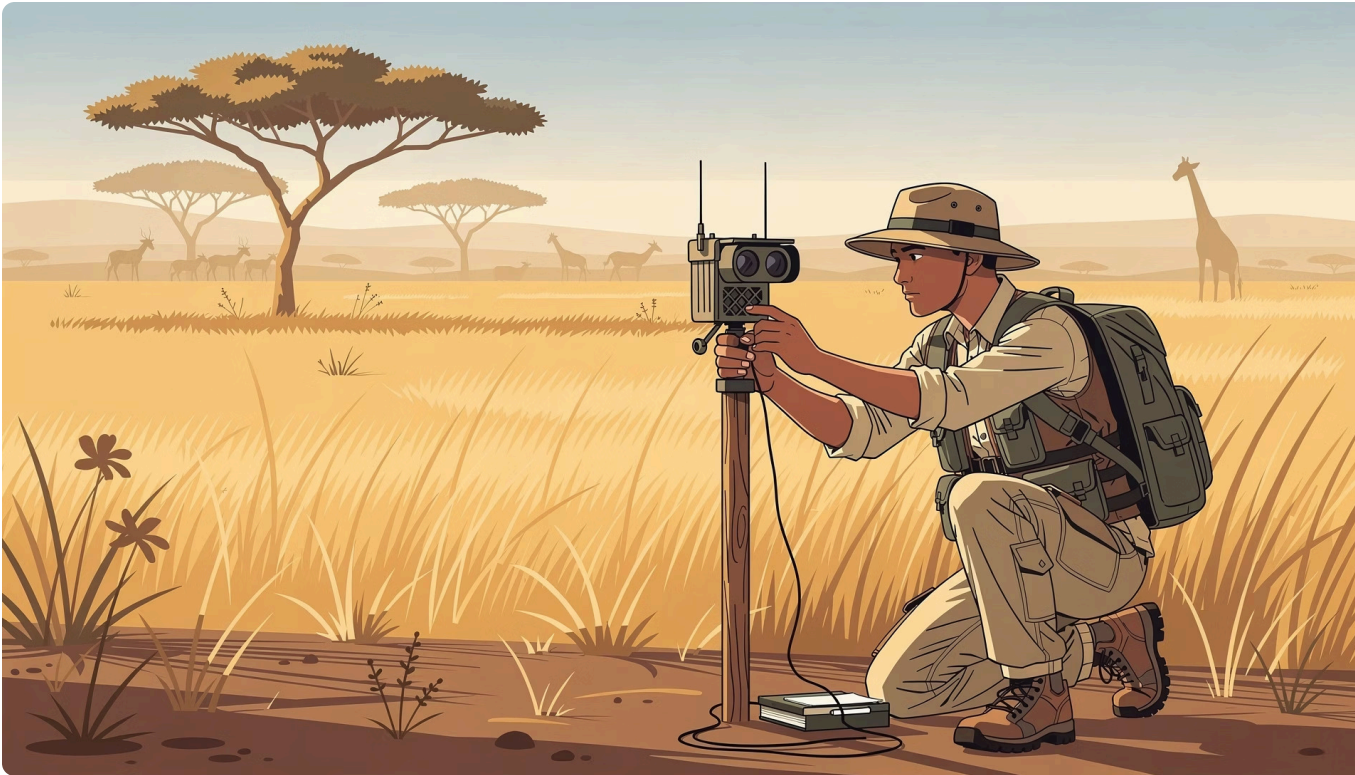




# TP : Reconnaissance d'animaux avec Python et réseaux de neurones

Un projet pratique pour découvrir l'intelligence artificielle appliquée à la vision par ordinateur

# Pourquoi reconnaître des animaux avec l'IA ?



## Applications concrètes dans la conservation

La reconnaissance automatique d'animaux révolutionne le suivi de la biodiversité et la protection des espèces menacées. Elle permet d'analyser rapidement des milliers d'images issues de caméras de surveillance écologique.

- Identification automatique d'éléphants, girafes et tigres sur le terrain
- Automatisation d'un travail long et fastidieux pour les biologistes
- Amélioration de la précision et gain de temps considérable

# Introduction aux réseaux de neurones convolutionnels (CNN : *Convolutional Neural Network*)

## Architecture inspirée du cerveau

Les CNN reproduisent le fonctionnement du cortex visuel humain pour analyser et interpréter les images de manière automatique et hiérarchique.

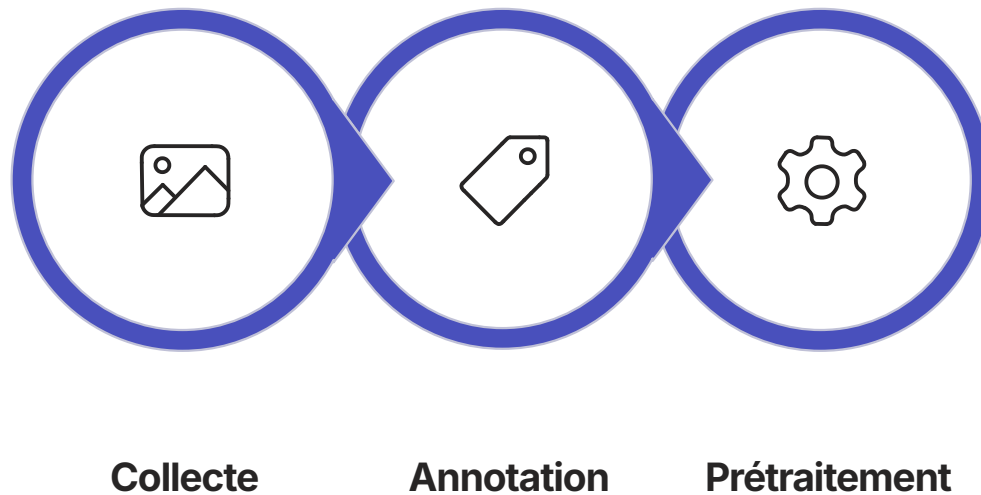
## Extraction automatique

Le réseau identifie automatiquement les caractéristiques pertinentes : formes, textures, motifs, contours et structures complexes des animaux.

## Performance supérieure

Les CNN surpassent largement les méthodes classiques de traitement d'image pour la reconnaissance visuelle et la classification d'objets.

# Collecte et préparation des données



La qualité des données détermine la performance finale de votre modèle. Prenez le temps de bien préparer cette étape fondamentale.

## Sources de données

Utilisez des plateformes comme Kaggle ou Wildlife RE-ID pour obtenir des images d'éléphants, girafes et tigres déjà annotées.

## Annotation et étiquetage

Chaque image doit être clairement étiquetée avec sa classe correspondante pour l'apprentissage supervisé.

## Prétraitement essentiel

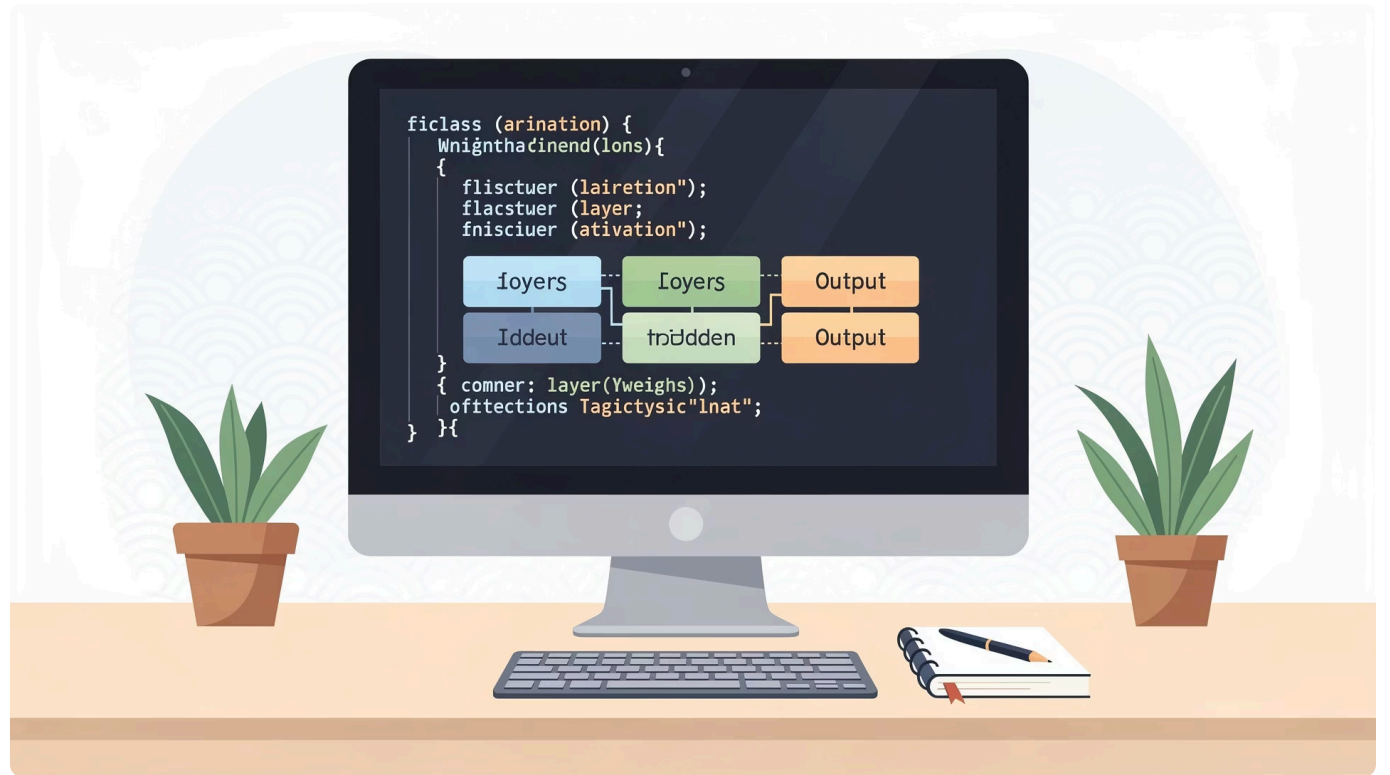
- Redimensionnement uniforme des images
- Normalisation des valeurs de pixels
- Augmentation : rotation, zoom, retournement

# Construction du modèle CNN en Python

## Bibliothèques recommandées

TensorFlow/Keras et PyTorch sont les frameworks les plus populaires pour créer des réseaux de neurones. Keras offre une API simple et intuitive, idéale pour débiter.

## Architecture du réseau



01

## Couches convolutionnelles

Extraction des caractéristiques visuelles

03

## Couches denses

Classification finale

02

## Couches de pooling

Réduction de la dimensionnalité

04

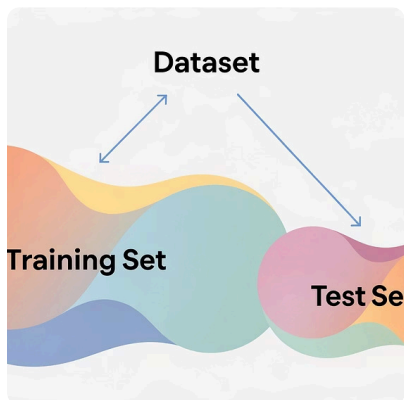
## Sortie softmax

Probabilités pour chaque classe

- ❏ Paramètres clés : taille des filtres (3x3 ou 5x5), nombre de couches (commencer avec 3-4), fonction d'activation ReLU pour la non-linéarité

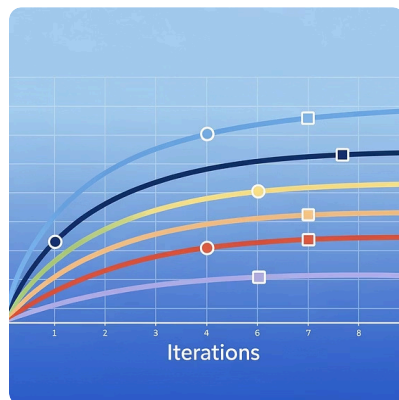
## ÉTAPE 3

# Entraînement et validation du modèle



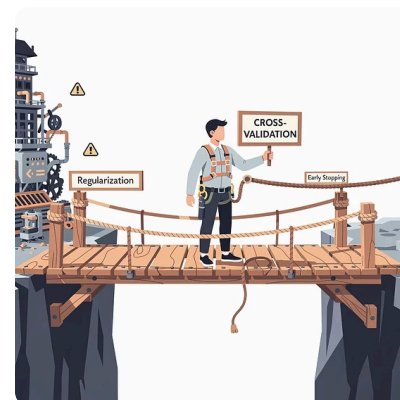
## Séparation des données

Divisez vos images en ensembles d'entraînement (70-80%) et de test (20-30%) pour évaluer objectivement les performances.



## Optimisation

Choisissez un optimiseur (SGD, Adam) et une fonction de perte adaptée comme l'entropie croisée pour la classification multi-classes.



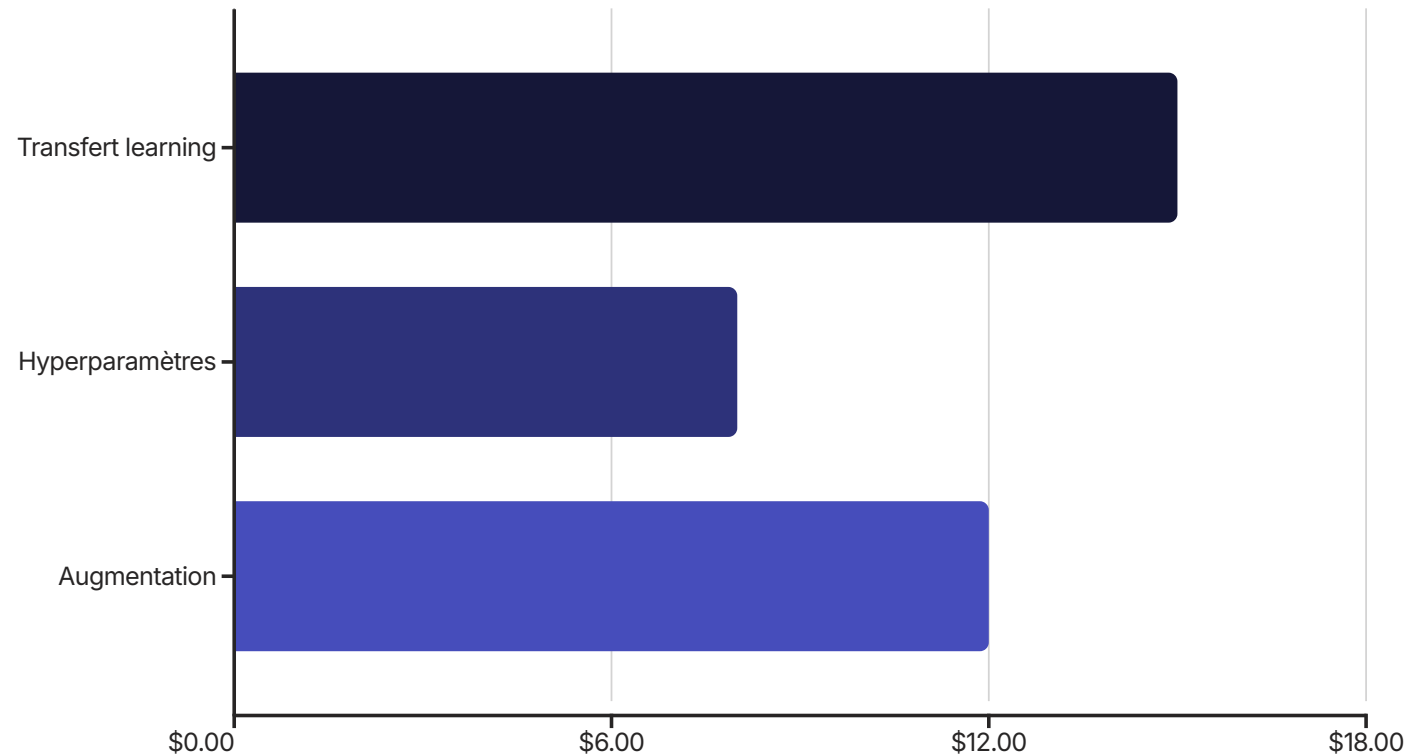
## Éviter le surapprentissage

Utilisez la validation croisée, l'augmentation des données et le dropout pour garantir la généralisation du modèle.

# Évaluation et amélioration

## Métriques de performance

Analysez les résultats de votre modèle avec des outils statistiques précis pour identifier ses forces et faiblesses.



*Impact estimé des différentes techniques d'amélioration sur la performance du modèle (en points de pourcentage)*

### Précision globale

Taux de prédictions correctes

### Rappel par classe

Capacité à détecter chaque animal

### Matrice de confusion

Visualisation des erreurs

# Exemples inspirants de projets similaires

## Projet Romain Gallerne

Reconnaissance de tigres, éléphants et renards utilisant CNN et GAN pour générer des images synthétiques d'entraînement.

## Initiative DeepFaune

Plus d'un million d'images annotées collectées pour créer des modèles de reconnaissance automatique d'espèces en milieu naturel.

## Résultats impressionnants

Les modèles récents atteignent une précision de 80% à 90% selon la qualité des données et l'architecture choisie.



# Défis et bonnes pratiques à garder en tête

1

## Qualité des données

La diversité et la qualité de votre dataset sont essentielles. Privilégiez des images variées en termes d'angles, d'éclairage et de contexte.

2

## Déséquilibre des classes

Si vous avez peu d'images de tigres comparé aux éléphants, utilisez des techniques de rééquilibrage ou d'augmentation ciblée.

3

## Limites du modèle

L'IA ne remplace pas l'expertise humaine. Les cas ambigus nécessitent toujours une validation par un spécialiste.

4

## Interprétabilité

Documentez vos choix et analysez les erreurs pour comprendre le comportement de votre réseau de neurones.

# Votre mission pour ce TP



## Collecter et préparer

Rassemblez des images d'éléphants, girafes et tigres



## Créer le modèle

Construisez votre CNN en Python avec TensorFlow/Keras



## Entraîner et évaluer

Optimisez et testez votre réseau de neurones



## Documenter

Rédigez un rapport clair sur votre démarche

**Objectif final :** Un programme Python fonctionnel capable de reconnaître automatiquement ces trois animaux avec une précision mesurable la validation nécessite une reconnaissance à plus de 66% , accompagné d'une analyse détaillée de vos résultats et propositions d'amélioration.

[Commencer le TP](#)

[Ressources complémentaires](#)

