

Rapport de Travaux Pratiques 2

Amélioration de Réseaux de Neurones Profonds

Joseph BOUIYODA
Département Génie Informatique, ENSPY
buiyodajoseph@gmail.com

21 janvier 2026

Table des matières

1	Introduction	3
I	Concepts Théoriques	3
2	Diagnostic de Performance : Biais vs. Variance	3
3	Régularisation et Normalisation	3
4	Algorithmes d'Optimisation Avancés	3
II	Exercices Pratiques et Analyses	4
5	Exercice 1 : Analyse Biais/Variance	4
5.1	Objectif	4
5.2	Résultats	4
5.3	Analyse et Réponse	4
6	Exercice 2 : Application de la Régularisation	4
6.1	Objectif	4
6.2	Résultats	4
6.3	Analyse et Réponse	5
7	Exercice 3 : Comparaison des Optimiseurs	5
7.1	Objectif	5
7.2	Résultats	5
7.3	Analyse	5
8	Exercice 4 : Batch Normalization	5
8.1	Objectif	5
8.2	Résultats	5
8.3	Analyse	6
9	Conclusion Générale	6

1 Introduction

Ce second rapport de Travaux Pratiques se concentre sur les techniques avancées d'ingénierie pour l'amélioration des réseaux de neurones. En partant du modèle de base développé dans le TP1, nous allons diagnostiquer ses faiblesses, notamment le surapprentissage, et appliquer systématiquement des solutions pour l'améliorer. Ce rapport détaillera l'implémentation et l'analyse de techniques de régularisation, la comparaison d'optimiseurs, et l'impact de la Batch Normalization, en s'appuyant sur des expérimentations suivies rigoureusement avec MLflow.

Première partie

Concepts Théoriques

2 Diagnostic de Performance : Biais vs. Variance

Division des données : Il est crucial de diviser les données en trois ensembles. L'**ensemble d'entraînement** est utilisé pour ajuster les poids du modèle. L'**ensemble de validation** sert à évaluer le modèle pendant le développement pour ajuster les hyperparamètres sans "contaminer" l'évaluation finale. L'**ensemble de test**, utilisé une seule fois à la fin, donne une mesure honnête et impartiale de la performance de généralisation du modèle final.

Analyse des résultats : Le diagnostic s'effectue en comparant l'erreur d'entraînement et l'erreur de validation. Un **biais élevé** (sous-apprentissage) se manifeste par une erreur élevée sur les deux ensembles. Une **variance élevée** (surapprentissage) se caractérise par une erreur faible sur l'entraînement mais significativement plus élevée sur la validation.

3 Régularisation et Normalisation

- **Régularisation L2 (Weight Decay) :** Cette technique ajoute une pénalité à la fonction de coût, proportionnelle à la somme des carrés des poids du modèle. Cela contraint le modèle à privilégier des poids de faible valeur, ce qui réduit sa complexité et sa capacité à mémoriser le bruit dans les données d'entraînement, améliorant ainsi sa généralisation.
- **Dropout :** Le Dropout consiste à désactiver aléatoirement une fraction de neurones à chaque étape de l'entraînement. Cela empêche les neurones de co-adapter leurs apprentissages et les force à devenir individuellement plus robustes, ce qui agit comme une puissante technique de régularisation.
- **Batch Normalization :** Cette technique normalise les activations de chaque couche pour avoir une moyenne nulle et une variance unitaire. Cela stabilise le processus d'apprentissage, combat le problème de la disparition/explosion du gradient, et permet d'utiliser des taux d'apprentissage plus élevés, accélérant ainsi considérablement la convergence du modèle.

4 Algorithmes d'Optimisation Avancés

Momentum : Améliore la SGD en introduisant une "inertie" qui est une moyenne mobile des gradients passés. Cela permet d'accélérer la descente dans les directions pertinentes et d'amortir les oscillations.

RMSprop : Adapte le taux d'apprentissage pour chaque paramètre individuellement, en le divisant par une moyenne mobile des gradients récents au carré. Il est ainsi moins sensible au choix initial du taux d'apprentissage.

Adam : Combine les avantages de Momentum et de RMSprop. Il conserve une mémoire des gradients passés (comme Momentum) et utilise un taux d'apprentissage adaptatif pour chaque paramètre (comme RMSprop). C'est pourquoi il est souvent considéré comme le choix par défaut, car il est efficace, rapide et robuste.

Deuxième partie

Exercices Pratiques et Analyses

5 Exercice 1 : Analyse Biais/Variance

5.1 Objectif

Diagnostiquer le comportement du modèle de base du TP1 en utilisant un ensemble de validation explicite pour identifier les problèmes de biais ou de variance.

5.2 Résultats

Le modèle a été entraîné sur 15 époques. Les métriques de la dernière époque sont les suivantes :

- Précision d'entraînement : **99.61%**
- Précision de validation : **98.22%**

5.3 Analyse et Réponse

Question : D'après les résultats, le modèle souffre-t-il de biais ou de variance ? Justifiez.

Réponse : Le modèle présente un **biais faible** mais souffre d'une **variance élevée (sur-apprentissage)**.

Justification : Le biais est faible car le modèle atteint une performance quasi parfaite sur l'ensemble d'entraînement (99.61%). Cependant, l'écart de performance de 1.39% avec l'ensemble de validation (98.22%) indique un début de surapprentissage. Ce diagnostic est confirmé par l'analyse des courbes d'apprentissage, où la perte de validation (`val_loss`) atteint son minimum à la 12ème époque (0.0592) avant de remonter, tandis que la perte d'entraînement continue de diminuer. Le modèle commence à mémoriser les données d'entraînement au détriment de sa capacité de généralisation.

6 Exercice 2 : Application de la Régularisation

6.1 Objectif

Corriger le surapprentissage identifié en ajoutant des techniques de régularisation (L2 et Dropout).

6.2 Résultats

Après ajout d'une régularisation L2 et d'un Dropout de 0.5, les métriques à la fin de l'entraînement étaient :

- Précision d'entraînement : **96.40%**
- Précision de validation : **98.02%**

6.3 Analyse et Réponse

Question : Les techniques de régularisation ont-elles amélioré les performances sur l'ensemble de validation ? Expliquez pourquoi.

Réponse : Oui, les techniques de régularisation ont été extrêmement efficaces pour lutter contre le surapprentissage et améliorer la robustesse du modèle.

Explication : L'effet le plus marquant est que la précision de validation (98.02%) est devenue **supérieure** à la précision d'entraînement (96.40%). Ce phénomène s'explique par le fait que le Dropout est actif pendant l'entraînement (le modèle fonctionne à capacité réduite) mais est désactivé pendant la validation (le modèle utilise 100% de ses neurones). Cela prouve que le surapprentissage a été complètement éliminé. En contraignant le modèle à apprendre des caractéristiques plus générales, la régularisation a amélioré sa capacité à bien performer sur de nouvelles données.

7 Exercice 3 : Comparaison des Optimiseurs

7.1 Objectif

Comparer la performance et la vitesse de convergence de trois optimiseurs (SGD avec momentum, RMSprop, Adam) en utilisant MLflow pour le suivi des expériences.

7.2 Résultats

Les performances finales sur l'ensemble de test après 10 époques sont résumées dans le Tableau 1.

Optimiseur	Précision de Test Finale
Adam	98.28%
RMSprop	98.23%
SGD with Momentum	97.38%

TABLE 1 – Performances finales des différents optimiseurs.

7.3 Analyse

L'analyse des courbes de convergence dans MLflow (voir Figure ??) révèle que les optimiseurs adaptatifs **Adam** et **RMSprop** convergent significativement plus vite que SGD, atteignant une haute précision en seulement 3-4 époques. Adam obtient la meilleure performance finale, confirmant son statut de choix par défaut robuste et efficace.

8 Exercice 4 : Batch Normalization

8.1 Objectif

Évaluer l'impact de l'ajout d'une couche de Batch Normalization sur la vitesse d'entraînement.

8.2 Résultats

Le modèle avec Batch Normalization a atteint une précision finale de test de **97.86%**.

8.3 Analyse

Le bénéfice principal de la Batch Normalization a été une **accélération nette de la convergence**. Le modèle a atteint une précision de validation de **97.62% dès la deuxième époque**, alors que le modèle Adam sans cette couche nécessitait quatre époques pour atteindre un niveau similaire. En stabilisant la distribution des activations, la Batch Normalization a permis un apprentissage plus rapide et stable, tout en ayant un léger effet de régularisation bénéfique.

9 Conclusion Générale

Ce TP a permis de suivre une démarche d'ingénierie complète : partir d'un modèle de base, diagnostiquer ses faiblesses (variance élevée), et appliquer des solutions ciblées pour l'améliorer. La régularisation a résolu le surapprentissage, la comparaison d'optimiseurs a mis en évidence l'efficacité d'Adam, et la Batch Normalization a démontré son utilité pour accélérer l'entraînement. L'utilisation systématique de MLflow a permis un suivi rigoureux et une analyse comparative objective, soulignant l'importance des bonnes pratiques MLOps dans le développement de modèles de Deep Learning performants.