Research project

UNSUPERVISED DATA AUGMENTATION FOR CONSISTENCY TRAINING

Jaad Belhouari
Léos Coutrot
Luc Yao

Université Paris-Saclay
Blaise Hanczar - M. Kadrajat

16 janvier 2025

université PARIS-SACLAY

Introduction

Contexte et Objectif

- Exploration d'**UDA**, méthode d'apprentissage semi-supervisée, sur le dataset **MNIST**.
- ***Objectif*** : Évaluer l'efficacité d'UDA dans un cadre d'apprentissage semi-supervisé avec très peu de labels.

Dataset MNIST

- **60000** images de chiffres manuscrits, **28x28** pixels
- 100 données **labélisées** | 59 900 **non labélisées**

L'article : *Unsupervised Data Augmentation for Consistency Training*

Méthodologie

- **Augmentation des images non labelisées** : un modèle "parfait" devrait prédir le même résultat pour une image et une autre légèrement bruitée.
- **Combinaison des fonctions de perte** : On combine les résultats des fonctions de pertes pour les prédictions supervisées et non-supervisées.
- Choix des **hyperparamètres** optimaux.

Fonction de Perte Total d'UDA

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim p_L(x)} \left[-\log p_\theta(y^*|x) \right]$$

$$+\lambda \mathbb{E}_{x \sim p_U(x)} \mathbb{E}_{\hat{x} \sim q(\hat{x}|x)} \left[\mathrm{CE}(p_{\tilde{\theta}}(y|x), p_\theta(y|\hat{x})) \right],$$

Architecture du réseau

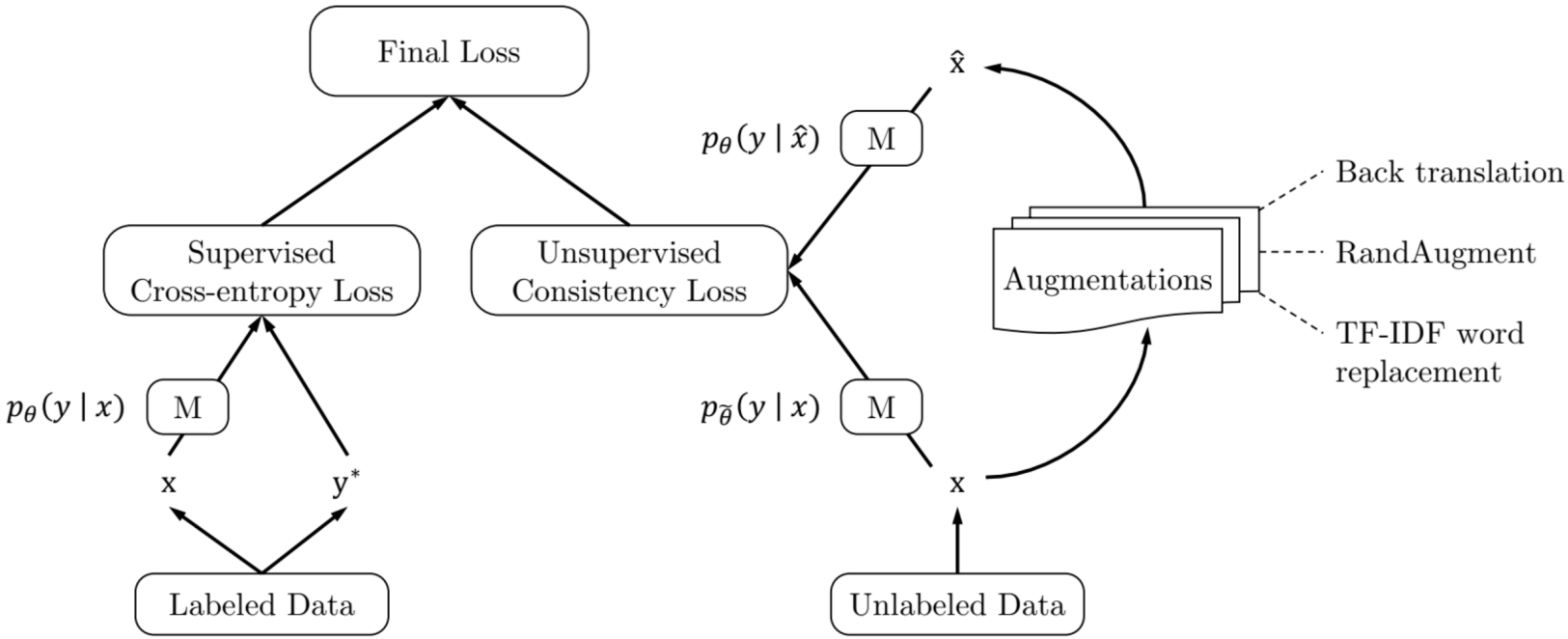

Figure 1 – Illustration d'UDA

Méthodologie

Axée en plusieurs étapes :

1. Analyse des données
2. Pré-traitement
3. Implémentation d'une baseline et d'un modèle UDA
4. Optimisation des hyperparamètres
5. Comparaison des résultats

Pré-traitement des Données

Normalisation des images

- Valeurs de pixel ramenés à $[0, 1]$ à partir de $[0, 255]$.

$$\text{Pixel normalisé} = \frac{\text{Pixel original}}{255}$$

$\rightarrow$ *Objectif* : **stabiliser** l'apprentissage et assurer une influence **comparable** des pixels.

Augmentation des données

Augmentation de Données Labelisées

- **Rotations aléatoires** (15 degrés)
- Ajustement de la **luminosité** (variation de 20%)
- Recadrage et zoom aléatoire (20%).

Augmentation de Données Non-Labelisées

- Utilisation de **RandAugment** pour bruiter les données non labelisées

Augmentation des données

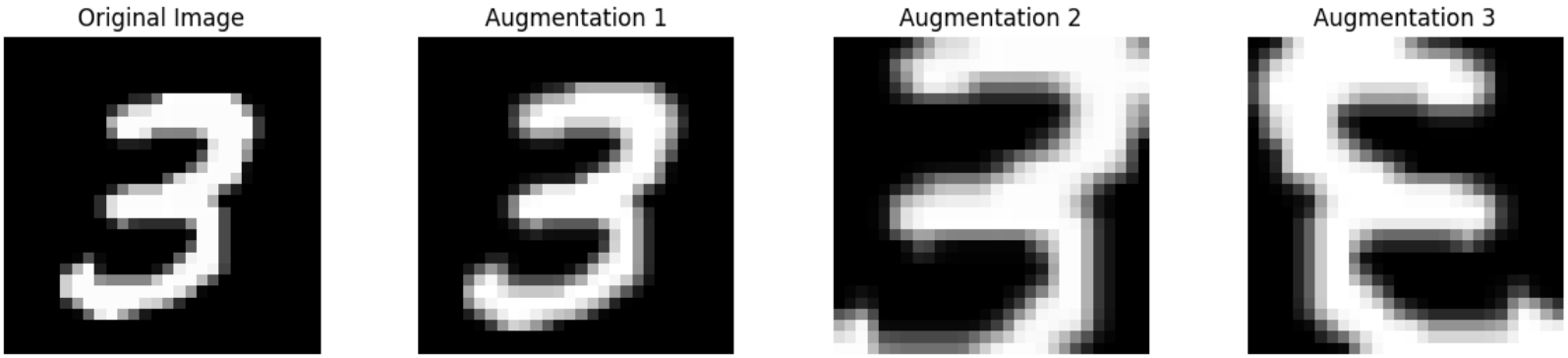

Figure 2 – Exemples de data augmentation réalisé sur le chiffre 3

Figure 3 – Exemples d'augmentations via RandAugment

Jeu de données d'entraînement et de validation

Méthode utilisée pour séparer les données

- Séparation des jeux de données d'entraînement et de validation non conventionnelle.
- Effectuée après avoir augmenté le jeu de données labelisé.
- Entraîne un biais dans les retours de notre jeu de données de validation mais permet de tirer profit au maximum de toutes nos données labelisées.

Architecture CNN

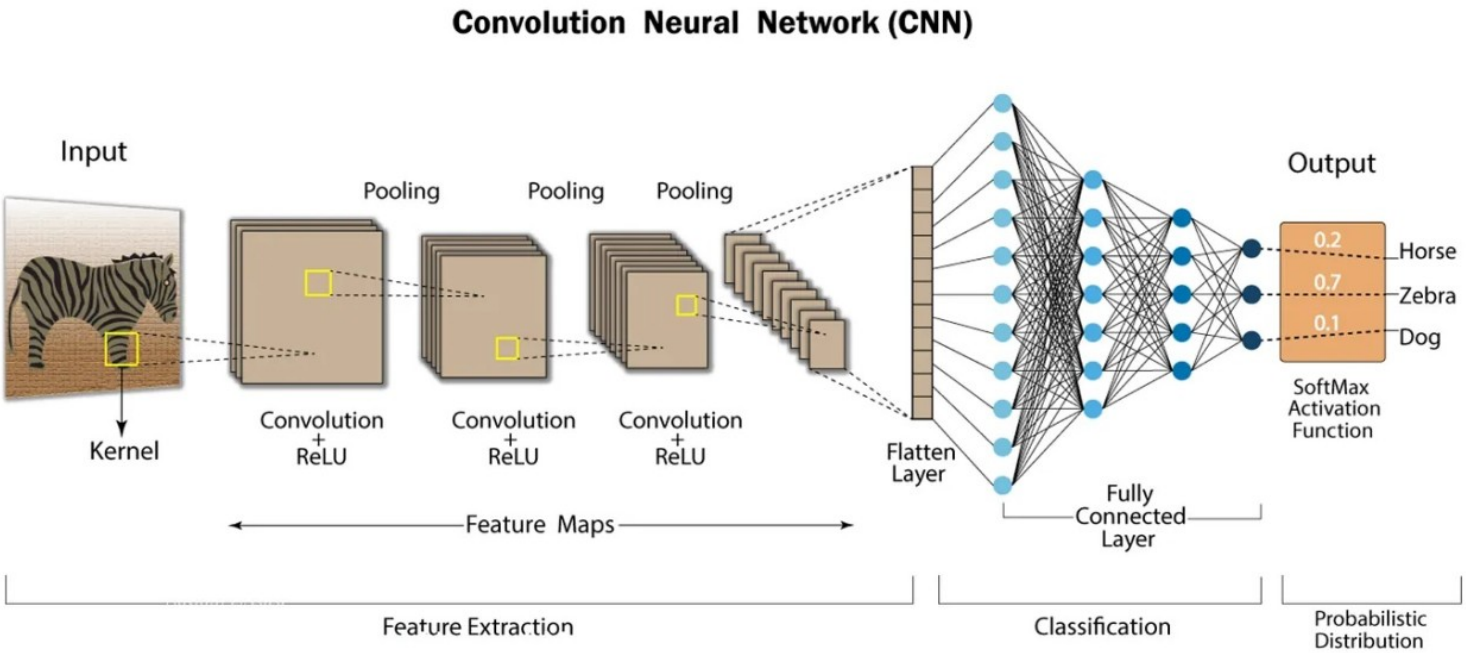

Figure 4 – Architecture classique d'un CNN

Architecture de la Baseline

Couche	Taille
Conv2D (32, 3, 'relu', padding='same')	(28, 28, 32)
Conv2D (32, 3, 'relu', padding='same')	(28, 28, 32)
MaxPooling2D (2, 2)	(14, 14, 32)
Dropout (0.25)	(14, 14, 32)
Flatten	(6272,)
Dense (512, 'relu')	(512,)
Dropout (0.5)	(512,)
Dense (10, 'softmax')	(10,)

Table 1 – Résumé de l'architecture de notre baseline

Architecture du modèle avancé

Couche	Taille
Conv2D (32, 3, 'relu', padding=1)	(28, 28, 32)
BatchNorm2D	(28, 28, 32)
Conv2D (32, 3, 'relu', padding=1)	(28, 28, 32)
BatchNorm2D	(28, 28, 32)
Conv2D (32, 5, 'relu', stride=2, padding=2)	(14, 14, 32)
BatchNorm2D	(14, 14, 32)
Dropout (0.3)	(14, 14, 32)
Conv2D (64, 3, 'relu', padding=1)	(14, 14, 64)
BatchNorm2D	(14, 14, 64)
Conv2D (64, 3, 'relu', padding=1)	(14, 14, 64)
BatchNorm2D	(14, 14, 64)
Conv2D (64, 5, 'relu', stride=2, padding=2)	(7, 7, 64)
BatchNorm2D	(7, 7, 64)
Dropout (0.4)	(7, 7, 64)
Flatten	(3136)
Dense (128, 'relu')	(128)
Dropout (0.5)	(128)
Dense (10, 'softmax')	(10)

Table 2 – Résumé de l'architecture du modèle avancé

Optimisation des hyper-paramètres

Fine-tuning du modèle

- Les hyperparamètres sont des paramètres qui ne sont **pas appris par le modèle**

Optimisation des hyper-paramètres

Fine-tuning du modèle

- Les hyperparamètres sont des paramètres qui ne sont **pas appris par le modèle**
- L'**optimisation** des hyperparamètres est une étape **nécessaire** dans l'apprentissage automatique

Optimisation des hyper-paramètres

Fine-tuning du modèle

- Les hyperparamètres sont des paramètres qui ne sont **pas appris par le modèle**
- L'**optimisation** des hyperparamètres est une étape **nécessaire** dans l'apprentissage automatique
- Impact **significatif** sur les performances du modèle

Optimisation des hyper-paramètres : Méthodologie

Recherche exhaustive

- Mise en place d'un learning rate scheduler personnalisé (fonctionne par palliers).

→ Recherche empirique en comprenant le sens de ces paramètres.

Optimisation des hyper-paramètres : Méthodologie

Recherche exhaustive

- Mise en place d'un learning rate scheduler personnalisé (fonctionne par palliers).
- Algorithme d'optimisation de descente de gradient stochastique SGD.

$\rightarrow$ Recherche empirique en comprenant le sens de ces paramètres.

Optimisation des hyper-paramètres : Méthodologie

Recherche exhaustive

- Mise en place d'un learning rate scheduler personnalisé (fonctionne par palliers).
- Algorithme d'optimisation de descente de gradient stochastique SGD.
- Optimisation d'autres hyperparamètres, tels que le **coefficients** λ, la **température** de la divergence K-L ou encore les **batch size**.

$\rightarrow$ Recherche empirique en comprenant le sens de ces paramètres.

Configuration finale (qui a offert les meilleurs performances)

Variable	Valeur
train_batch_size	10
unsup_batch_size	64
eval_batch_size	64
learning_rate	5×10^{-5}
train_steps	5000
uda_temp	0.3
unsup_coeff	1.1
log_steps	100
eval_steps	100

Table 3 – Configuration des hyperparamètres

Résultats des modèles sans utilisation d'UDA

Métrique	Baseline	Modèle avancé
Test Accuracy	0.79	0.23

→ *objectif* : **améliorer** les performances de cette ligne de base.

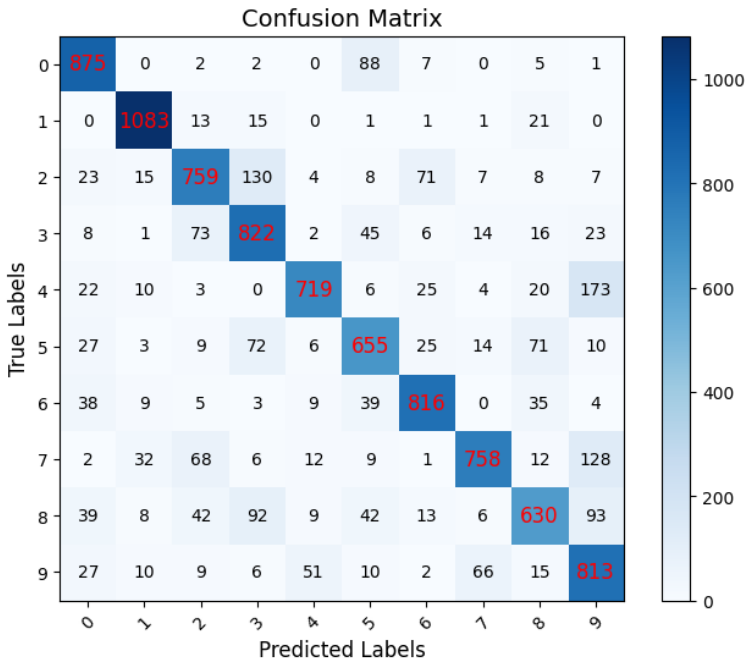

Figure 5 – Matrice de confusion de la Baseline

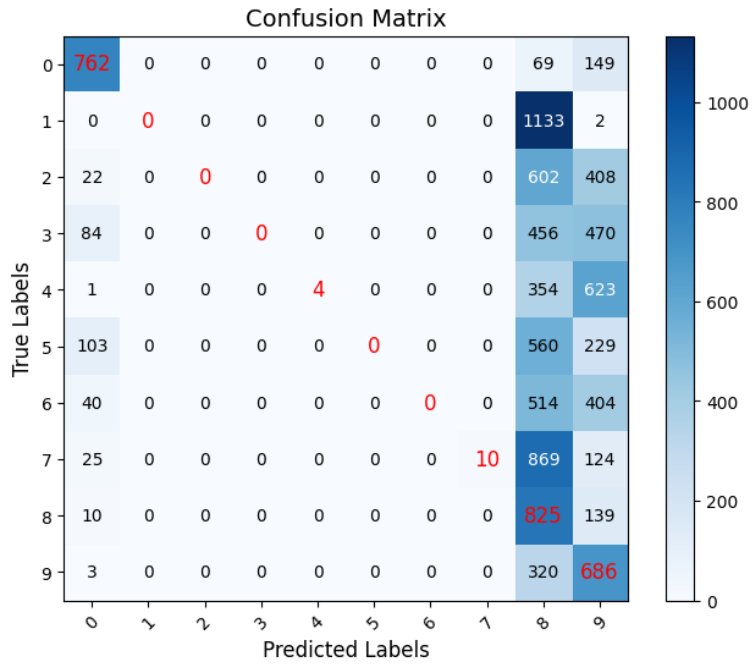

Figure 6 – Matrice de confusion du modèle avancé

Résultats Baseline + UDA

Métrique	Valeur
Test Accuracy	0.88

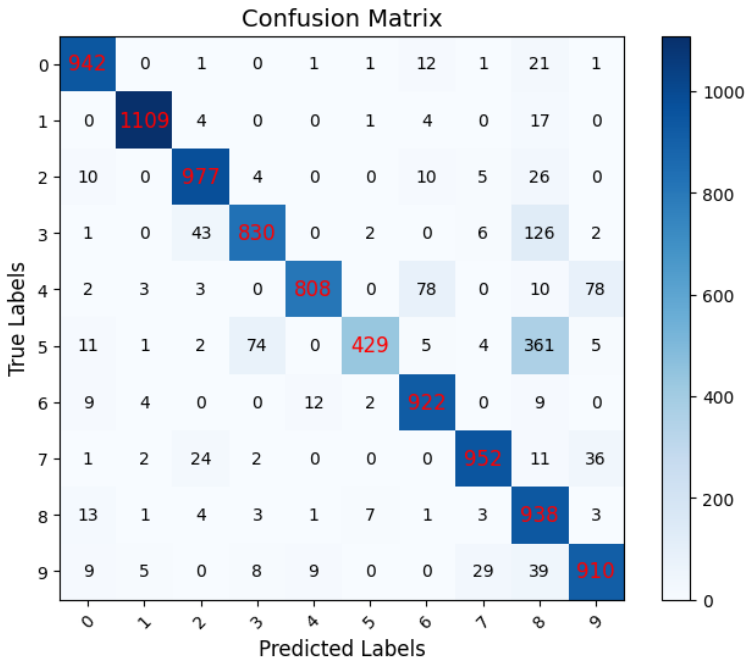

Figure 7 – Matrice de confusion de la baseline avec UDA

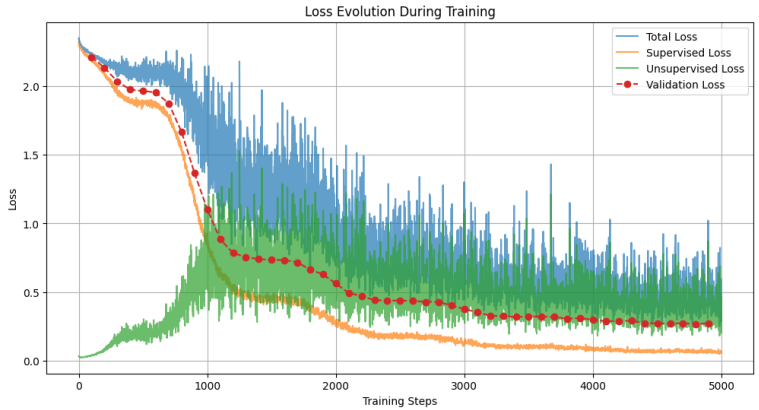

Figure 8 – Loss de la baseline avec UDA

Résultats du modèle avancé + UDA

Métrique	Valeur
Test Accuracy	0.97

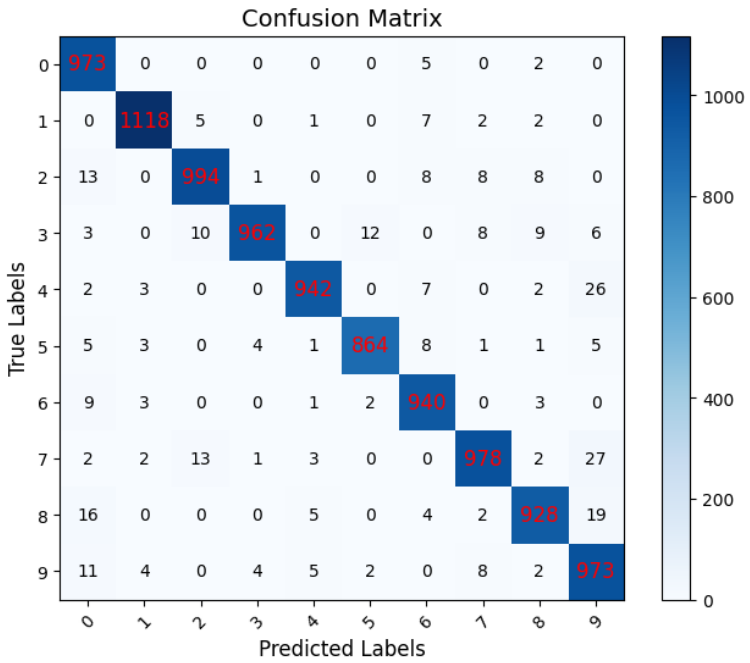

Figure 9 – Matrice de confusion du modèle avancé avec UDA

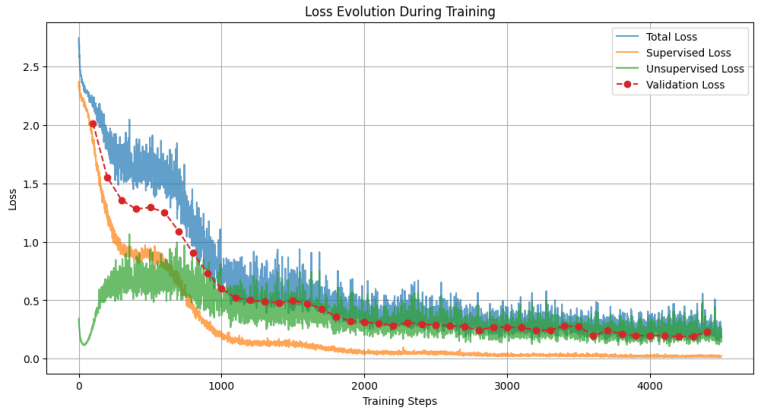

Figure 10 – Loss du modèle avancé + UDA

Comparaison : Impact d'UDA

Modèle	Utilisation de UDA	Temps d'exécusion	Accuracy
Modèle témoin	Non	3min (20 epochs)	0.79
Modèle témoin	Oui	20min (4000 train step)	0.88
Modèle avancé	Non	5min (20 epochs)	0.23
Modèle avancé	Oui	30min (5000 train step)	**0.97**

Table 4 – Résumé des performances des modèles explorés

→ UDA permet de **systématiquement** améliorer les performances de nos modèles.

Interprétation des Résultats

- **Excellente Performance**
  → Offre systématiquement une amélioration des performances finales.

- **Avantages**
  → Facilement reproductible dans de nombreux contextes
  → L'accès à de grands jeux de données labelisées est parfois très délicat

Perspectives d'Amélioration

- **Amélioration de l'optimisation des hyperparamètres**
  $\rightarrow$ utilisation de GridSearch pour certains hyperparamètres
- **Utilisations de méthodes ensemblistes**
  $\rightarrow$ Bagging, lourd mais pourrait améliorer la précision finale
- Mise en place d'un **Early Stopping**