

# Projet Module DL

## Herbiers & CrossViT : Segmentation, pondération par patch et interprétabilité

Sujet de projet à remettre avant le 27 janvier 2026, 23h59  
Dr. Youcef Sklab

### Contexte

Nous étudions l'identification / description de spécimens d'**herbiers** numérisés. Chaque image présente une plante sur une planche avec des éléments non végétaux (étiquettes, codes-barres, règles). Pour un même échantillon, on dispose d'une *image non segmentée* (planche complète) et d'une *image segmentée* (fond supprimé / masqué, ne laissant que la plante). L'objectif est d'analyser l'impact de la segmentation sur l'apprentissage avec une architecture **type Cross-ViT** ([lien](#)) (deux branches) et d'introduire une **pondération par patch** basée sur la densité de plante.

### Exemple visuel



FIGURE 1 – Exemple d'images d'herbier.

### Objectifs

- 01. Comparer** quatre configurations CrossViT (deux branches *Small/Large*) :
- A** : images *non segmentées* uniquement ;
  - B** : images *segmentées* uniquement ;
  - C1** : *segmentées* → *Large*, *non segmentées* → *Small* ;
  - C2** : *segmentées* → *Small*, *non segmentées* → *Large*.

- O2. Modifier l'architecture pour utiliser **deux branches de même résolution** (même taille de patch, même nombre de tokens), recevant respectivement l'image non segmentée et l'image segmentée, avec une fusion croisée identique.
- O3. **Pondération par patch** : proposer une fonction  $f(r)$  pour calculer le *poids*  $w_p$  d'un patch à partir du *ratio de pixels plante*  $r_p$  (dans l'image segmentée), et *propager* ce poids aux patches co-localisés de l'image non segmentée.  
**Important** : conserver la *même grille de patches* (même taille de patch et même alignement) entre non segmentée et segmentée pour pouvoir partager les poids par patch.
- O4. **Interprétabilité** : analyser les **cartes d'attention** (*heatmaps / attention rollout*) et quantifier le recouvrement avec la plante via l'**Intersection over Union (IoU)**, puis **intégrer** ce recouvrement (IoU) dans la **fonction de perte**.
- O5. **Test** : refaire les entraînements de l'objectif 2 en utilisant la fonction de perte avec terme IoU.

## Données

- Paires d'images alignées ([Lien](#)) : (*non segmentée, segmentée*) pour un même spécimen, avec masques de segmentation (ou images binaires segmentées) permettant de compter, pour chaque patch, le *nombre de pixels plante* (le **lien de téléchargement** sera communiqué au groupe par message *Microsoft Teams*).
- Étiquettes cibles : **Présence / Absence d'épines**.

## AVERTISSEMENT – CONFIDENTIALITÉ DES DONNÉES

Il est **formellement interdit** de diffuser, partager ou publier les données de ce projet, **sous quelque forme que ce soit** (extraits, images, fichiers, métadonnées), **sur Internet ou toute plateforme publique** (sites web, réseaux sociaux, dépôts de code, forums, etc.).

## Partie 1 — CrossViT de base & comparaisons

### Architecture de référence

Deux encodeurs ViT (*Small/Large*) avec *fusion croisée* (cross-fusion / cross-attention) et tête de classification (token [CLS] ou moyenne des tokens). Vous pouvez réutiliser une implémentation existante ([lien](#)).

### Configurations à évaluer

A, B, C1, C2 comme défini plus haut. Garder les **mêmes hyperparamètres** (taux d'apprentissage, époques, augmentations) pour une comparaison *équitable*.

### Protocole

Split train (80%) / val (20%) reproductible. Reporter Accuracy et F1-score ; fournir les courbes d'apprentissage.

## Partie 2 — Deux branches de même résolution

Adapter l'architecture pour que les deux branches aient la **même résolution de patchs** (même *patch size*, même nombre de tokens). Les poids ne sont pas partagés entre branches. Entrées :

- Branche 1 : image **non segmentée** ;
- Branche 2 : image **segmentée**.

Refaire l'évaluation et comparer à la meilleure configuration de la Partie 1.

## Partie 3 — Pondération par patch & perte pondérée

### Poids par patch (côté segmenté)

Pour chaque image segmentée, découper le masque en patches alignés avec le ViT. Pour un patch  $p$  :

$$r_p = \frac{\#\text{pixels plante}}{\#\text{pixels du patch}} \in [0, 1], \quad w_p = f(r_p).$$

Exemples de  $f$  (mais pas que !) :

- **Linéaire** :  $w_p = \varepsilon + r_p$  ;
- **Puissance** :  $w_p = (\varepsilon + r_p)^\gamma$ ,  $\gamma \in [0.5, 2]$  ;
- **Normalisation (par image)** :  $\tilde{w}_p = \frac{w_p}{\frac{1}{P} \sum_q w_q}$  (moyenne unitaire).

### Propagation aux images non segmentées

Assigner à chaque patch de l'image non segmentée le **même poids**  $w_p$  que le patch *co-localisé* de l'image segmentée (mêmes coordonnées de patch).

## Partie 4 — Heatmaps / Rollout & IoU

**Rappel (heatmaps & attention rollout).** *Heatmap* : visualisation continue de l'importance spatiale (moyenne des poids d'attention par patch, gradient  $\times$  activation, etc.). *Attention rollout* : agrégation multi-couches des matrices d'attention (moyenne des têtes, ajout de l'identité, renormalisation, produit cumulatif) pour estimer l'influence globale de chaque patch vers le token [CLS]. Ces cartes sont interpolées à la taille image et superposées à l'entrée pour l'interprétation.

- Attention rollout** : pour chaque couche  $l$ , moyenne des têtes  $A^{(l)}$ , ajout de l'identité, renormalisation, puis produit cumulatif  $\tilde{A} = \prod_l A^{(l)}$  afin d'estimer l'influence globale des patches sur [CLS].
- Heatmaps** : superposer la carte (rollout ou attention) aux images (segmentées ou non) pour interpréter les décisions.
- IoU (évaluation)** : binariser la carte (p. ex. seuil à un quantile, 0.8) et calculer l'Intersection over Union avec le masque plante :

$$\text{IoU} = \frac{|M_{\text{att}} \cap M_{\text{plante}}|}{|M_{\text{att}} \cup M_{\text{plante}}|}.$$

Rapporter moyenne et écart-type de l'IoU sur val/test pour A/B/C1/C2 et la variante même résolution.

- IoU dans la fonction de perte** : intégrer l'information de localisation (segmentée) directement dans l'entraînement via une perte auxiliaire. Justifiez votre choix parmi les options ci-dessous et comparez à la version sans régularisation IoU.

### Intégration dans la fonction de perte

Intégrer le score IoU dans la fonction de perte afin d'améliorer l'attention du modèle sur la plante elle-même pendant l'entraînement, c'est-à-dire introduire une pénalité lorsque le modèle se focalise sur l'arrière-plan au lieu de la plante.

# Évaluation et livrables

## Métriques

- **Classification** : Accuracy, F1-score (et matrice de confusion) ;
- **Interprétabilité** : IoU (rollout vs masque plante) ;
- **Ablations** : tableau comparant A/B/C1/C2 + variante même résolution  $\pm$  loss pondérée ;
- **Courbes** : pertes train/val, F1 par époque.

## À rendre

1. Un **rapport** (6–10 pages) : données, protocoles, architectures, ablations, choix de  $f(r)$ , analyse des heatmaps/IoU, discussion.
2. Le **code** et/ou notebooks reproductibles (seed, versions, README).
3. **Tableaux/figures** : résultats chiffrés et visualisations (heatmaps superposées).