

Architecture de la Solution pour l'Entraînement YOLOv5 sur Données de Prunes

Lien des données dataset et labels(<https://files.fm/u/6av6h6kzns>)

Cette solution permet d'entraîner un modèle YOLOv5 pour détecter différents défauts sur des prunes (meurtries, fissurées, pourries, tachetées, intactes, non mûres).

1. Structure des Fichiers et Dossiers

La solution repose sur une organisation stricte des données et un script Python automatisé pour l'entraînement.

1.1. Arborescence des Données (ici se trouve le dataset et label : (<https://files.fm/u/6av6h6kzns>)

```
/
└─ data_plum/
    ├── images/
    │   ├── train/      Images d'entraînement (ex: crackedplum6.png)
    │   └─ val/        Images de validation
    └─ labels/
        ├── train/      Fichiers .txt des annotations YOLO
        └─ val/         Fichiers .txt des annotations de validation
```

1.2. Fichiers Clés

| Fichier/Dossier | Rôle |

| - |

| custom-data-plum.yaml | Configuration YOLOv5 (chemins des données et noms des classes) |

| yolov5s.pt | Modèle pré-entraîné YOLOv5 (transfer learning) |

| train.py (YOLOv5) | Script d'entraînement officiel |

2. Fonctionnement du Script

Le script principal effectue 3 étapes clés :

2.1. Préparation des Données

Vérification des permissions (accès en écriture au dossier labels/)

Suppression des anciens fichiers cache (train.cache, val.cache)

Validation de la structure des dossiers (images + annotations)

2.2. Entraînement du Modèle

- Commande YOLOv5 exécutée :

bash

python train.py \

--img 640 \

--batch 16 \

--epochs 20 \

--data custom-data-plum.yaml \

--weights yolov5s.pt \

--project results \

--name plumdetection

- Paramètres clés :

- Taille d'image (--img 640)

- Batch size (--batch 16)

- Nombre d'époques (--epochs 20)

- Fichier de configuration (--data)
- Modèle de base (--weights yolov5s.pt)

2.3. Gestion des Erreurs

Problèmes résolus automatiquement :

- Permissions bloquantes → Vérification + correction
- Fichiers cache corrompus → Suppression avant entraînement
- Annotations manquantes → Arrêt propre avec message clair

3. Configuration Requise

3.1. Environnement Python

Librairies installées :

```
bash
```

```
pip install torch==1.12.1 torchvision==0.13.1 tensorboard==2.10.0 yolov5
```

3.2. Fichier YAML (custom-data-plum.yaml)

```
yaml
```


```
train: C:/Users/Moi/venv2/intro/dataplum/images/train
```

```
val: C:/Users/Moi/venv2/intro/dataplum/images/val
```

```
nc: 6  Nombre de classes
```

```
names: ['bruised', 'cracked', 'rotten', 'spotted', 'unaffected', 'unripe']
```

4. Résultats Attendus

 Dossier de sortie (results/) :

results/

└─ plumdetection/


│ └─ weights/

│ │ └─ best.pt Meilleur modèle

│ │ └─ last.pt Dernier modèle

│ └─ trainbatchX.jpg Exemples d'entraînement

└─ results.png Métriques (précision, rappel)

 Visualisation avec TensorBoard :

bash

tensorboard --logdir results

5. Améliorations Possibles

Optimisations suggérées :

✓ Augmentation des données (data augmentation)

✓ Fine-tuning (ajustement des hyperparamètres)

✓ Export vers ONNX/TensorRT pour déploiement

Conclusion

Cette solution automatise l'entraînement YOLOv5 pour la détection de défauts sur des prunes, en garantissant :

Une structure de données claire

Une gestion robuste des erreurs

Un modèle optimisé prêt à l'emploi