

Projet 01 : Deep Learning Détection de globules blanc

Objectif : “Développement d'un modèle de réseau de neurones pour la détection de globules blancs dans une image microscopique de frottis sanguin”.

Préparation de l'environnement pour le projet

> Deep Learning Détection de globules blanc.

- ❶ Téléchargez le fichier projet ZIP via les Ressources Udemmy (**section #02**) , et décompressez-le.
- ❷ Lancez VSCode, puis ajouter le répertoire projet décompressé à VSCode
- ❸ Lancez un terminal, puis créez un environnement virtuel : `python -m venv env_deeplearning`
- ❹ Autorisez l'exécution de Scripts sous Windows : `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser`
- ❺ Activez l'environnement virtuel: `.\env_deeplearning\Scripts\Activate.ps1`
- ❻ Installez les librairies du projet : `pip install -r requirements.txt`

Votre Environnement est Prêt ! 

Plan projet 01 deep learning

- #00 Formation Express Python
- #01 Annotation des Images
- #02 Création des Masques
- #03 Développement du Modèle CNN
- #04 Évaluation et Tests
- #05 Application Desktop Deep Learning
- #06 BONUS : Utilisation de TensorBoard

00 - Formation Express Python

Pour débiter, nous avons une **formation express** sur Python.

- **Objectif** : Comprendre les bases du langage, les structures de données (listes, dictionnaires, etc.), la programmation orientée objet (POO), et les fondamentaux du fonctionnement général de Python.
- **Compétence clé** : Acquérir les bases nécessaires pour être autonome avec Python.
- **Notebook associé** : [00_Formation_Express_Python.ipynb](#).

01 - Annotation des images

Nous entrerons ensuite dans la première étape pratique : **l'annotation des images microscopiques**, où il s'agit d'identifier les zones contenant des globules blancs.

- **Objectif** : Comprendre le processus d'annotation d'images et apprendre à manipuler des données étiquetées.
- **Compétence clé** : Préparer vos données pour un projet de Deep Learning.
- **Notebook associé** : [01_Annotation_images.ipynb](#).

02 - Création des masques

Une fois les images annotées, nous allons générer des **masques binaires** pour la segmentation des globules blancs.

- **Objectif** : Apprendre à créer des masques précis à partir des annotations.
- **Compétence clé** : Maîtriser la segmentation d'images pour un projet de vision par ordinateur.
- **Notebook associé** : [02_Creation_masques.ipynb](#).

03 - Développement du modèle CNN

Nous passerons à l'étape la plus technique : **le développement d'un modèle de réseau de neurones convolutifs (CNN)**.

- **Objectif** : Construire et entraîner un modèle CNN, analyser les courbes de perte et de précision en temps réel, et ajuster les hyperparamètres pour optimiser les performances.
- **Compétence clé** : Comprendre les concepts fondamentaux des CNN et maîtriser leur mise en œuvre.
- **Notebook associé** : [03_WBC_model_CNN.ipynb](#).

04 - Evaluation & Tests

Une fois le modèle entraîné, nous l'évaluerons sur des données de validation.

- **Objectif** : Vérifier ses performances et effectuer les ajustements nécessaires pour maximiser son efficacité.
- **Compétence clé** : Tester un modèle sur des jeux de données inconnus et analyser les résultats obtenus.
- **Notebook associé** : [04_Evaluation_model.ipynb](#).

05 - Application Desktop Deep Learning

Enfin, nous intégrerons le modèle dans une **application desktop** pour rendre son utilisation accessible à un utilisateur final.

- **Objectif** : Développer une interface graphique fluide et fonctionnelle.
- **Compétence clé** : Intégrer un modèle de Deep Learning dans un environnement utilisateur.
- **Notebook associé** : [05_Application_WBC.ipynb](#).

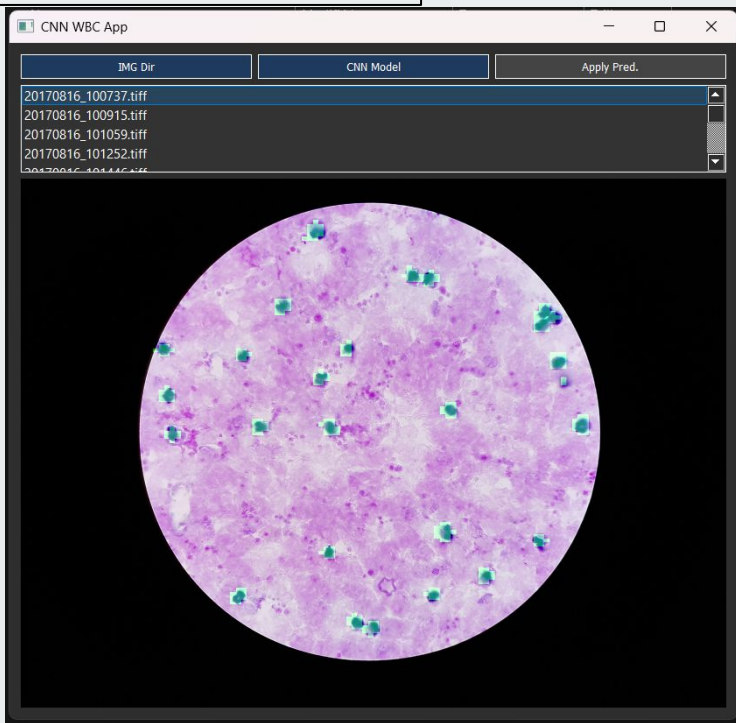
06 - BONUS - Visualisation avec Tensorboard

Pour finir, nous découvrirons comment utiliser **TensorBoard** pour visualiser les courbes d'entraînement et les performances du modèle.

- **Objectif** : Explorer les outils de visualisation avancés pour analyser les métriques de votre modèle.
- **Compétence clé** : Apprendre à interpréter les courbes d'entraînement et les graphiques de performances.
- **Notebook associé** : [06_BONUS_Tensorboard.ipynb](#).

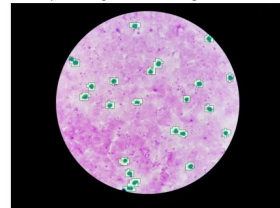
#01 Deep Learning et Intelligence Artificielle: “Détection de Globules Blanc en Médecine”.

Application de bureau

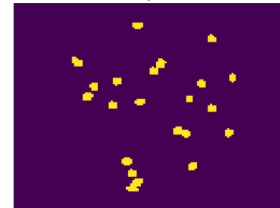


Entraînement Réseau de neurones

Deep Learning : Détection de globules blanc

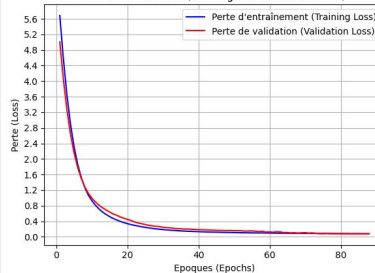


Binary Mask

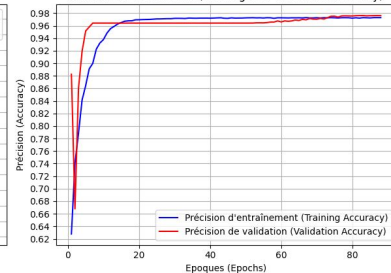


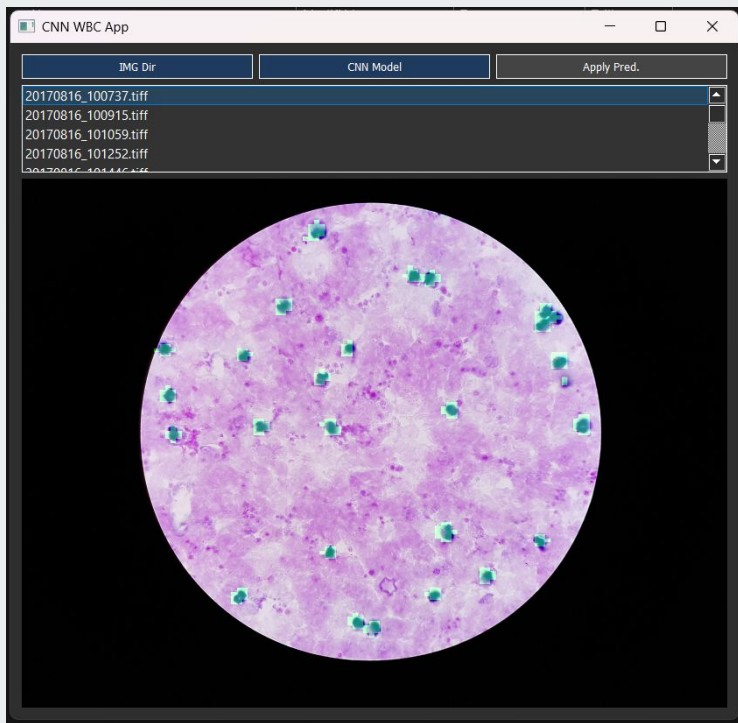
Modèle: V1.0 - Test Perte (Loss): 7.28, Test Précision (Accuracy): 97.62

Courbe de la Perte (Training and Validation Loss)



Courbes de la Précision (Training and Validation Accuracy)







Réseau convolutionnel (CNN) : Algorithme basé sur des réseaux de neurones, utilisant des filtres (matrices) pour extraire automatiquement les traits remarquables d'une image, tels que les contours, textures et motifs.

Jupyter Notebook : Environnement interactif de programmation permettant d'écrire et d'exécuter du code Python avec visualisation des résultats.

TensorFlow : Bibliothèque open-source pour le calcul numérique et le développement de modèles d'apprentissage profond.

Environnement virtuel Python : Espace isolé pour installer des bibliothèques spécifiques à un projet sans interférer avec d'autres projets.

Descente de Gradient : C'est une méthode qui permet à un modèle d'apprendre en ajustant progressivement ses paramètres. À chaque étape, le modèle fait un petit pas dans la direction qui réduit l'erreur, jusqu'à ce qu'il trouve les meilleurs paramètres pour faire des prédictions précises.

Filtre: Matrice, souvent de taille 3x3, avec des valeurs spécifiques, appliquée à une image pour faire ressortir des propriétés remarquables, comme les bords ou textures.

Fonction d'activation (Activation Function) : Fonction appliquée à la sortie d'un neurone pour introduire de la non-linéarité dans le modèle.

Apprentissage supervisé (Supervised Learning) : Méthode d'apprentissage où le modèle est entraîné à partir de données étiquetées.

Overfitting : Phénomène où un modèle s'adapte trop aux données d'entraînement et généralise mal sur les nouvelles données.

Batch Size : Nombre d'échantillons traités ensemble avant la mise à jour des paramètres du modèle.

Epoch : Une passe complète sur l'ensemble des données d'entraînement durant l'entraînement d'un modèle.

Optimiseur (Optimizer) : Algorithme utilisé pour minimiser la fonction de perte et ajuster les poids du réseau.

Fonction de perte (Loss Function) : Mesure utilisée pour évaluer l'erreur entre les prédictions du modèle et les véritables valeurs.