# Sujet 1 : Détection précoce de maladies des plantes

**Objectif**
Développer un modèle capable de reconnaître automatiquement différentes maladies à partir de photos de feuilles de plantes.

**Données :** Dataset : PlantVillage ou PlantDoc

**Outils recommandés**

- PyTorch / TensorFlow
- Modèles : CNN simples (ResNet, EfficientNet), ou vision transformers
- Librairies : OpenCV pour pré-traitement

**Étapes**

1. Prétraitement des images (taille, contraste, suppression du bruit).
2. Entraînement d'un modèle de classification.
3. Évaluation : précision, rappel, matrice de confusion.
4. Déploiement simple : prototype d'application web/mobile (Streamlit ou Flask).

**Livrables**

- Rapport (méthodologie + résultats).
- Notebook Python bien documenté.
- Démo (mini-application ou script).

**Impact attendu**
Aider les agriculteurs à détecter les maladies tôt, réduire les pertes de production.

# Sujet 2 : Analyse d’images médicales pour la prévention de maladies

**Objectif**
Concevoir un système de classification pour détecter automatiquement des anomalies sur des images médicales (par ex. cancer de la peau).

**Données :** Dataset : ISIC Skin Cancer Dataset

**Outils recommandés**

- PyTorch / TensorFlow
- Modèles : ResNet, InceptionV3, EfficientNet
- Techniques : Data augmentation pour améliorer la robustesse

**Étapes**

1. Chargement et nettoyage du dataset.
2. Entraînement d’un modèle pour classer « bénin » vs « malin ».
3. Comparaison de plusieurs architectures.
4. Déploiement d’un outil d’aide au diagnostic (ex : interface simple avec upload d’image).

**Livrables**

- Rapport détaillé avec métriques médicales (AUC, précision, sensibilité).
- Code reproductible.
- Démo (appli Streamlit pour tester).

**Impact attendu**
Contribuer à un dépistage plus précoce dans les régions à faible accès médical.

# Sujet 3 : Surveillance de la pollution et des déchets plastiques

**Objectif**
Créer un modèle de détection pour identifier automatiquement les déchets plastiques (bouteilles, sacs, etc.) dans les images ou vidéos.

**Données :** Dataset : TrashNet ou vidéos annotées.

**Outils recommandés**

- YOLOv8 / Faster R-CNN
- OpenCV pour le traitement vidéo
- PyTorch / Ultralytics

**Étapes**

1. Collecte et annotation (si nécessaire) d'images de déchets.
2. Entraînement d'un détecteur d'objets.
3. Test sur vidéos de rivières ou plages.
4. Résultats visuels : bounding boxes et comptage.
5. Optionnel : tableau de bord (Streamlit/Dash) pour visualiser les résultats.

**Livrables**

- Rapport d'analyse et métriques de détection.
- Code + modèle entraîné.
- Démonstration vidéo avec détection en direct.

**Impact attendu**
Aider ONG, municipalités ou associations à surveiller la pollution plastique.

# Sujet 4 : Accessibilité pour personnes malvoyantes

**Objectif**
Développer un système de reconnaissance d'objets du quotidien et fournir une sortie vocale pour les décrire.

**Données :** Dataset : COCO, Open Images Dataset

**Outils recommandés**

- YOLOv8 / Detectron2
- Text-to-Speech (gTTS, pyttsx3)
- OpenCV pour caméra en temps réel

**Étapes**

1. Entraîner ou utiliser un modèle pré-entraîné de détection.
2. Détecter des objets de base (chaise, table, feu rouge, billets, etc.).
3. Générer une description vocale de l'objet détecté.
4. Démo : caméra en temps réel + retour audio.

**Livrables**

- Rapport (approche, résultats, limites).
- Code complet avec instructions d'installation.
- Démo fonctionnelle sur webcam.

**Impact attendu**
Améliorer l'autonomie et la sécurité des personnes malvoyantes.

# Sujet 5 : Surveillance intelligente des foules et modélisation des comportements collectifs

### Contexte et motivation

La gestion des foules (événements sportifs, festivals, lieux publics, manifestations) est un enjeu majeur pour la sécurité et la prévention des accidents (mouvements de panique, bousculades, engorgements).
Les caméras de surveillance permettent de capter en temps réel le flux des individus, mais il reste difficile de **comprendre et prédire les comportements collectifs**.

Les approches récentes combinent :

- **Vision par ordinateur** pour détecter et suivre les individus dans la foule.
- **Réseaux complexes** pour modéliser les interactions entre personnes (ex. graphe de proximité, graphes dynamiques).
- **Machine Learning** pour classer ou prédire des situations à risque (attroupements soudains, propagation de panique).
- **Systèmes multi-agents** pour simuler et valider différents scénarios de comportements collectifs.

### Problématique

Comment développer un système intelligent capable de :

1. Détecter et suivre les individus dans une foule à partir de vidéos en temps réel.
2. Construire un graphe dynamique représentant les interactions sociales (réseaux complexes).
3. Utiliser le machine learning pour prédire les comportements collectifs et identifier des situations dangereuses.
4. Simuler les dynamiques de la foule avec une approche multi-agents afin de tester des stratégies de gestion (ex. évacuation).

### Objectifs du projet

- **Objectif principal** : Concevoir un prototype intégrant vision par ordinateur, analyse de graphes et simulation multi-agents pour la détection et la prévention de comportements collectifs à risque.
- **Objectifs secondaires** :
    - Développer un module de suivi de personnes (tracking vidéo).
    - Construire un graphe dynamique des interactions (distances, regroupements).
    - Appliquer des algorithmes de ML (SVM, Random Forest, ou DL) pour la détection de comportements anormaux.
    - Simuler différents scénarios avec un SMA (agents représentant des individus avec des règles comportementales).

### Données et outils prévus

- **Données** : datasets publics de vidéos de foules (UCY crowd dataset, ETH dataset, PETS dataset).
- **Outils** : Python, PyTorch/TensorFlow, OpenCV, NetworkX (analyse de graphes), Mesa ou JADE (SMA).

**Méthodologie prévue**

1. Détection et suivi des individus dans les vidéos.
2. Construction du graphe dynamique des interactions.
3. Classification et détection de comportements anormaux via ML.
4. Simulation multi-agents pour tester différents scénarios (panique, évacuation).
5. Évaluation des performances et visualisation des résultats.

**Livrables attendus**

- Code source documenté.
- Rapport scientifique (~25 pages).
- Prototype de simulation multi-agents + détection vidéo.
- Visualisations graphiques (réseaux dynamiques, heatmaps).
- Soutenance avec démonstration.

**Planning projet (10 semaines)**

**Semaine 1 : Introduction & cadrage**

- Présentation du sujet de recherche choisi.
- Revue de littérature rapide (articles, projets similaires).
- Définition des objectifs précis (ex : classification, détection, temps réel).
- Organisation de l'équipe et choix des outils (PyTorch, TensorFlow, OpenCV, YOLO, etc.).

**Livrable** : Document d'intention (2 pages) avec objectifs, état de l'art, planning de groupe.

**Semaine 2 : Préparation des données, prétraitement et exploration**

- Téléchargement du dataset choisi (PlantVillage, ISIC, TrashNet, COCO…).
- Nettoyage : suppression doublons, renommage, organisation des dossiers.
- Annotation supplémentaire si besoin (LabelImg, Roboflow).
- Analyse des données (dimensions, nombre de classes, équilibre).
- Visualisation des exemples.
- Prétraitements (redimensionnement, normalisation, data augmentation).

**Livrable** :

- Dataset organisé + notebook de préparation.
- Rapport court (2-3 pages) sur la qualité du dataset + code de prétraitement.

**Semaine 3 : Modèle de base (baseline) & Améliorations**

- Implémentation d'un premier modèle simple (CNN classique ou modèle pré-entraîné).
- Évaluation avec métriques de base (accuracy, précision, rappel, F1-score).
- Sauvegarde du modèle entraîné.
- Test de modèles plus avancés (ResNet, EfficientNet, Vision Transformers, YOLOv8 selon le sujet).
- Comparaison des performances avec baseline.
- Visualisation des résultats (matrice de confusion, exemples bien/mal classés).

**Livrable** :

- Notebook avec premier modèle + résultats initiaux.
- Tableau comparatif des modèles + résultats.

**Semaine 4 : Optimisation**

- Hyperparam tuning (batch size, learning rate, optimiseur).
- Expérimentation avec data augmentation avancée.
- Début des tests sur validation croisée.

**Livrable** : Rapport d'expérimentation (paramètres testés + meilleures performances).

**Semaine 5 : Évaluation robuste**

- Test sur données non vues.
- Analyse des erreurs fréquentes.
- Comparaison avec travaux existants (benchmarks du dataset).

**Livrable** : Rapport intermédiaire + graphiques de résultats.

**Semaine 6 : Déploiement prototype**

- Création d'une petite application (Streamlit, Flask ou interface Python).
- Intégration du modèle entraîné.
- Interface simple (upload d'image / détection en live via webcam).

**Livrable** : Prototype fonctionnel (exécutable localement).

**Semaine 7 – Tests utilisateurs / scénarios réels**

- Simulation d'un cas réel (ex : prise de photo de plante malade, vidéo de pollution, webcam pour malvoyant).
- Collecte de feedback sur la facilité d'usage.

**Livrable** : Notes de tests + ajustements sur le prototype.

**Semaine 8 – Améliorations finales**

- Correction des problèmes détectés.
- Ajout éventuel d'extensions (tableau de bord, retour vocal, cartes interactives).
- Préparation de la version finale du modèle.

**Livrable** : Version finale du prototype.

**Semaine 9 – Rédaction du rapport**

- Rédaction scientifique structurée :
    - Introduction + problématique
    - État de l'art
    - Méthodologie
    - Résultats et discussion
    - Conclusion et perspectives
- Mise en page académique.

**Livrable** : Version quasi finale du rapport.

**Semaine 10 – Soutenance & livraison**

- Préparation d'une présentation PowerPoint/Poster (10-15 min).

- Démo du prototype.
- Soutenance orale.

**Livrables finaux** :

1. Rapport scientifique complet (≈ 20 pages).
2. Code source documenté (GitHub/Google Drive).
3. Prototype fonctionnel (appli ou notebook interactif).
4. Présentation orale.