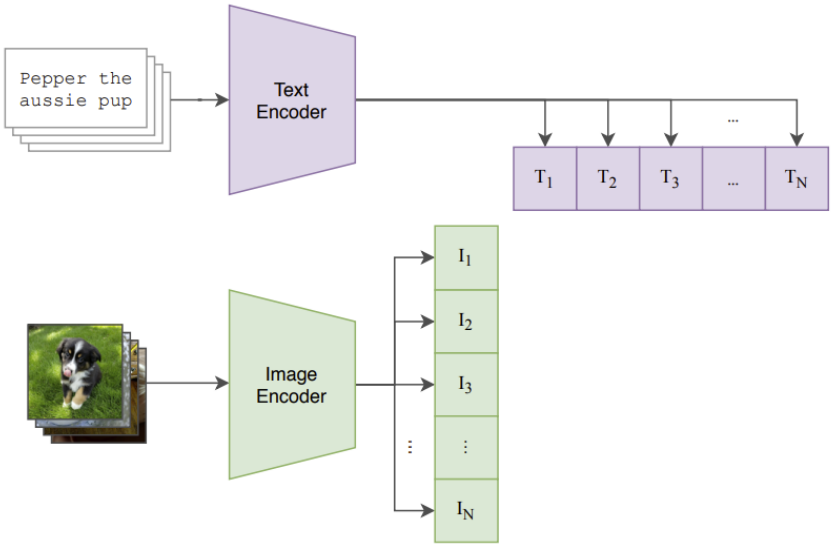
Rapport détaillé :

Développement d'un moteur de recherche avec un modèle multi-modal



Auteurs : Bilal El Biyadi, Yvan Louamba
Matière : Deep Learning
Organisation : CY Tech
Tuteur : Youcef SKLAB
Spécialité : Ing 3 IA

6 Janvier 2025

Sommaire

1 Introduction et contexte

Le présent rapport décrit la mise au point d'un moteur de recherche multimodal visant à associer des images d'herbier à des descriptions textuelles. L'objectif est de permettre à un utilisateur de rechercher efficacement une plante par mots-clés ou descriptions en langage naturel, et de lui présenter les images les plus pertinentes.

Pour ce faire, nous nous sommes appuyés sur une technologie d'apprentissage contrastif (**CLIP**) pré-entraînée, ainsi que sur une approche alternative **BERT + ResNet**. L'étude s'articule autour de :

- La mise en place d'un pipeline complet (acquisition des données, génération d'annotations, filtrage, entraînement, évaluation, interface de visualisation).
- La comparaison de plusieurs stratégies de fine-tuning et de différents modèles (CLIP, BERT + ResNet).
- La résolution de problèmes de stabilité numérique (`NaN`) lors de l'utilisation de CLIP.
- L'optimisation de la correspondance image-texte sur des données botaniques spécialisées.

2 Présentation des datasets (images et annotations)

Dans ce projet, nous avons exploité deux **datasets** distincts, tous deux issus d'un *même répertoire d'images* d'herbier numérisé.

Figure 1: Exemple d'image de feuille issue du répertoire d'images d'herbier numérisé

Chaque dataset dispose cependant d'une structure d'annotations et d'un nombre de **features** différents. Nous n'avons retenu, au total, qu'environ **300 échantillons** (images + annotations) pour chacun des deux fichiers d'annotation, afin de constituer un ensemble gérable manuellement et adapté à nos contraintes de temps de calcul.

2.1 Dataset 1 : structure simplifiée

Le premier dataset se présente sous la forme suivante (fichier `annotations-herbier.csv`) :

```
code;epines;feuille_ext_acuminee
ANG 419,00;1.0;1.0
ANG 430,00;1.0;0.0
ANG 2056,00;1.0;0.0
ANG 5088,00;1.0;0.0
...
```

Il comporte :

- `code` : identifiant unique (ex. `ANG 419,00`).
- `epines` : variable binaire (0/1) indiquant la présence ou l'absence d'épines.
- `feuille_ext_acuminee` : variable binaire (0/1) précisant la forme de la feuille.

Ce dataset est relativement *basique* : chaque plante est décrite par deux attributs seulement, toujours liées aux mêmes *images* dans un répertoire commun.

2.2 Dataset 2 : structure plus riche

Le second dataset (`annotations-herbier_2.0.csv`) contient plus de **features** :

```
T;code;E;BF;TF
0.0;CLF 289236,0000;1.0;1.0;1.0
0.0;P03327766;1.0;1.0;1.0
0.0;P04681621;0.0;0.0;1.0
1.0;P03550384;0.0;0.0;1.0
...
```

On y trouve :

- `T` : Tige (Herbacée, ligneuse) encodée en 0/1.
- `code` : identifiant unique (`P03327766`, `CLF 289236,0000`, etc.).
- `E` : Épine (0/1 : absence ou présence).
- `BF` : Bordure des feuilles (0/1 : lisse ou dentée).
- `TF` : Taille des feuilles (ex. 0/1/2, *Moins d'1 cm*, *Entre 1 cm et 10 cm*, *Plus de 10 cm*).

Ces attributs donnent une description *plus riche* de la plante, et nécessitent un prompt plus détaillé lors de la génération de la phrase descriptive. Tout comme pour le dataset 1, `code` renvoie aux *mêmes images* situées dans le répertoire commun.

3 Phase de préparation du dataset multimodal

3.1 Génération des paires (image, description)

Pour chacun des deux datasets, la démarche est similaire :

1. Lecture du fichier CSV et vérification de l'existence de l'image (à partir du `code`, qui pointe au même répertoire).
2. Génération d'une **courte description textuelle** via un *prompt* adapté aux attributs du dataset (OpenAI GPT, ou Google Gemini 1.5 Flash).
3. Stockage final sous la forme d'un fichier JSON contenant {`image_filename`, `code`, `description`}.

Genèse des données via Google Gemini 1.5 Flash

Nous avons également tenté de générer les données descriptives avec le modèle **Google Gemini 1.5 Flash**, lequel produit des phrases de qualité équivalente à OpenAI GPT-4. Cependant, nous avons constaté un *temps de génération plus élevé* pour un résultat globalement similaire. Par conséquent, nous avons conservé **OpenAI GPT-4** pour la majeure partie de la génération, car il offrait un meilleur compromis entre qualité et vitesse.

3.2 Filtrage et nettoyage

Après génération des paires :

- Nous éliminons les lignes dont l'image est introuvable dans le répertoire commun.
- Nous supprimons les descriptions vides.
- Nous scindons l'ensemble en *train* / *val* / *test* (70 %, 15 %, 15 %).

Ainsi, nous disposons d'un nombre final d'environ 300 échantillons pour chaque dataset.

3.3 Chargement dans le pipeline

Dans les scripts d'entraînement (CLIP ou BERT+ResNet), nous :

1. Lisons le JSON contenant (`image_filename`, `description`).
2. Vérifions la validité de l'image (ouvertures, etc.) et tokenisons le texte.
3. Chargeons le tout dans un *DataLoader* PyTorch (mini-lots).

4 Analyse du modèle CLIP

4.1 Difficultés rencontrées

4.1.1 `NaN` lors de l'entraînement

Au début, l'usage de `CLIP` a généré des `NaN` *même sur un dummy input*. Nous avons attribué ce problème à un *overflow* numérique (`fp16`). La solution a consisté à :

- Forcer le **passage en float32** (`model.float()`).
- Consommer plus de mémoire GPU, mais éviter les saturations en demi-précision.

4.1.2 Limitation du tokenizer CLIP à 77 tokens

Par ailleurs, CLIP ne traite que *77 tokens*, tronquant le texte au-delà. Pour nos prompts relativement brefs (description botanique concise), cette limite ne s'est pas avérée pénalisante, tant que nous veillons à respecter cette contrainte.

4.2 Test CLIP sans fine-tuning (Baseline)

Nous avons d'abord testé **CLIP pré-entraîné** sur chacun des deux datasets, établissant un point de référence (*baseline*). Les résultats sont synthétisés ci-dessous :

Dataset 1 (simplifié)	**Train Acc.**	**Val Acc.**	**Test Acc.**
Baseline CLIP (prétr.)	29.52%	–	28.89%

Table 1: Performances initiales (dataset 1).

Dataset 2 (plus riche)	**Train Acc.**	**Val Acc.**	**Test Acc.**
Baseline CLIP (prétr.)	29.52%	28.89%	22.22%

Table 2: Performances initiales (dataset 2).

On note que le test accuracy du dataset 2 (22.22 %) est plus bas que celui du dataset 1 (28.89 %). La plus grande richesse en features textuelles ne se retrouve pas exploitée sans fine-tuning.

4.3 Approche de fine-tuning 1 : déblocage partiel des couches de CLIP

Nous avons restreint l'apprentissage à certaines couches seulement (ex. *projection heads* et parfois la *dernière couche du transformeur*). Concrètement, dans le *fine-tuning 1*, nous avons débloqué :

- `visual.proj` et `text_projection`.
- Quelques couches de projection associées, sans modifier le gros des `visual.transformer.resblocks`.

Approche FT1 (dataset 1)	**Train Acc.**	**Val Acc.**	**Test Acc.**
Finetuning partiel (1er essai)	25.24%	26.67%	26.67%
Finetuning partiel (2e essai)	72.86%	46.67%	48.89%

Table 3: Approche de fine-tuning partiel pour dataset 1.

Sur le dataset 1 : Le second essai (différents réglages) montre un net bond en test (48.89 %), dépassant le baseline.

Approche FT1 (dataset 2)	**Train Acc.**	**Val Acc.**	**Test Acc.**
Finetuning partiel	62.38%	37.78%	31.11%

Table 4: Approche de fine-tuning partiel pour dataset 2.

Sur le dataset 2 : On gagne environ 9 points (22.22% $\rightarrow$ 31.11%) sur le test.

4.4 Approche de fine-tuning 2 : déblocage plus profond + data augmentation

Pour la *fine-tuning 2*, nous avons **débloqué davantage de couches** :

- `visual.proj`, `text_projection`,
- Les dernières couches du `visual.transformer.resblocks.NN` (par exemple 11),
- `visual.ln_post`, `ln_final`.

Nous avons aussi incorporé de la *data augmentation* (flip, recadrage, etc.) :

Approche FT2 (dataset 1)	**Train Acc.**	**Val Acc.**	**Test Acc.**
Finetuning + DataAug Profond	72.86%	46.67%	48.89%

Table 5: Approche de fine-tuning plus profond pour dataset 1.

Sur le dataset 1 : Les performances sur le dataset 1 montrent que l'approche de fine-tuning avec augmentation des données suggère une amélioration des performances.

Approche FT2 (dataset 2)	Train Acc.	Val Acc.	Test Acc.
Finetuning + DataAug Profond	69.52%	44.44%	35.56%

Table 6: Approche de fine-tuning plus profond pour dataset 2.

Sur le dataset 2 : Dans chaque cas, la *test accuracy* est encore améliorée, aboutissant à 48.89 % pour *dataset 1* et 35.56 % pour *dataset 2*. L'écart (13 points) reflète possiblement la complexité supérieure du second dataset.

5 Postprocessing : développement du moteur de recherche

Après l'entraînement et le fine-tuning des différents modèles (CLIP ou BERT+ResNet), nous avons mis en place un *postprocessing* visant à :

- **Sauvegarder les poids finetunés** (CLIP ou BERT+ResNet) dans différents fichiers (`.pth`).
- **Générer et stocker les embeddings d'images** (en JSON), en utilisant le modèle sélectionné ou fine-tuné.
- Mettre à disposition une **interface utilisateur** (via Streamlit) pour charger *au choix* un modèle (CLIP ou BERT+ResNet) et les embeddings associés.

5.1 Sauvegarde des poids et embeddings

Chaque configuration de fine-tuning (partiel ou profond) aboutit à un fichier de poids (`.pth`). Nous chargeons ensuite ces poids dans un script de *postprocessing* qui :

1. Charge le modèle choisi (CLIP ou BERT+ResNet) avec l'architecture correspondante.
2. Parcourt **toutes les images** du dataset, applique les transformations de preprocessing, puis encode l'image en embedding.
3. Normalise cet embedding L2, et le stocke (avec le `image_filename` et `description`) dans un fichier JSON.

Chaque fichier JSON (embeddings) est ainsi associé à un modèle particulier (*clip_finetuned* ou *bert_finetuned*). L'utilisateur peut ensuite choisir quel modèle exploiter lors de la requête.

5.2 Similitudes moyennes observées

Nous avons remarqué, lors du calcul final des *embeddings* et des comparaisons texte-image :

Curieusement, **CLIP** aboutit à des similarités moyennes avoisinant 0.25, tandis que **BERT+ResNet** plafonne à 0.07, alors que ce dernier pouvait afficher *de bons résultats*

Modèle	Sim. Moyenne Observée
CLIP (finetuned)	≈ 0.25
BERT+ResNet (finetuned)	≈ 0.07

Table 7: Scores de similarité moyens observés (embeddings).

en termes d'accuracy top-1. Cette perte d'information inexpliquée ne reflète pas nécessairement la performance réelle du moteur (puisque le *classement* reste pertinent). C'est un point qu'il conviendra d'approfondir pour comprendre l'origine exacte de cet écart de résultat entre les valeurs obtenues après les phases de finetunning et les score de similarités obtenus.

5.3 Définition de l'interface et de la requête

L'interface Web (Streamlit) propose :

- Un champ de saisie pour la **query** (description textuelle).
- Un sélecteur ou un argument pour choisir **soit CLIP (finetuné), soit BERT+ResNet (finetuné)** ou d'autres variantes.

Une fois le modèle et ses embeddings chargés, la requête textuelle est encodée via la partie *texte* correspondante (CLIP-text ou BERT). On calcule la *similarité* cosinus avec chacun des embeddings d'images sauvegardés, puis on affiche les images **top-k** les plus pertinentes.

5.4 Gestion multi-modèles

Grâce à ce mécanisme :

1. L'utilisateur spécifie le modèle qu'il souhaite utilisé
2. Le script charge le fichier `finetuned_xxx.pth` correspondant et le JSON d'*embeddings* approprié (`image_embeddings_xxx.json`).
3. L'interface se lance avec ces paramètres, permettant de **tester plusieurs configurations** sans modifier le code principal.

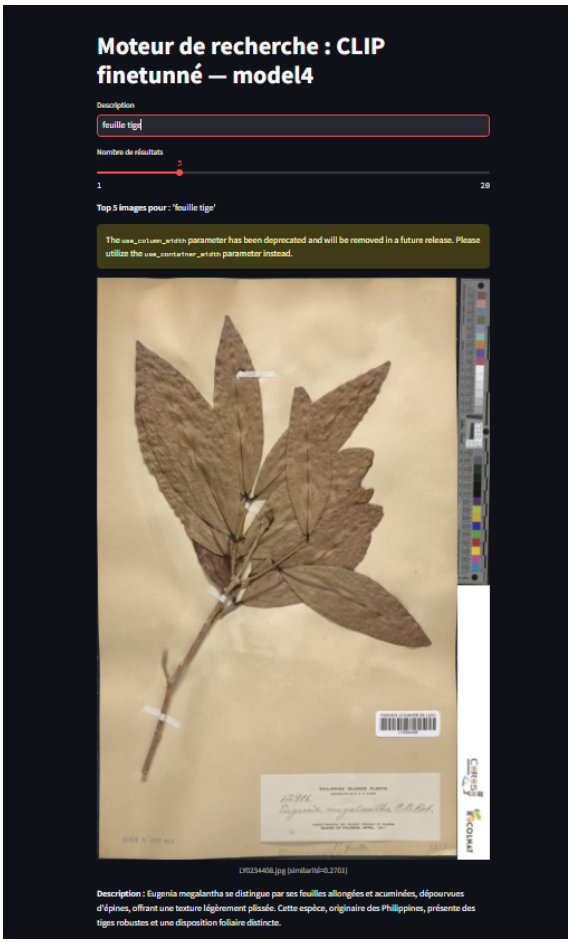


Figure 2: Exemple d'interface graphique (Streamlit) pour la recherche.

6 Tableau final : récapitulatif complet des résultats

Pour conclure, nous présentons un **tableau global** de tous les résultats mentionnés dans ce rapport, afin de comparer rapidement CLIP (sur deux datasets) et BERT+ResNet (sur un dataset herbier d'environ 300 échantillons) :

Modèle	Configuration	Train Acc.	Val Acc.	Test Acc.
CLIP - DS1	Baseline (prétr.)	29.52%	–	28.89%
	Fine-tune Partiel (1er)	25.24%	26.67%	26.67%
	Fine-tune Partiel (2e)	72.86%	46.67%	48.89%
CLIP - DS2	Baseline (prétr.)	29.52%	28.89%	22.22%
	Fine-tune Partiel	62.38%	37.78%	31.11%
	Fine-tune Profond + DataAug	69.52%	44.44%	35.56%
BERT+ResNet	Baseline (No FT)	24.00%	33.33%	24.44%
	Fine-tuning (10 époques)	34.76%	42.22%	47.33%

Table 8: Tableau récapitulatif final des résultats (tous modèles et configurations).

7 Conclusion et perspectives

Les expériences menées dans le cadre de ce projet mettent en évidence la pertinence du *transfert de connaissances* pour la recherche image-texte dans le domaine botanique, et ce à partir d'un volume de données relativement restreint (300 échantillons). Nous avons :

- Créé deux datasets d'annotations distinctes (un simplifié, un plus riche), chacun d'environ 300 images annotées (même répertoire d'images).
- Généré deux fichiers JSON multimodaux via deux prompts différents (pour adapter la description aux attributs respectifs).
- Observé que **CLIP**, après un fine-tuning (partiel ou approfondi), améliorait sensiblement la précision top-1 (+10 à +20 points).
- Essayé **BERT + ResNet**, qui, bien que plus lourd et moins intuitif pour la correspondance image-texte, a vu sa *test accuracy* presque doubler (de $\sim 24\%$ à $\sim 47\%$) grâce à un réentraînement partiel.
- Étudié **Google Gemini 1.5 Flash** pour la génération textuelle, constatant un *temps de génération plus élevé* pour une qualité proche de GPT-4, ce qui nous a incités à conserver principalement OpenAI GPT-4.

Nous avons également relevé un *phénomène de compression* des similarités (scores autour de 0.07–0.27) après la sauvegarde et la réutilisation respective du modèle Bert/Resnet et CLIP. Il reste à déterminer si cela nuit à la qualité du classement top-k ou si c'est une simple variation d'échelle. Du point de vue *opérationnel*, l'interface **Streamlit** (déployée via `ngrok` en environnement Colab) s'avère suffisante pour démontrer la faisabilité et l'utilité d'un moteur de recherche multimodal : l'utilisateur saisit une requête textuelle, et le système renvoie les images pertinentes selon l'espace latent appris.

En guise de perspectives, un entraînement sur un **plus grand nombre d'époques** et l'ajout de **nouvelles données** (images / annotations) permettraient de dépasser les scores actuels.

Ainsi, cette étude constitue une preuve de concept solide que les gains observés grâce au fine-tuning soulignent combien il est crucial d'adapter un modèle pré-entraîné à un domaine spécifique, bien que les performances restent insuffisantes sur notre cas d'étude.