New

Deep Learning avec PyTorch

Concevoir, entraîner, optimiser et déployer des réseaux de neurones avec PyTorch pour développer des applications d'intelligence artificielle

Présentiel ou en classe à distance

3 jours (21 h)

Prix inter : 2.490,00 € HT
Forfait intra : 7.690,00 € HT

Réf.: IA420

La formation **Deep Learning avec PyTorch** permet de concevoir, entraîner, optimiser et déployer des **modèles de deep learning** avec **PyTorch**. Elle couvre les réseaux de neurones, les **CNN**, les **LSTM**, les **Transformers**, l'optimisation des architectures, ainsi que l'industrialisation des modèles avec **TorchServe**, Docker et la surveillance en production. Une formation complète pour développer des applications d'intelligence artificielle performantes et les déployer dans un environnement professionnel.

A qui s'adresse cette formation ?

Pour qui

- Data scientists, AI engineers

Prérequis

- Connaissances en Python, en Machine Learning et en mathématiques
- Disposez-vous des connaissances nécessaires pour suivre cette formation ? Testez-vous !

Programme

1 - Fondamentaux de PyTorch

- Découvrir l'écosystème PyTorch, son positionnement, ses cas d'usage et l'organisation de ses modules
- Créer et manipuler des tenseurs PyTorch : types, formes et opérations vectorisées
- Exécuter des calculs sur CPU et GPU, transférer les données et gérer les périphériques
- Comprendre le moteur autograd, le graphe de calcul dynamique et le calcul automatique des gradients
- Appliquer la descente de gradient à une régression simple

Atelier

Créer et manipuler des tenseurs : indexation, redimensionnement et opérations

Exécuter des opérations sur GPU et comparer les performances avec le CPU

Calculer des gradients avec autograd sur des expressions simples

Mettre en oeuvre manuellement une descente de gradient sur une régression linéaire

2 - Architecture des réseaux de neurones avec PyTorch

- Utiliser le module torch.nn : couches, fonctions d'activation et conteneurs
- Définir un modèle avec l'approche nn.Module et la méthode forward
- Concevoir l'architecture d'un réseau de neurones entièrement connecté

- Choisir les fonctions de perte selon la nature du problème
- Initialiser les poids et appliquer les bonnes pratiques de conception

Atelier

Définir une architecture dense à l'aide d'une classe nn.Module

Sélectionner les fonctions d'activation et la fonction de perte adaptées

Instancier le modèle et examiner sa structure et ses paramètres

Réaliser une passe avant (forward) sur un lot de données de test

3 - Entraîner un modèle de Deep Learning avec PyTorch

- Utiliser Dataset et DataLoader pour charger, regrouper et mélanger les données
- Comprendre la boucle d'entraînement PyTorch : passe avant, calcul de la perte et rétropropagation
- Utiliser les optimiseurs de torch.optim, notamment SGD et Adam, et régler le taux d'apprentissage
- Séparer les données d'entraînement et de validation et suivre les métriques
- Sauvegarder et recharger un modèle avec state_dict

Atelier

Préparer les données avec Dataset et DataLoader

Écrire une boucle d'entraînement complète avec un optimiseur et une fonction de perte

Suivre la perte et les métriques sur les jeux d'entraînement et de validation

Sauvegarder le modèle entraîné et le recharger pour vérifier son fonctionnement

4 - Concevoir des modèles CNN pour la classification d'images

- Comprendre le principe de la convolution : filtres, cartes de caractéristiques et pooling
- Concevoir l'architecture d'un réseau de neurones convolutif (CNN)
- Prétraiter et augmenter les données avec torchvision
- Appliquer le transfert d'apprentissage à partir de modèles préentraînés
- Réaliser le réglage fin (fine-tuning) d'un modèle existant

Atelier

Préparer un jeu d'images avec torchvision et l'augmentation de données

Construire et entraîner un CNN pour une tâche de classification

Appliquer le transfert d'apprentissage à partir d'un modèle préentraîné

Comparer les performances d'un modèle entraîné à partir de zéro avec celles d'un modèle affiné

5 - Réseaux LSTM et GRU avec Keras

- Comprendre les enjeux et les modes de représentation des données séquentielles
- Concevoir des réseaux de neurones récurrents avec Keras, LSTM et GRU
- Comprendre les mécanismes de mémoire et la succession des blocs LSTM

Atelier

Préparer un jeu de données pour la prédiction de séries temporelles

Transformer les données séquentielles pour l'entraînement

Construire et entraîner un réseau récurrent LSTM ou GRU

6 - Architecture Transformer et mécanisme d'attention

- Comprendre le mécanisme d'attention et l'architecture
- Transformer Exploiter des modèles Transformers préentraînés avec KerasNLP
- Choisir une architecture selon la nature de la tâche

Atelier

Utiliser un modèle Transformer préentraîné pour une tâche séquentielle

Configurer les hyperparamètres du modèle

Comparer les résultats des approches LSTM et Transformers

7 - Améliorer les performances des modèles de Deep Learning

- Évaluer un modèle à l'aide de métriques et d'une matrice de confusion
- Diagnostiquer le surapprentissage et le sous-apprentissage
- Appliquer des techniques de régularisation : dropout, weight decay et early stopping
- Identifier les problèmes de gradients : disparition, explosion et normalisation par lots
- Optimiser les performances grâce au réglage de la taille des lots, à la précision mixte et au profilage
- Suivre les expériences avec TensorBoard
  Atelier

  Analyser les courbes d'apprentissage et identifier le surapprentissage

  Appliquer des techniques de régularisation et mesurer leurs effets

  Activer la précision mixte et profiler l'entraînement pour améliorer sa vitesse

  Visualiser l'entraînement et les métriques avec TensorBoard

8 - Préparer un modèle PyTorch pour la production

- Passer du mode entraînement au mode inférence en appliquant les bonnes pratiques
- Exporter un modèle avec TorchScript et torch.export
- Utiliser le format ONNX pour assurer l'interopérabilité et l'exécution multiplateforme
- Optimiser l'inférence grâce à la quantification et à la réduction de la taille du modèle
- Mesurer la latence et le débit d'un modèle
  Atelier

  Exporter un modèle entraîné avec TorchScript

  Convertir le modèle au format ONNX et vérifier l'équivalence des résultats

  Appliquer la quantification et mesurer les gains de taille et de latence

  Comparer les performances d'inférence avant et après l'optimisation

9 - Déployer un modèle PyTorch en production

- Comparer les architectures de déploiement : inférence par lots et en temps réel
- Servir un modèle avec TorchServe et comprendre sa configuration
- Exposer un modèle à travers une API d'inférence
- Conteneuriser le service avec Docker Appliquer les bonnes pratiques de mise en production d'un modèle de Deep Learning
  Atelier

  Empaqueter un modèle et le servir avec TorchServe

  Exposer le modèle à travers une API et tester des requêtes d'inférence

  Conteneuriser le service d'inférence avec Docker

  Réaliser un test de charge et mesurer la latence et le débit

10 - Suivre les performances et la qualité des modèles

- Comprendre les enjeux de la surveillance d'un modèle de Deep Learning en production
- Suivre les indicateurs techniques : latence, débit, taux d'erreur et utilisation des ressources
- Surveiller la qualité du modèle et détecter la dérive des données et des prédictions
- Journaliser les prédictions et mettre en place des alertes Définir des stratégies de mise à jour et de réentraînement des modèles
  Atelier

  Instrumenter le service d'inférence pour collecter la latence et le taux d'erreur

  Journaliser les prédictions et les données d'entrée

  Détecter une dérive des données dans un flux de prédictions

  Définir des seuils d'alerte et une stratégie de réentraînement

Les objectifs de la formation

- Concevoir et entraîner des réseaux de neurones adaptés à différents cas d'usage
- Évaluer et optimiser les performances de modèles de Deep Learning
- Déboguer et optimiser les architectures de réseaux de neurones.
- Déployer des modèles de Deep Learning en environnement de production

Evaluation

- Pendant la formation, le formateur évalue la progression pédagogique des participants via des QCM, des mises en situation et des travaux pratiques. Les participants passent un test de positionnement avant et après la formation pour valider leurs compétences acquises.

Les points forts de la formation

- Une formation couvrant l'ensemble du cycle de vie d'un modèle de Deep Learning, de sa conception à son déploiement et à son exploitation en production
- Des ateliers pratiques s'appuyant sur PyTorch et son écosystème (TorchVision, TorchServe, ONNX et TensorBoard) pour concevoir, entraîner et optimiser des modèles sur des cas d'usage concrets
- Un volet dédié au déploiement opérationnel des modèles, permettant d'acquérir les bonnes pratiques de mise en production et de gagner en autonomie dans un contexte professionnel

Dates et villes 2026 - Référence IA420

Dernières places disponibles Session garantie

A distance

du 4 nov. au 6 nov.	du 7 déc. au 9 déc.

Paris

du 4 nov. au 6 nov.	du 7 déc. au 9 déc.