# Rapport TP Deep Learning
# De la conception au déploiement

Étudiant ENSPY

7 octobre 2025

## 1 Introduction

Ce rapport présente la réalisation du TP de Deep Learning portant sur la conception, l'entraînement et le déploiement d'un modèle de classification des chiffres manuscrits MNIST.

## 2 Partie 1 : Fondations du Deep Learning

### 2.1 Concepts Théoriques

**Question 1 : Différence entre descente de gradient classique et SGD**

La descente de gradient classique utilise l'ensemble du dataset pour calculer le gradient à chaque itération, ce qui est coûteux en mémoire et temps de calcul. La SGD (Stochastic Gradient Descent) utilise un échantillon aléatoire (batch) pour estimer le gradient, permettant des mises à jour plus fréquentes et une convergence plus rapide, particulièrement adaptée aux grands datasets du deep learning.

**Rétropropagation du gradient**

La rétropropagation calcule les gradients des poids en propageant l'erreur de la sortie vers l'entrée du réseau, utilisant la règle de dérivation en chaîne pour ajuster chaque poids proportionnellement à sa contribution à l'erreur finale.

### 2.2 Exercice 1 : Construction du réseau de neurones

**Question 1 : Utilité des couches Dense et Dropout**

Les couches Dense (fully-connected) connectent chaque neurone à tous les neurones de la couche suivante, permettant l'apprentissage de relations complexes. La couche Dropout désactive aléatoirement des neurones pendant l'entraînement pour éviter le surapprentissage. La fonction softmax normalise les sorties en probabilités pour la classification multi-classes.

**Question 2 : Optimiseur Adam**

Adam combine les avantages de RMSprop et Momentum en adaptant le taux d'apprentissage pour chaque paramètre individuellement et en utilisant des moyennes mobiles des gradients et de leurs carrés, offrant une convergence plus stable et rapide que la SGD simple.

**Question 3 : Vectorisation et calculs par lots**

La vectorisation permet de traiter plusieurs échantillons simultanément via des opérations matricielles optimisées. Le paramètre `batch_size=128` traite 128 images à la fois, exploitant le parallélisme des GPU et améliorant l'efficacité computationnelle.

# 3 Partie 2 : Ingénierie du Deep Learning

## 3.1 Versionnement avec Git

Le projet a été versionné avec Git, permettant le suivi des modifications et la collaboration. Les commandes utilisées :

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <URL>
git push -u origin master
```

## 3.2 Suivi avec MLflow

MLflow a été intégré pour tracer les paramètres (epochs, batch_size, dropout_rate) et métriques (test_accuracy) de chaque expérimentation, facilitant la comparaison des modèles.

## 3.3 Conteneurisation Docker

L'application Flask a été conteneurisée avec Docker, incluant :
— Image de base Python 3.9-slim
— Installation des dépendances
— Exposition du port 5000
— API REST pour les prédictions

## 3.4 Déploiement et CI/CD

**Question 1 : Pipeline CI/CD**
Un pipeline GitHub Actions pourrait automatiser :

1. Construction de l'image Docker à chaque push
2. Tests automatisés du modèle
3. Déploiement sur Google Cloud Run ou AWS ECS
4. Mise à jour automatique en production

**Question 2 : Indicateurs de monitoring**
Trois types d'indicateurs clés :

1. **Performance** : Latence des prédictions, throughput, temps de réponse
2. **Qualité** : Accuracy en production, distribution des prédictions, détection de drift
3. **Infrastructure** : Utilisation CPU/mémoire, disponibilité du service, erreurs HTTP

# 4 Conclusion

Ce TP a permis de maîtriser le cycle complet d'un projet de Deep Learning, de la conception du modèle jusqu'au déploiement en production, en intégrant les bonnes pratiques d'ingénierie logicielle (versionnement, conteneurisation, monitoring).