 = 0.91$).

6.2 Pistes d'amélioration

1. **Entraînement end-to-end** : Connecter A et B, entraîner conjointement avec loss combinée
2. **Rééquilibrage** : SMOTE ou class weights pour Slight-Left-Turn (6%)
3. **Architecture directe** : Réseau unique 24→4 sans compression intermédiaire
4. **Régularisation** : Dropout (20-30%) et/ou L2 sur les poids
5. **Seuillage adaptatif** : Seuils calibrés par classe au lieu de argmax
6. **Ensemble** : Bagging + vote majoritaire pour réduire la variance

7 Conclusion

Ce projet a permis de concevoir, implémenter et évaluer un système de contrôle robotique basé sur deux réseaux de neurones artificiels en cascade.

Le Modèle A de régression accomplit efficacement sa mission de réduction dimensionnelle, transformant les 24 capteurs ultrason en 4 valeurs agrégées interprétables. Avec un R^2 moyen de 0.954 et seulement 1464 paramètres, il offre un excellent compromis entre précision et légèreté. L'analyse du front de Pareto a guidé le choix de l'architecture optimale [40, 10].

Le Modèle B de classification démontre une capacité remarquable à déterminer la commande de pilotage, atteignant 98.1% d'accuracy sur les données agrégées exactes. L'analyse SHAP valide la cohérence physique des décisions : le capteur droit domine la prise de décision, ce qui correspond à la tâche de longer un mur à droite.

Le système intégré atteint un taux de réussite de 84%, acceptable pour une première implémentation. La dégradation de 14 points par rapport au Modèle B seul s'explique par la propagation des erreurs de régression. Les erreurs observées sont majoritairement des confusions "bénignes" entre `Move-Forward` et `Slight-Right-Turn`, suggérant une navigation légèrement plus rectiligne que prévu.

Ce travail démontre la faisabilité d'une approche modulaire pour le pilotage autonome, avec une architecture totale de moins de 2000 paramètres, compatible avec un déploiement embarqué. Les perspectives d'amélioration incluent l'entraînement end-to-end, le rééquilibrage des classes, et l'exploration d'architectures alternatives.

A Annexe : Code Python

A.1 Construction du Modèle A

Listing 1: Fonction de construction du Modèle A (régression)

```python
def build_model(layers_list, input_dim=24, output_dim=4, lr=1e-3):
    """
    Construit un reseau de regression multi-sorties.
    Args:
        layers_list: Liste du nombre de neurones par couche cachee
        input_dim: Dimension d'entree (24 capteurs)
        output_dim: Dimension de sortie (4 valeurs agregees)
        lr: Learning rate pour Adam
    """
    model = keras.Sequential()
    model.add(layers.Input(shape=(input_dim,)))
    for n in layers_list:
        model.add(layers.Dense(n, activation="relu"))
    model.add(layers.Dense(output_dim, activation="linear"))

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=lr),
        loss="mse",
        metrics=[keras.metrics.MeanAbsoluteError(name="mae")]
    )
    return model
```

A.2 Construction du Modèle B

Listing 2: Fonction de construction du Modèle B (classification)

```python
def build_classif(layers_list, input_dim=4, num_classes=4, lr=1e-3)
    :
    """
    Construit un reseau de classification multiclasse.
    Args:
        layers_list: Liste du nombre de neurones par couche cachee
        input_dim: Dimension d'entree (4 valeurs agregees)
        num_classes: Nombre de classes (4 commandes)
        lr: Learning rate pour Adam
    """
    model = keras.Sequential()
    model.add(layers.Input(shape=(input_dim,)))
    for n in layers_list:
        model.add(layers.Dense(n, activation="relu"))
    model.add(layers.Dense(num_classes, activation="softmax"))

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=lr),
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )
    return model
```

A.3 Test du système complet

Listing 3: Évaluation du système intégré (pipeline A+B)

```
# === Prediction en cascade ===
# Etape 1: Regression (24 capteurs -> 4 valeurs)
y_pred_4sensors = model_A.predict(X_test_scaled, verbose=0)

# Etape 2: Classification (4 valeurs -> probabilites)
proba = model_B.predict(y_pred_4sensors, verbose=0)

# Etape 3: Decision (argmax)
y_pred_idx = np.argmax(proba, axis=1)
y_pred_commands = label_encoder.inverse_transform(y_pred_idx)

# === Evaluation ===
accuracy = np.mean(y_pred_commands == y_test_true)
print(f"Taux de reussite systeme complet: {accuracy*100:.2f}%")

# Matrice de confusion
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test_true, y_pred_commands)
print(classification_report(y_test_true, y_pred_commands))
```