AgroParisTech INRAE

MLOps pour la reproductibilité et la traçabilité des expériences d'apprentissage

Structurer des projets de recherche avec UV, Hydra et MLflow

Julian Agudelo et Joseph Allyndrée

MIA
PARIS-SACLAY
EKINOCS

23/04/2026

Plan

Motivations

Le challenge de la reproductibilité

Reproductibilité

Capacité à pouvoir (par une tierce personne) reproduire des résultats, en réutilisant un dispositif expérimental décrit.

Le challenge de la reproductibilité

Reproductibilité

Capacité à pouvoir (par une tierce personne) reproduire des résultats, en réutilisant un dispositif expérimental décrit.
Cela implique de facto différents niveaux :

- Méthodes (décrire en détail les protocoles utilisés)
- Données (données accessibles)
- Code (traitement des données)

Le challenge de la reproductibilité

Reproductibilité

Capacité à pouvoir (par une tierce personne) reproduire des résultats, en réutilisant un dispositif expérimental décrit.
Cela implique de facto différents niveaux :

- Méthodes (décrire en détail les protocoles utilisés)
- Données (données accessibles)
- Code (traitement des données)

Pourquoi est-ce important que la recherche soit reproductible ?

Le challenge de la reproductibilité

Il faut que la science produite par la recherche soit juste !

- Crise de la reproductibilité autour de 2015
- Papier fondamental : "Scientists lift the lid on reproducibility"

Entache la **crédibilité** de la recherche et empêche une science **construite** sur les développements passés !

BAKER 2016 ; COLLABORATION 2015

Le défi de la traçabilité dans l'apprentissage profond

Le code classique est programmé explicitement

- exigence $\rightarrow$ fonctionnalité $\rightarrow$ code

Le défi de la traçabilité dans l'apprentissage profond

Le code classique est programmé explicitement

- exigence $\rightarrow$ fonctionnalité $\rightarrow$ code

Un réseau neuronal est appris implicitement

- La logique finale est répartie entre le code et les paramètres appris
- Processus de développement par « essai-erreur »

Le défi de la traçabilité dans l'apprentissage profond

Le code classique est programmé explicitement

- exigence $\rightarrow$ fonctionnalité $\rightarrow$ code

Un réseau neuronal est appris implicitement

- La logique finale est répartie entre le code et les paramètres appris
- Processus de développement par « essai-erreur »
  - Données
  - Boucle de pré-traitement
  - Architecture du modèle
  - Hyperparamètres
  - Régime d'entraînement

ARAVANTINOS et DIEHL 2019

Artefacts et traçabilité

Artefact

Document livré avec le logiciel

- Code source
- Code exécutable
- Résultats de tests

Artefacts et traçabilité

Artefact

Document livré avec le logiciel

- Code source
- Code exécutable
- Résultats de tests

Traçabilité

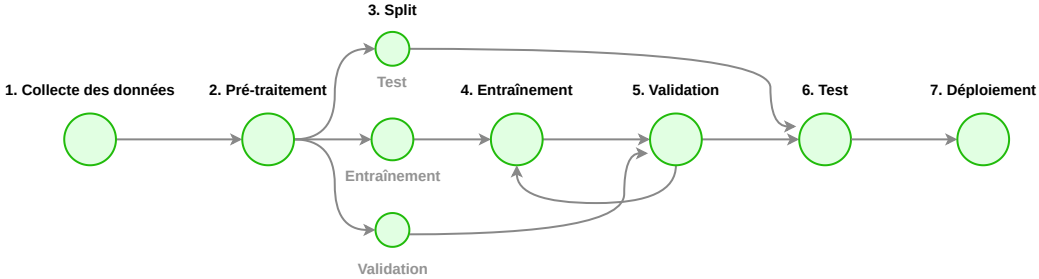
Les artefacts sont liés entre eux
e.g. : le code exécutable est dérivé du code source

- **Trace** : relation entre deux artefacts
- **Traçabilité** : ensemble des artefacts et de leurs relations

À ceci, il faut ajouter la contrôle des versions !

ARAVANTINOS et DIEHL 2019

Artefacts en apprentissage profond



Entraînement :

1. Collecter des données et les pré-traiter
   Ensemble de données brutes et fonctions de pré-traitement

Entraînement :

1. Collecter des données et les pré-traiter
   Ensemble de données brutes et fonctions de pré-traitement
2. Diviser l'ensemble de données
   Ensembles de données d'apprentissage, de validation et de test

Entraînement :

1. Collecter des données et les pré-traiter
   Ensemble de données brutes et fonctions de pré-traitement
2. Diviser l'ensemble de données
   Ensembles de données d'apprentissage, de validation et de test
3. Concevoir l'architecture
   Architecture

Artefacts en apprentissage profond

Entraînement :

1. Collecter des données et les pré-traiter
   Ensemble de données brutes et fonctions de pré-traitement
2. Diviser l'ensemble de données
   Ensembles de données d'apprentissage, de validation et de test
3. Concevoir l'architecture
   Architecture
4. Définir le « protocole d'apprentissage »
   Fonction de perte et paramètres d'apprentissage (e.g. hyperparamètres, dropout, taux d'apprentissage, optimiseur)

Artefacts en apprentissage profond

Entraînement :

1. Collecter des données et les pré-traiter
   Ensemble de données brutes et fonctions de pré-traitement
2. Diviser l'ensemble de données
   Ensembles de données d'apprentissage, de validation et de test
3. Concevoir l'architecture
   Architecture
4. Définir le « protocole d'apprentissage »
   Fonction de perte et paramètres d'apprentissage (e.g. hyperparamètres, dropout, taux d'apprentissage, optimiseur)
5. Entraîner le modèle
   Paramètres appris

ARAVANTINOS et DIEHL 2019

Inférence :

1. Effectuer un post-traitement du réseau entraîné (e.g. enlever le dropout)
   Architecture d'inférence

Inférence :

1. Effectuer un post-traitement du réseau entraîné (e.g. enlever le dropout)
   Architecture d'inférence
2. Tester le modèle sur l'ensemble de validation
   Real numbers representing accuracy

Inférence :

1. Effectuer un post-traitement du réseau entraîné (e.g. enlever le dropout)
   Architecture d'inférence
2. Tester le modèle sur l'ensemble de validation
   Real numbers representing accuracy
3. Modifier l'architecture ou la configuration d'apprentissage (3–4) en fonction des résultats

Inférence :

1. Effectuer un post-traitement du réseau entraîné (e.g. enlever le dropout)
   Architecture d'inférence
2. Tester le modèle sur l'ensemble de validation
   Real numbers representing accuracy
3. Modifier l'architecture ou la configuration d'apprentissage (3–4) en fonction des résultats
4. Évaluer la qualité à l'aide de l'ensemble de test
   Métriques finales

ARAVANTINOS et DIEHL 2019

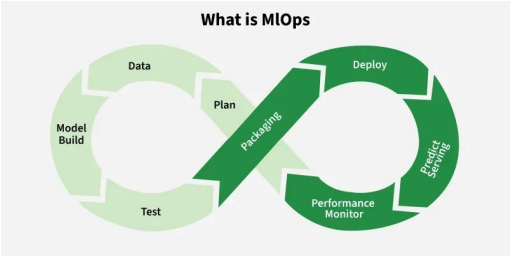
MLOps : Machine Learning Operations

Suivre, tracker et superviser l'entrainement et le déploiement de modèles d'IA à l'échelle.

TRAÇABILITÉ

MLOps

- Les données changent en permanence
- Les hyperparamètres changent
- Les architectures des modèles évoluent
- Protocoles de test différentes
- Monitoring une fois déployé



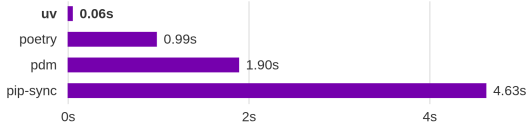
In spaghetti code, the relations between the pieces of code are so tangled that it is nearly impossible to add or change something without unpredictably breaking something somewhere else.

Outils pour le MLOPS

Des outils MLOps pour gérer du code scientifique

UV

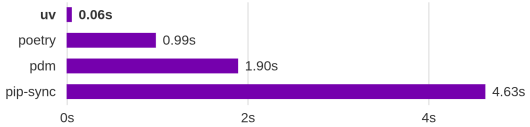
UV, un outil de gestion tout-en-un



- Gestionnaire de paquets
  `pip, poetry`

Lien vers la documentation de uv : https://docs.astral.sh/uv/

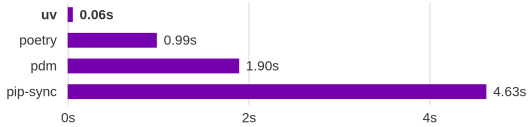
UV, un outil de gestion tout-en-un



- Gestionnaire de paquets
  `pip, poetry`
- Gestionnaire d'environnements virtuels
  `venv, conda`

Lien vers la documentation de uv : https://docs.astral.sh/uv/

UV, un outil de gestion tout-en-un



- Gestionnaire de paquets
  `pip, poetry`
- Gestionnaire d'environnements virtuels
  `venv, conda`
- Résolveur de dépendances
  `pip`

Lien vers la documentation de uv : https://docs.astral.sh/uv/

La philosophie derrière UV

Versionné, reproductible et portable

- Spécifier la version exacte de l'interpréteur
- Configuration déclarative du projet
- Favoriser une collaboration reproductible
- Considérer les environnements comme partie du projet

La philosophie derrière UV

Versionné, reproductible et portable

UV par rapport à d'autres outils de reproductibilité

Avantages

- Installation et configuration minimale
- Courbe d'apprentissage extrêmement rapide
- C'est tout aussi facile de créer un projet que de reprendre celui de quelqu'un d'autre

UV par rapport à d'autres outils de reproductibilité

Avantages

- Installation et configuration minimale
- Courbe d'apprentissage extrêmement rapide
- C'est tout aussi facile de créer un projet que de reprendre celui de quelqu'un d'autre

Inconvénients

- La reproductibilité commence au niveau de la version de Python
- Reproductibilité partielle entre les systèmes d'exploitation

UV vs Docker

	UV	docker
Niveau	Python	Système
Gestion	Dépendences	Environnement
Isolation	Venv	Conteneur OS
Usage	Développement	Déploiement
Performance	Très rapide	Plus lourd

UV vs Docker

	UV	docker
Niveau	Python	Système
Gestion	Dépendences	Environnement
Isolation	Venv	Conteneur OS
Usage	Développement	Déploiement
Performance	Très rapide	Plus lourd

Actuellement, UV est utilisé comme gestionnaire de paquets pour Python dans docker

Il est de 10 à 100 fois plus rapide que pip !

Hydra

Deux fonctionalités principales :

1. Créer une configuration hiérarchique par composition
2. La modifier via des fichiers de configuration et la ligne de commande

Lien vers la documentation d’Hydra : https://hydra.cc/

YADAN 2019

Un exemple minimaliste d'utilisation d'Hydra

1. **Un dossier de configuration**

```
.
├── conf
│   └── config.yaml
└── main.py
```

Un exemple minimaliste d'utilisation d'Hydra

1. Un dossier de configuration

```
.
├── conf
│   └── config.yaml
└── main.py
```

2. Des paramètres configurables

```
### conf/config.yaml

batch_size: 10
lr: 1e-4
```

Un exemple minimaliste d'utilisation d'Hydra

1. Un dossier de configuration

```
.
├── conf
│   └── config.yaml
└── main.py
```

2. Des paramètres configurables

```yaml
### conf/config.yaml

batch_size: 10
lr: 1e-4
```

3. Un script qui utilise la configuration

```python
import hydra

@hydra.main(config_path="conf", config_name="config")
def func(cfg: DictConfig):
    # On fait un print des paramtres, par exemple
    print(f"The batch size is {cfg.batch_size}")
    print(f"The learning rate is {cfg.lr}")

if __name__ == "__main__":
    func()
```

Un exemple minimaliste d'utilisation d'Hydra

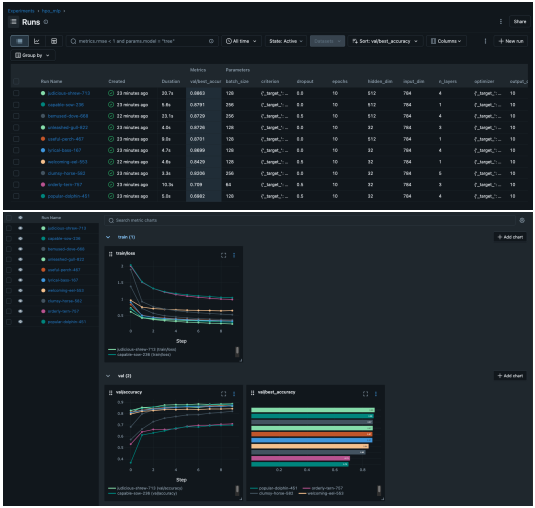
3. Un script qui utilise la configuration

```python
import hydra

@hydra.main(config_path="conf", config_name="config")
def func(cfg: DictConfig):
    # On fait un print des paramtres, par exemple
    print(f"The batch size is {cfg.batch_size}")
    print(f"The learning rate is {cfg.lr}")

if __name__ == "__main__":
    func()
```

3. Une semantique de modification simple

```
uv run script.py batch_size=32
```

MLflow



La clé de voûte :

- Logging explicite dans le code
- Gestion des artifacts
- Interface graphique intuitive

Lien vers la documentation de MLflow : https://mlflow.org/get-started

ZAHARIA et al. 2018

Example minimaliste d'utilisation de MLflow

```python
import mlflow

def training_loop():
    mlflow.set_experiment(exp_name) # instantiating mlflow
    with mlflow.start_run(run_id=id): # logging performance related to a run id
        ...
        mlflow.log_metrics(
                {"test/accuracy": float(accuracy),
                "test/macro_f1": float(f1_score), }

        # logging artifacts
        mlflow.log_figure(fig, "confusion_matrix.png")
```

TP

Installation

```
# For Unix based systems
$ curl -LsSf https://astral.sh/uv/install.sh | sh

# For Windows
$ powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

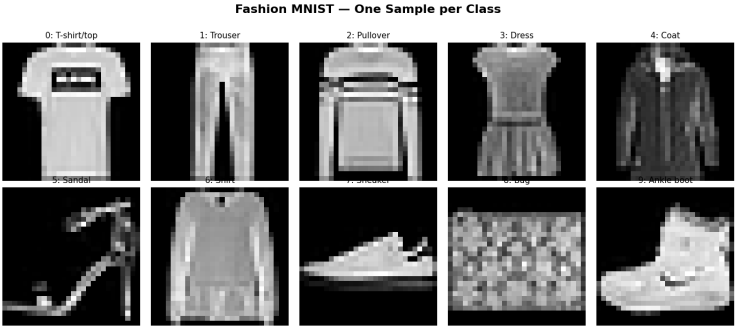
Quelques commandes pour appréhender uv :

1. `uv init`
2. `uv add {package_name}`
3. `uv remove {package_name}`
4. `uv sync`
5. `uv run {script.py}`

Structure du TP

1. Showcase projet complet d'apprentissage par Régression Logistique
2. Prise en main d'un projet avec un MLP et un CNN
3. TP compétitif sur Fashion-MNIST

Présentation du jeu de donnée Fashion-MNIST



10 classes ; 28x28x1 (grayscale : from 0 (black) to 255 (white)) images !

Lien vers le jeu de donnée Fashion-MNIST : Fashion MNIST

XIAO, RASUL et VOLLGRAF 2017

TP de prise de contact - Régression Logistique

Objectifs

1. Identifier les blocs structurants (architecture du projet) dans le code
2. Identifier le code nécessaire pour intégrer Hydra et MLFlow
3. Lancer plusieurs expériences depuis le terminal en utilisant la syntaxe d'Hydra
4. Lancer l'interface graphique de MLflow, explorer les expériences et effectuer des comparaisons

Définition Régression logistique

En supposant que la cible y_i prend des valeurs dans l'ensemble $\{0, 1\}$ pour le point de données i. Une fois ajustée, une RL prédit la probabilité $P(y_i = 1|X_i)$ de la classe positive comme :

$$\hat{p}(X_i) = \frac{1}{1 + \exp(-X_i w - w_0)}$$

Définition Régression logistique

En supposant que la cible y_i prend des valeurs dans l'ensemble $\{0, 1\}$ pour le point de données i. Une fois ajustée, une RL prédit la probabilité $P(y_i = 1|X_i)$ de la classe positive comme :

$$\hat{p}(X_i) = \frac{1}{1 + \exp(-X_i w - w_0)}$$

Problème d'optimisation

En tant que problème d'optimisation, la RL binaire avec un terme de régularisation $r(w)$ et une force de régularisation inverse C minimise la fonction de coût suivante :

$$\min_w \sum_{i=1}^{n} (-y_i \log(\hat{p}(X_i)) - (1 - y_i) \log(1 - \hat{p}(X_i))) + \frac{r(w)}{C}$$

TP prise en main - Régression Logistique

Le code pour l'exemple de la régression logistique est disponible ici :

Github

https://github.com/J-ally/Deep_repro_LR

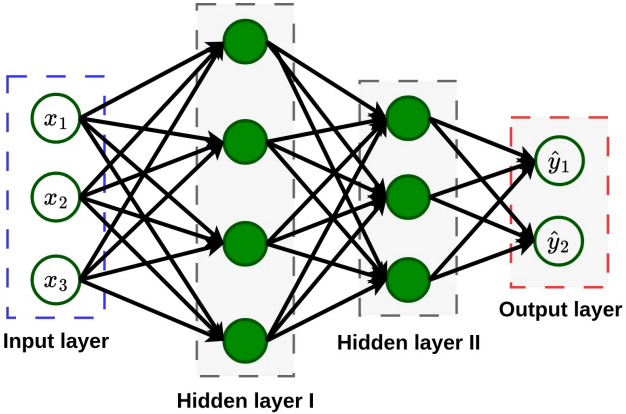
```
$ git clone https://github.com/J-ally/Deep_repro_LR.git
```

TP prise en main - Architectures neuronales

Objectifs

1. Compléter les bouts de code “##TODO##”
2. Lancer des expériences d’entraînement des modelés
3. Vérifier les expériences avec MLflow
   inspecter les artefacts d’entraînements et les indicateurs de performance
4. Évaluer les modèles entraînés

Perceptron multicouche (MLP)



Réseau neuronal convolutif (CNN)

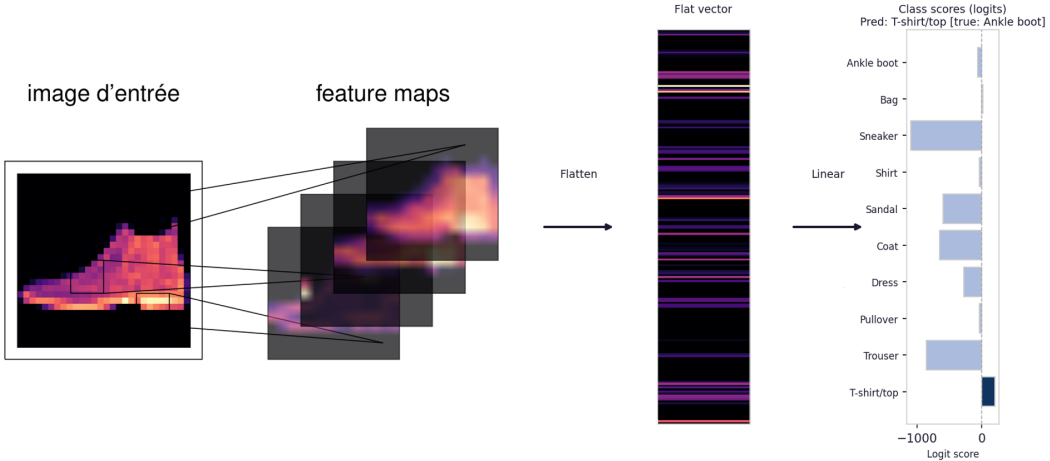


Illustration d'une convolution unique pour une prédiction multiclasse

Le code pour l'exemple des architectures neuronales MLP et CNN est disponible ici :

Github

https://github.com/J-ally/Deep_repro

```
$ git clone https://github.com/J-ally/Deep_repro.git
```

TP "compétitif"

Une fois la prise en main du code faite :

Jouer avec le code pour obtenir des meilleurs résultats sur la tâche !

L'entièreté des résultats sera loggé sur un mlflow commun qui permettra d'avoir en temps réél les performances des / participants.

TP "compétitif"

Une fois la prise en main du code faite :

Jouer avec le code pour obtenir des meilleurs résultats sur la tâche !

L'entièreté des résultats sera loggé sur un mlflow commun qui permettra d'avoir en temps réél les performances des / participants.

Obtenir des meilleurs performances :

- Changer les hyperparamètres
- Changer les paramètres d'entraînement
- Essayer d'autres optimiseurs
- Pour les plus courageux : proposer de nouveaux modèles

Merci

References I

- AKIBA, Takuya et al. (2019). “Optuna : A Next-generation Hyperparameter Optimization Framework”. In : *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- ARAVANTINOS, Vincent et Frederik DIEHL (mai 2019). *Traceability of Deep Neural Networks*. en. arXiv :1812.06744 [cs.LG]. DOI : 10.48550/arXiv.1812.06744. URL : http://arxiv.org/abs/1812.06744 (visité le 26/05/2026).
- BAKER, Monya (mai 2016). “1,500 scientists lift the lid on reproducibility”. In : *Nature* 533.7604, p. 452-454. ISSN : 1476-4687. DOI : 10.1038/533452a. URL : https://doi.org/10.1038/533452a.
- COLLABORATION, Open Science (2015). “Estimating the reproducibility of psychological science”. In : *Science* 349.6251, aac4716. DOI : 10.1126/science.aac4716. URL : https://www.science.org/doi/abs/10.1126/science.aac4716.
- SOHL-DICKSTEIN, Jascha (2024). “The boundary of neural network trainability is fractal”. In : *arXiv preprint arXiv :2402.06184*.

References II

- XIAO, Han, Kashif RASUL et Roland VOLLGRAF (2017). “Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms”. In : *arXiv preprint arXiv :1708.07747*.
- YADAN, Omry (2019). *Hydra - A framework for elegantly configuring complex applications*. Github. URL : https://github.com/facebookresearch/hydra.
- ZAHARIA, Matei A. et al. (2018). “Accelerating the Machine Learning Lifecycle with MLflow”. In : *IEEE Data Eng. Bull.* 41, p. 39-45. URL : https://api.semanticscholar.org/CorpusID:83459546.

Optuna

Optimisation d'hyperparamètres

- Hyperparamètres régissent le réseau de neurone lui-même
- Combinaisons d'hyperparamètres forment un espace de recherche complexe

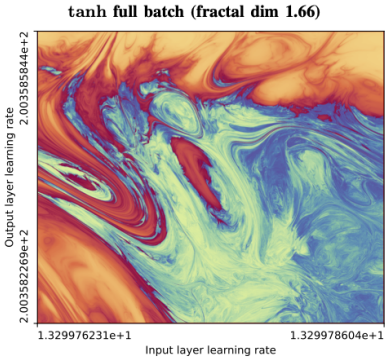


Figure – Tiré de SOHL-DICKSTEIN 2024 : Recherche par grille en deux dimensions sur les hyperparamètres

Optuna

- Recherche bayesienne d'hyperparamètres
- Recherche itérative avec pruning et early stopping
- TPE et CMA-ES

Lien vers la documentation d'optuna : https://optuna.org/

AKIBA et al. 2019