Université Lumière Lyon 2

Institut de la communication

Deep Learning : Étude progressive des réseaux de neurones sur le jeu de données HAR

Réalisé par : Damba KONE
Enseignant : Walid Bechkit

23 Novembre 2025

1 Introduction

Ce projet vise à classifier six activités humaines à partir du jeu de données HAR, constitué de mesures d'inertie d'un smartphone. Nous étudions plusieurs architectures de réseaux de neurones multicouches (MLP) afin d'analyser l'impact de la profondeur, des fonctions d'activation, des optimiseurs et des techniques de régularisation sur la performance et la généralisation.

La méthodologie suit quatre étapes principales : exploration des données, mise en place d'un modèle simple (baseline), extension vers des architectures plus profondes, et évaluation finale avec interprétabilité via t-SNE et UMAP. L'objectif est d'identifier des configurations performantes tout en limitant le sur-apprentissage.

2 Analyse Exploratoire des Données

2.1 Présentation du jeu de données

Le jeu de données utilisé provient du projet *Human Activity Recognition Using Smartphones*. Il contient des mesures issues de capteurs (accéléromètres et gyroscopes) pour différentes activités humaines. Chaque observation est décrite par plusieurs centaines de variables continues, dérivées de signaux temporels et fréquentiels.

La variable cible `Activity` comporte 6 classes :

- `WALKING`
- `WALKING_UPSTAIRS`
- `WALKING_DOWNSTAIRS`
- `SITTING`
- `STANDING`
- `LAYING`

Les colonnes incluent des caractéristiques telles que : `tBodyAcc-mean()-X`, `tBodyAcc-mean()-Y`, `tBodyAcc-std()-X`, etc.

2.2 Distribution des classes

La Figure 1 illustre la répartition des activités dans le jeu d'entraînement. Les classes sont relativement équilibrées, bien que `LAYING` et `STANDING` soient légèrement plus fréquentes.

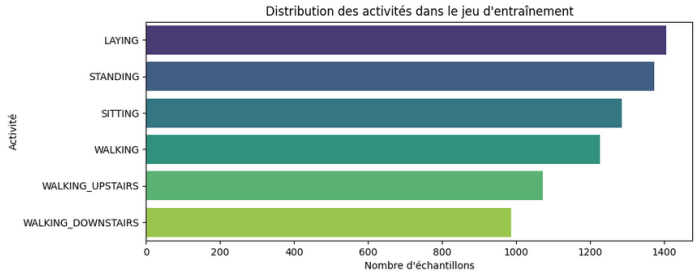


FIGURE 1 – Distribution des activités dans le jeu d'entraînement

2.3 Statistiques descriptives

Le Tableau 1 présente quelques statistiques pour des variables représentatives. Les valeurs sont centrées autour de zéro, avec des amplitudes allant de -1 à 1, ce qui confirme la nécessité d'une normalisation avant l'apprentissage.

TABLE 1 – Statistiques descriptives pour quelques variables

Variable	Moyenne	Écart-type	Min	Max
tBodyAcc-mean()-X	0.274	0.070	-1.000	1.000
tBodyAcc-mean()-Y	-0.018	0.041	-1.000	1.000
tBodyAcc-mean()-Z	-0.109	0.057	-1.000	1.000
tBodyAcc-std()-X	-0.605	0.449	-1.000	1.000
tBodyAcc-std()-Y	-0.511	0.503	-0.999	0.916
tBodyAcc-std()-Z	-0.605	0.419	-1.000	1.000

2.4 Corrélations entre variables

La Figure 2 montre la matrice de corrélation pour un sous-ensemble de variables. On observe une forte corrélation entre les composantes des écarts-types (`std`), ce qui laisse entendre une redondance importante dans les données.

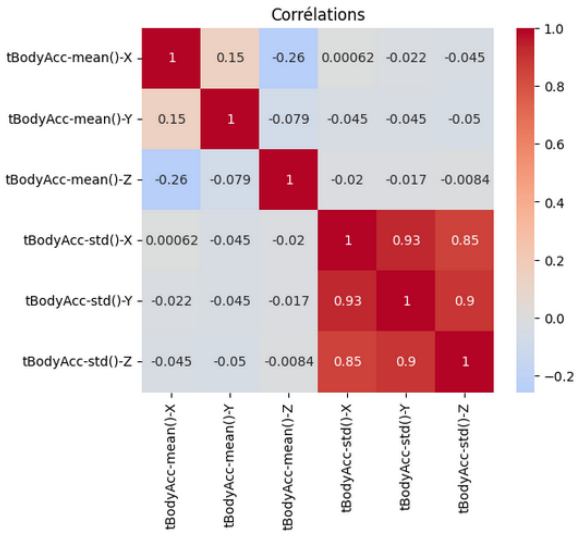


FIGURE 2 – Corrélations entre quelques variables

2.5 Distribution et valeurs extrêmes

Les Figures 3 et 4 illustrent la distribution des variables et la présence de valeurs extrêmes. Les boxplots révèlent des outliers, mais ceux-ci semblent cohérents.

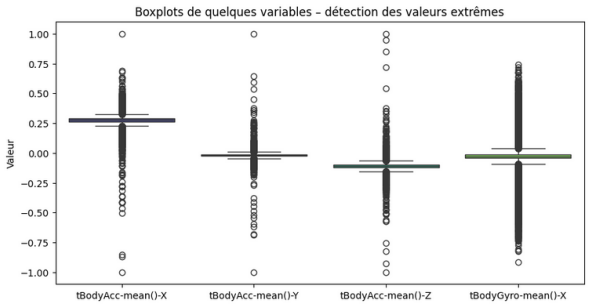


FIGURE 3 – Boxplots de quelques variables

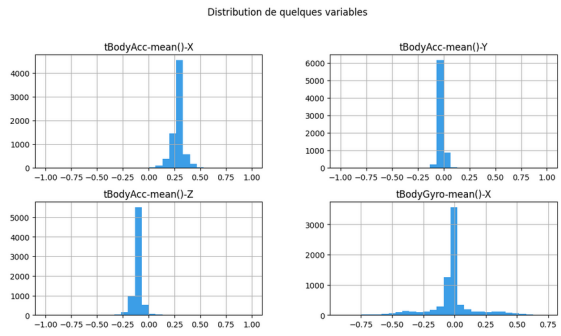


FIGURE 4 – Distribution de quelques variables

2.6 Qualité des données et prétraitement

L'analyse exploratoire montre que :

- Le jeu de données ne contient pas de valeurs manquantes.
- Une normalisation des variables est nécessaire pour homogénéiser les échelles.
- La cible `Activity` doit être encodée via un encodage numérique.
- La forte multicolinéarité suggère qu'une réduction de dimension (PCA) ou une sélection de variables pourrait être envisagée.

Conclusion : Les données sont globalement propres et prêtes pour l'apprentissage après normalisation et encodage de la cible.

3 Première architecture de réseau : Modèle à une seule couche cachée

3.1 Architecture

Nous utilisons un MLP comportant :

- une couche cachée de 64 neurones,
- une activation parmi ReLU, Sigmoid, Tanh,
- une sortie à 6 classes,
- une perte *CrossEntropyLoss*.

3.2 Expérimentations

Neuf combinaisons ont été testées (3 activations $\times$ 3 optimiseurs : SGD, SGD Momentum, Adam). Les courbes loss/accuracy montrent une convergence rapide mais un sur-apprentissage marqué, surtout avec ReLU/Tanh + Adam.

3.3 Résultats

ReLU et Tanh offrent les meilleures performances (F1-macro ≈ 0.98 avec Adam). Sigmoid converge plus lentement mais généralise légèrement mieux. SGD Momentum offre un bon compromis rapidité/stabilité. Les Figures présentant les courbes d'apprentissage pour l'ensemble des combinaisons activation–optimiseur se trouvent dans le Notebook.

TABLE 2 – Performances des modèles selon la fonction d'activation et l'optimiseur

Activation	Optimiseur	Best Epoch	Train Acc	Validate Acc	F1 Macro	Validate Loss
tanh	adam	30	0.9997	0.9850	0.9854	0.0536
relu	adam	26	0.9973	0.9844	0.9849	0.0542
sigmoid	adam	28	0.9946	0.9830	0.9838	0.0526
tanh	sgd_momentum	19	0.9935	0.9823	0.9828	0.0573
relu	sgd_momentum	26	0.9896	0.9816	0.9822	0.0593
sigmoid	sgd_momentum	27	0.9869	0.9762	0.9769	0.0740
relu	sgd	29	0.9827	0.9721	0.9732	0.0895
tanh	sgd	30	0.9820	0.9721	0.9730	0.0950
sigmoid	sgd	29	0.9418	0.9375	0.9378	0.2412

3.4 Conclusion

Cette première série d'expérimentations met en évidence l'importance du choix de l'optimiseur et de la fonction d'activation dans la performance d'un réseau à une seule couche cachée. Les meilleurs résultats sont obtenus avec les combinaisons `ReLU + Adam` et `ReLU + SGD Momentum`, mais certains modèles montrent déjà des signes de sur-apprentissage, ce qui motive l'introduction de techniques de régularisation dans la partie suivante.

4 Extension de l'architecture : réseaux plus profonds

4.1 Augmentation de la taille de la couche cachée

Nous avons évalué l'impact de l'augmentation de la taille des couches en augmentant la couche cachée de 64 à 128 puis 256 neurones. Les mêmes combinaisons activation–optimiseur que dans la Partie 2 ont été testées. (Voir les figures dans le Notebook)

L'analyse des courbes d'accuracy et de loss montre que tous les modèles atteignent une précision d'entraînement très élevée, mais la précision sur validation reste inférieure, révélant un surapprentissage. Ce phénomène s'accentue lorsque la taille de la couche cachée augmente (64 à 128 neurones).

- `ReLU + Adam / ReLU + SGD Momentum` : convergence rapide, train proche de 1.0, validation autour de 0.94–0.95. Donc il y a un surapprentissage marqué.
- `ReLU + SGD` : convergence plus lente, écart entre train/validation modéré. Donc on a un surapprentissage léger, ce qui indique une bonne généralisation.
- `Tanh + Adam / Tanh + SGD Momentum` : convergence rapide, train autour de 1.0, validation à proximité de 0.94. Il y a également un surapprentissage modéré.
- `Tanh + SGD` : montée progressive, écart faible entre train (autour de 0.97) et validation (vers 0.93). Il y a une généralisation correcte, surapprentissage quasi nul.
- `Sigmoid + Adam` : convergence rapide, écart pas très important entre train/validation. Il y a aussi un surapprentissage modéré.
- `Sigmoid + SGD` : convergence lente, train autour de 0.90, validation entre 0.85–0.90. On n'observe ni de surapprentissage, ni de sous-apprentissage.
- `Sigmoid + SGD Momentum` : train autour de 0.98, validation entre 0.95–0.98. On observe un écart minimal mais un bon équilibre entre apprentissage et généralisation.

`ReLU` et `Tanh` entraînent un surapprentissage modéré, surtout avec Adam ou SGD Momentum. Sigmoid converge plus lentement mais généralise mieux. **Augmenter la taille des couches accentue le surapprentissage**.

4.2 Réseaux à plusieurs couches cachées

Nous avons implémenté des architectures plus profondes comportant 2 et 3 couches cachées.

Plusieurs combinaisons ont été testées :

- architectures à 2 et 3 couches cachées ;
- différentes fonctions d'activation (`ReLU`, `Tanh`, `Sigmoid`) ;
- optimisateurs `SGD`, `SGD Momentum` et `Adam`.

Les modèles profonds en général convergent plus rapidement, mais certains présentent un surapprentissage marqué dès les premières époques.

4.3 Synthèse

L'augmentation de la profondeur et de la taille des couches améliore légèrement les performances, mais accentue également l'écart entre les courbes d'entraînement et de validation.

Résumé du surapprentissage selon les combinaisons :

- Surapprentissage très marqué : `ReLU/Tanh + Adam` ou `SGD Momentum`, surtout avec **3 couches** et $\geq$ **128 neurones**.
- Surapprentissage modéré : `Tanh + Adam` et `Tanh + SGD Momentum`.
- Surapprentissage quasi absent : `SGD simple` et `Sigmoid + SGD Momentum`.
- Instabilité et faible performance : `Sigmoid + SGD`.

5 Régularisation et robustesse

5.1 Méthodologie pour la régularisation

Pour réduire le sur-apprentissage observé sur les modèles profonds, plusieurs techniques de régularisation ont été intégrées : **Dropout** (p = 0.1–0.5), **Weight Decay** (10^{-4}–10^{-3}), **Batch Normalization** après chaque couche cachée et **Early Stopping**.

Un pipeline a permis de combiner ces régularisations avec différentes profondeurs, tailles de couches, fonctions d'activation et optimiseurs. Les performances ont été évaluées via la *loss*, l'*accuracy* et le *F1-macro* sur les ensembles d'entraînement et de validation.

5.2 Effet des régularisations sur le sur-apprentissage

Avant régularisation, les courbes *Train* vs *Validation* divergeaient après 15–20 époques, en particulier pour les combinaisons `ReLU + Adam` et `Tanh + Adam`. Après application des régularisations :

- Les courbes sont beaucoup plus proches, surtout avec **Dropout = 0.3** et **Weight Decay** = 10^{-4} **ou** 10^{-3}.
- **Batch Normalization** a stabilisé les oscillations pour Tanh et Sigmoid.
- **Early Stopping** a limité les fluctuations en fin d'entraînement.

5.3 Impact par configuration

- `Sigmoid + SGD` : reste faible (accuracy < 0.8), même avec régularisation.
- `Tanh + SGD` : nette amélioration, validation accuracy proche de 0.90 avec Dropout + Weight Decay.
- `ReLU + Adam` : bonne amélioration, l'écart entre train/validation réduit.
- `ReLU + SGD Momentum` : amélioration notable, mais moins rapide qu'Adam.
- `Tanh + Adam` : stabilisation, validation accuracy > 0.93 avec régularisation.

5.4 Configurations optimales

Les meilleures combinaisons observées sont :

- `ReLU + Adam + Dropout(0.3) + Weight Decay(`10^{-4}`)` : meilleure combinaison.
- `Tanh + Adam + Dropout(0.3)` : bon compromis pour robustesse.
- `BatchNorm` : bénéfique surtout pour Sigmoid et Tanh (réduit oscillations).

5.5 Remarques importantes

— `Sigmoid + SGD` reste mauvais malgré la régularisation.
— Trop de Dropout ($p = 0.5$) ralentit la convergence sans gain significatif.

5.6 Conclusion

La régularisation a eu un impact positif :

— Moins de sur-apprentissage.
— Courbes plus stables.
— Meilleure généralisation pour ReLU et Tanh avec Adam.

Cependant, certains choix d'activation/optimiseur peuvent être inadaptés selon le type de problème ou de jeu de données.

5.7 Configurations retenues pour la suite

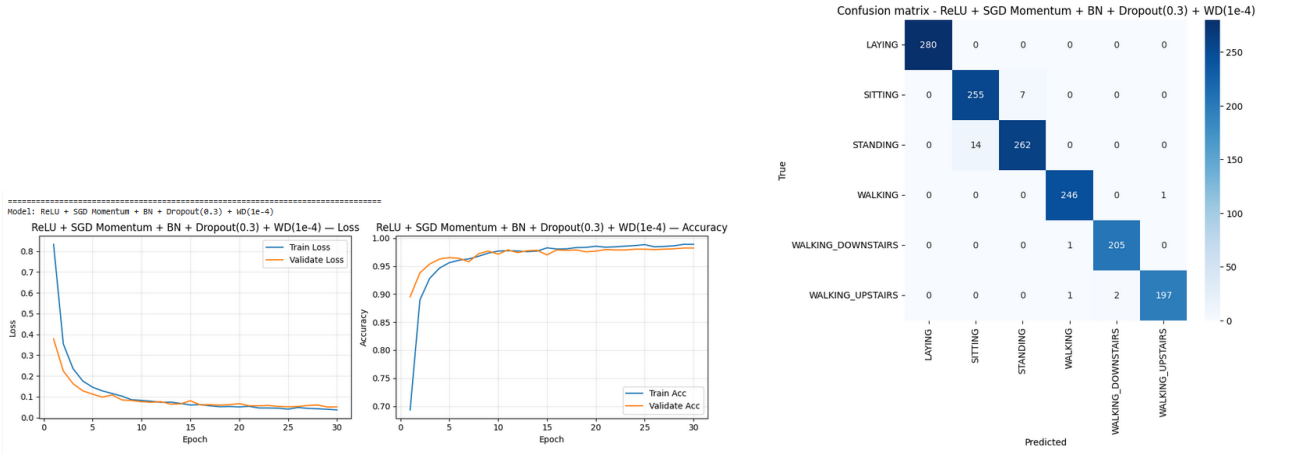
Après analyse des différentes techniques de régularisation, nous avons sélectionné les deux configurations les plus performantes pour approfondir l'étude et réaliser les évaluations finales :

— `ReLU + SGD Momentum` | Hidden = 128 | Layers = 2 | Dropout = 0.3 | Weight Decay = 10^{-4}
— `Tanh + Adam` | Hidden = 128 | Layers = 2 | Dropout = 0.3 | Weight Decay = 10^{-4}

6 Visualisation et analyse des performances

6.1 ReLU + SGD Momentum avec régularisation

La Figure 5a illustre les courbes de *loss* et d'*accuracy* pour la configuration : **ReLU + SGD Momentum | Hidden = 128 | Layers = 2 | Dropout = 0.3 | Weight Decay** = 10^{-4} **| BatchNorm**. On observe une convergence rapide et des courbes train/validation très proches, signe d'une bonne généralisation.



(a) Courbes Loss et Accuracy pour ReLU + SGD Momentum avec régularisation

(b) Matrice de confusion pour ReLU + SGD Momentum

FIGURE 5 – Courbes d'entraînement et matrice de confusion pour ReLU + SGD Momentum

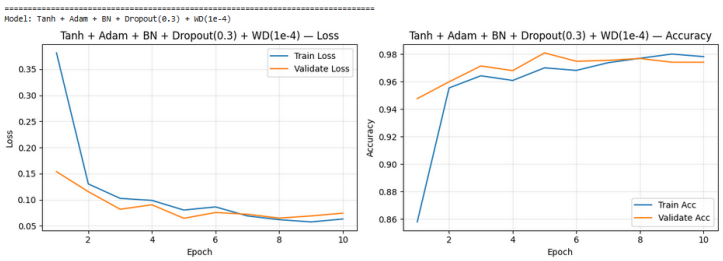
Le rapport de classification sur le jeu de validation montre une **accuracy globale de 98%** et un **F1-macro de 0.98**. Les classes dynamiques (WALKING, UPSTAIRS, DOWNSTAIRS) sont parfaitement séparées, tandis que la confusion principale concerne les postures statiques (SITTING vs STANDING).

Classe	Precision	Recall	F1-score	Support
LAYING	1.00	1.00	1.00	280
SITTING	0.95	0.97	0.96	262
STANDING	0.97	0.95	0.96	276
WALKING	0.99	1.00	0.99	247
WALKING_DOWNSTAIRS	0.99	1.00	0.99	206
WALKING_UPSTAIRS	0.99	0.98	0.99	200
Accuracy		0.98 (1471 échantillons)		
Macro avg	0.98	0.98	0.98	1471
Weighted avg	0.98	0.98	0.98	1471

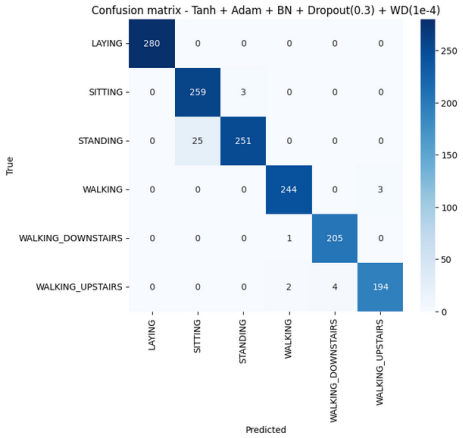
TABLE 3 – Classification report pour ReLU + SGD Momentum (validation)

6.2 Tanh + Adam avec régularisation

La Figure 6a présente les courbes pour la configuration : **Tanh + Adam | Hidden = 128 | Layers = 2 | Dropout = 0.3 | Weight Decay** = 10^{-4} **| BatchNorm**. Cette configuration atteint une **accuracy de 98%** et un **F1-macro de 0.98**, avec une convergence rapide et des courbes stables.



(a) Courbes Loss et Accuracy pour Tanh + Adam avec régularisation



(b) Matrice de confusion pour Tanh + Adam

FIGURE 6 – Courbes d'entraînement et matrice de confusion pour ReLU + SGD Momentum

La matrice de confusion (Figure 6b) confirme que la confusion principale reste entre SITTING et STANDING, mais elle est légèrement plus marquée que pour ReLU + SGD Momentum.

Classe	Precision	Recall	F1-score	Support
LAYING	1.00	1.00	1.00	280
SITTING	0.91	0.99	0.95	262
STANDING	0.99	0.91	0.95	276
WALKING	0.99	0.99	0.99	247
WALKING_DOWNSTAIRS	0.98	1.00	0.99	206
WALKING_UPSTAIRS	0.98	0.97	0.98	200
Accuracy		0.97 (1471 échantillons)		
Macro avg	0.98	0.98	0.97	1471
Weighted avg	0.98	0.97	0.97	1471

TABLE 4 – Classification report (validation)

6.3 Discussion

Ces deux configurations offrent un excellent compromis entre stabilité, rapidité de convergence et capacité de généralisation. ReLU + SGD Momentum présente des courbes légèrement plus régulières, tandis que Tanh + Adam converge plus vite mais avec une confusion statique un peu plus élevée. Les deux atteignent des performances très proches (F1-macro ≈ 0.98).

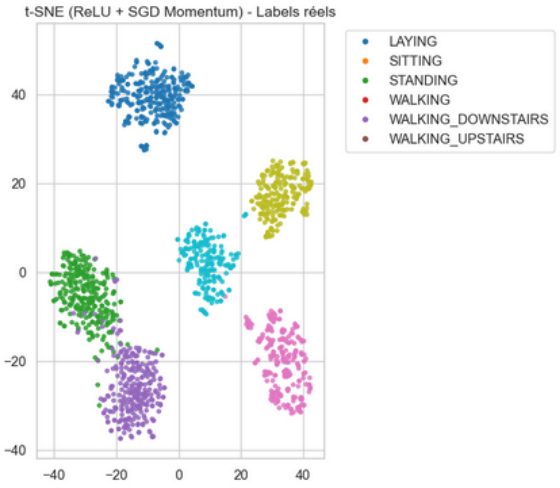
7 Interprétabilité et réduction de dimension

7.1 Principe des méthodes

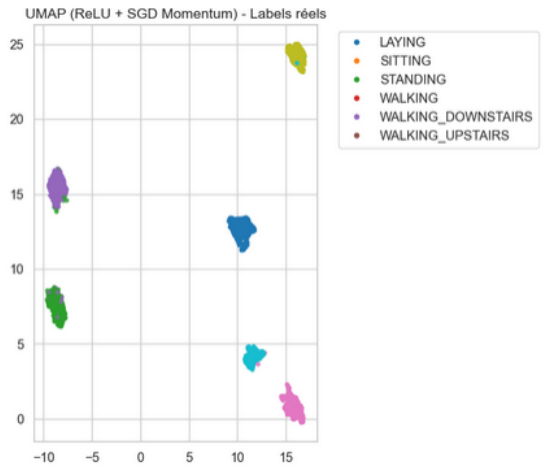
Les techniques de réduction de dimension comme **t-SNE** et **UMAP** permettent de projeter les activations de la dernière couche cachée dans un espace 2D afin de visualiser la structure latente des données. Bien que ces méthodes soient approximatives, elles offrent des insights sur la capacité du modèle à séparer les classes.

7.2 Visualisation des activations

La Figure 7 et la Figure 8 présentent les projections t-SNE et UMAP pour les deux configurations retenues.



(a) t-SNE (ReLU + SGD Momentum)

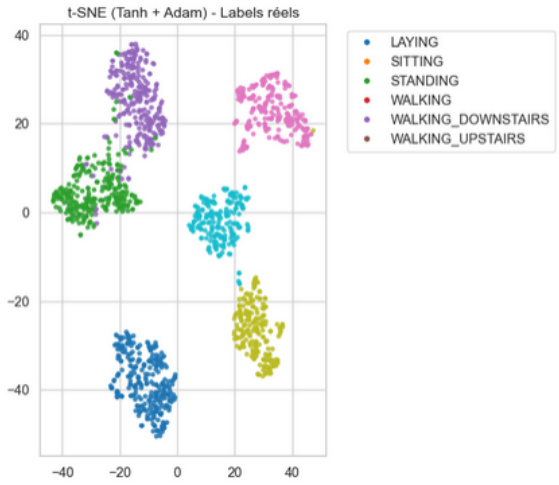


(b) UMAP (ReLU + SGD Momentum)

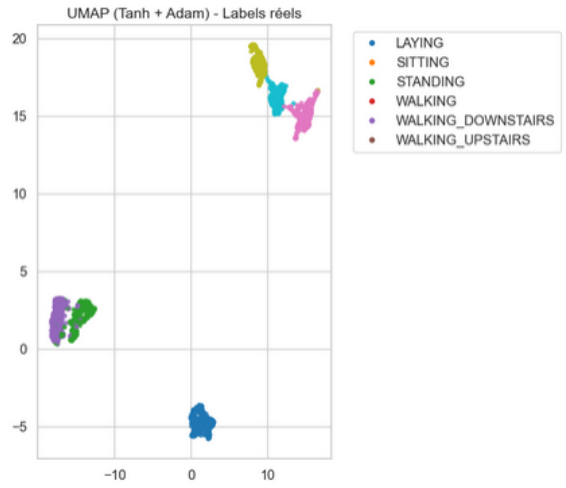
FIGURE 7 – Visualisation des activations pour ReLU + SGD Momentum

7.3 Analyse et comparaison

- Les deux modèles montrent une séparation nette des classes, confirmant leur bonne capacité de généralisation.
- **ReLU + SGD Momentum** : clusters compacts et bien séparés, très peu d'erreurs. La confusion STANDING/SITTING reste visible.
- **Tanh + Adam** : clusters également distincts, mais chevauchement plus marqué entre STANDING et SITTING, cohérent avec la matrice de confusion.
- UMAP offre une séparation encore plus nette que t-SNE, avec des erreurs concentrées dans les zones de chevauchement.



(a) t-SNE (Tanh + Adam)



(b) UMAP (Tanh + Adam)

FIGURE 8 – Visualisation des activations pour Tanh + Adam

8 Prédiction finale sur le jeu de test

9 Conclusion

Ce projet avait pour objectif d'étudier l'impact de la complexité des réseaux de neurones, des fonctions d'activation, des optimiseurs et des techniques de régularisation sur la classification des activités humaines à partir du jeu de données HAR. Après une analyse exploratoire, la mise en place de plusieurs architectures et l'application de régularisations, deux configurations optimales ont été retenues :

- **ReLU + SGD Momentum | Hidden = 128 | Layers = 2 | Dropout = 0.3 | Weight Decay** = 10^{-4} **| BatchNorm**
- **Tanh + Adam | Hidden = 128 | Layers = 2 | Dropout = 0.3 | Weight Decay** = 10^{-4} **| BatchNorm**

9.1 Performance finale sur le jeu de test

L'évaluation du meilleur modèle (**ReLU + SGD Momentum**) sur le jeu de test montre une excellente capacité de généralisation :

- **Accuracy** : 95.39 %
- **F1-macro** : 95.43 %

TABLE 5 – Rapport de classification sur le jeu de test

Classe	Précision	Rappel	F1-score
LAYING	1.00	0.98	0.99
SITTING	0.95	0.90	0.93
STANDING	0.89	0.96	0.92
WALKING	0.98	0.95	0.97
WALKING_DOWNSTAIRS	0.98	0.96	0.97
WALKING_UPSTAIRS	0.93	0.97	0.95
Macro avg	0.96	0.95	0.95
Accuracy globale		0.9539	

Les principales confusions concernent les postures statiques (**SITTING vs STANDING**) et certaines transitions dynamiques (**WALKING_UPSTAIRS vs WALKING_DOWNSTAIRS**), ce qui est cohérent avec la nature des signaux.

9.2 Synthèse et perspectives

En résumé :

- La régularisation (Dropout, Weight Decay, BatchNorm) a permis de réduire le sur-apprentissage et d'améliorer la stabilité.
- Les deux configurations retenues atteignent des performances élevées et une bonne robustesse.
- Les visualisations t-SNE et UMAP apportent des insights utiles sur la structure latente des données.

10 Utilisation de ChatGPT

Conformément aux consignes, j'ai utilisé ChatGPT pour :

- Automatiser certaines fonctions, notamment la création de boucles permettant de tester plusieurs configurations en même temps (nombre de couches, tailles, fonctions d'activation, optimiseurs) sans avoir à tout relancer à chaque fois.
- Générer des fonctions d'affichage pour visualiser les courbes de loss, d'accuracy et matrices de confusion, en factorisant le code pour le rendre plus modulaire.
- Reformuler certaines analyses pour reduire le nombre de pages du rapport.

J'ai précieusement vérifier toutes les sorties générées avant de les utiliser. Tout le code du projet est basé uniquement sur la méthodologie de l'enseignant et les exemples de TD.

Références

- UCI Machine Learning Repository. Human Activity Recognition Using Smartphones Dataset. Disponible sur Kaggle : `https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones`.
- Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.
- McInnes, L., Healy, J., & Melville, J. (2018). UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv :1802.03426*.
- Pedregosa, F., et al. (2011). Scikit-learn : Machine Learning in Python. *JMLR*, 12, 2825–2830.
- McInnes, L., Healy, J., Saul, N., & Großberger, L. (2020). UMAP Documentation. `https://umap-learn.readthedocs.io`
- Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. (2013). A Public Domain Dataset for Human Activity Recognition Using Smartphones. *ESANN 2013*.