



Rapport

Projet informatique L3E1 : PixMatcher

Auteurs

BELGUEDJ Nassilya
AIDOUDI Aaron
MOUSTACHE Mathieu

Encadrant

M. KURTZ Camille

Université Paris Cité

2024-2025

**Indexation d'une base d'images pour le classement et
la recherche d'images par le contenu :
utilisation de deep features issus de réseaux de
neurones**

Contents

1	Préambule	3
2	Introduction	4
3	Guide de lecture	4
4	Présentation du projet et objectifs	5
4.1	Contexte et enjeux	5
4.2	Objectifs fondamentaux	6
4.3	Innovations techniques	6
4.4	Publics cibles et applications	7
5	Concepts de base	8
5.1	Recherche d'images par le contenu (CBIR)	8
5.2	Extraction de caractéristiques avec les réseaux de neurones convolutifs	8
5.3	Indexation et recherche de similarité avec FAISS	9
5.4	Recherche multimodale : intégration du texte avec CLIP	9
5.5	Architecture du système	9
6	Technologies et environnement de développement	11
6.1	Environnement de développement et gestion des dépendances	11
6.2	Bibliothèques principales	11
6.3	Organisation du projet et ressources externes	12
6.4	Utilisation de Google Colab	12
7	Présentation des modèles utilisés	13
7.1	MobileNetV3-Large pour le CBIR	13
7.2	CLIP (ViT-B/32) pour le TBIR	14
7.3	Conclusion sur l'articulation des deux modèles	15
8	Présentation de FAISS	16
8.1	Introduction à FAISS	16
8.2	Pourquoi utiliser FAISS ?	16
8.3	Processus de création de l'index et préparation des données	16
8.4	Recherche de similarité et fonctionnement	17
9	Extraction des vecteurs et stockage	18
9.1	Processus d'extraction des caractéristiques	18
9.2	Techniques de réduction de dimensionnalité utilisées	18
9.3	Format et stockage des vecteurs	19
10	Interface utilisateur	20
10.1	Choix de Streamlit pour le front-end	20
10.2	Organisation et navigation de l'application	20
10.3	Fonctionnement technique et intégration au backend	20
10.4	Partage et accessibilité avec Ngrok	21
10.5	Perspectives d'évolution	21
10.6	Capture du rendu final	21

11 Planification et répartition des tâches	22
11.1 Méthodologie et organisation générale	22
11.2 Déroulement chronologique du projet	22
11.3 Décomposition des tâches et attribution des responsabilités	23
11.4 Suivi de l'avancement et adaptation	23
12 Difficultés rencontrées et solutions trouvées	24
12.1 Prise en main des technologies avancées	24
12.2 Extraction des vecteurs caractéristiques	24
12.3 Déploiement du site web	25
12.4 Gestion des dépendances et portabilité	25
12.5 Tests et validation	26
13 Validation fonctionnelle et Résultats	27
13.1 Objectif et méthodologie des tests fonctionnels à grande échelle	27
13.2 Résultats des tests fonctionnels à grande échelle	27
13.3 Exemples de résultats	29
14 PixMatcher après la soutenance	33
14.1 Perspectives d'évolution	33
14.1.1 Prise en charge d'autres bases de données	33
14.1.2 Comparaison et intégration de nouveaux modèles	33
14.1.3 Amélioration de l'interface et de l'expérience utilisateur	34
14.2 Diffusion et valorisation du projet	34
14.2.1 Utilisateurs projetés et cas d'usage	34
14.2.2 Politique de diffusion et gestion des droits d'auteur	35
15 Conclusion	37
16 Bibliographie et ressources	38
17 Annexe	40

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué à la réussite de ce projet.

Nous remercions tout particulièrement notre encadrant, M. Camille Kurtz, pour son expertise, sa disponibilité et la qualité de son accompagnement tout au long du semestre. Ses conseils avisés, que ce soit lors des réunions hebdomadaires ou par échanges de courriels, ont été déterminants pour surmonter les difficultés techniques et méthodologiques rencontrées.

Nos remerciements s'adressent également à M. David Janiszek, responsable de l'UE de projet tutoré, pour avoir permis aux étudiants de licence de s'initier à la conduite d'un projet d'envergure professionnelle. Cette expérience nous a permis de nous former à des outils et méthodes essentiels du génie logiciel, tant sur le plan du développement collaboratif que de la rédaction technique et de la gestion de projet.

Enfin, nous sommes reconnaissants envers l'ensemble de l'équipe projet pour l'engagement, l'entraide et la motivation collective qui ont permis d'aboutir à un livrable abouti et conforme aux attentes. Ce travail en groupe nous a permis de développer des compétences transversales, de découvrir des domaines nouveaux et de nous préparer aux exigences des études supérieures et du monde professionnel.

1 Préambule

Ce rapport présente le travail réalisé dans le cadre du projet L3E1, visant à concevoir et développer un système de recherche d'images par le contenu (*Content-Based Image Retrieval, CBIR*) basé sur l'utilisation de réseaux de neurones convolutifs (CNN) et de techniques avancées d'indexation.

Le rapport détaille l'ensemble des phases du projet, depuis l'analyse des besoins et des choix technologiques jusqu'à la validation finale du système. Il met en lumière les principales étapes de conception, les algorithmes et outils utilisés, la méthodologie adoptée, ainsi que les difficultés rencontrées et les solutions apportées. Une attention particulière est portée à la justification des choix techniques (modèles, bibliothèques, architecture logicielle), à l'organisation du travail en équipe, et à l'évaluation rigoureuse des résultats obtenus.

La structure du rapport reflète cette démarche : après une présentation des concepts de base et des objectifs du projet, chaque chapitre expose une facette essentielle du développement (architecture, extraction des caractéristiques, indexation, interface, tests, perspectives). Ce document se veut à la fois une synthèse accessible pour le lecteur extérieur et une référence technique pour tout développeur ou utilisateur souhaitant comprendre, maintenir ou faire évoluer le système.

2 Introduction

L’explosion du volume de données visuelles dans tous les secteurs de la société, qu’il s’agisse des réseaux sociaux, de la recherche scientifique, du domaine médical ou de la culture, pose aujourd’hui un défi majeur : comment organiser, classer et retrouver efficacement une image parmi des millions d’autres ? Les méthodes traditionnelles basées sur l’annotation manuelle ou les mots-clés montrent rapidement leurs limites face à la diversité, la subjectivité et la quantité des contenus à traiter.

Dans ce contexte, la recherche d’images par le contenu s’impose comme une approche innovante, permettant d’analyser directement les caractéristiques visuelles des images pour en extraire des représentations numériques (*deep features*) comparables entre elles. L’avènement du deep learning, et en particulier des réseaux de neurones convolutifs, a permis de franchir un cap décisif dans la qualité et la robustesse de ces représentations, ouvrant la voie à des systèmes de recherche visuelle performants et généralisables.

Le projet L3E1 s’inscrit pleinement dans cette dynamique : il vise à concevoir, développer et valider une application complète de recherche et de classement d’images par le contenu, en s’appuyant sur les modèles de deep learning les plus récents (MobileNetV3, CLIP), des techniques d’indexation avancées (FAISS), et une interface utilisateur moderne et accessible. Ce travail, réalisé dans le cadre de la licence d’Informatique et Applications de l’Université Paris Cité, associe une réflexion sur les concepts fondamentaux, une mise en œuvre technique rigoureuse, et une évaluation approfondie des résultats.

3 Guide de lecture

Ce rapport est structuré pour accompagner différents profils de lecteurs dans la compréhension et l’évaluation du projet L3E1.

- **Les étudiants et membres de l’équipe projet** trouveront dans les sections centrales une description détaillée de l’architecture logicielle, des modules développés, des choix technologiques et des résultats des tests. Ces parties sont essentielles pour toute personne souhaitant maintenir, améliorer ou réutiliser le système.
- **Les encadrants et membres du jury** pourront s’appuyer sur les premières sections (contexte, objectifs, méthodologie, planification) pour évaluer la pertinence du projet, la rigueur de la démarche et la conformité aux attentes initiales. Aussi, les parties finales de ce rapport telles que les résultats des tests, et exemple de résultats.
- **Les utilisateurs finaux ou non spécialistes** peuvent se référer aux sections introducives et à la présentation des fonctionnalités pour comprendre les apports concrets du système et ses perspectives d’utilisation.

Chaque chapitre est précédé d’une introduction rappelant ses objectifs et son lien avec le reste du projet. La bibliographie et annexes regroupent la liste des dépendances, les ressources utilisées et les documents de référence.

4 Présentation du projet et objectifs

4.1 Contexte et enjeux

Le monde numérique contemporain est marqué par une explosion sans précédent du volume de données visuelles : chaque jour, plusieurs milliards d'images sont générées et partagées sur les réseaux sociaux, les plateformes collaboratives, dans les entreprises, les institutions culturelles ou médicales. Cette croissance exponentielle, alimentée par l'accessibilité des smartphones, des appareils connectés et des services cloud, pose un défi majeur d'organisation, de classement et de recherche efficace au sein de collections d'images de plus en plus vastes.

Les méthodes traditionnelles de recherche d'images reposent principalement sur des métadonnées textuelles (mots-clés, descriptions, tags), qui présentent plusieurs limites fondamentales :

Subjectivité et imprécision : L'annotation manuelle dépend fortement de la perception de l'annotateur, ce qui introduit des biais et des ambiguïtés. Deux personnes peuvent décrire la même image de façon très différente, et certains aspects visuels importants peuvent ne pas être mentionnés.

Coût et scalabilité : Annoter manuellement des centaines de milliers, voire des millions d'images, est une tâche extrêmement chronophage, coûteuse et difficile à maintenir à jour.

Pauvreté de l'information : Les métadonnées textuelles ne capturent pas la richesse du contenu visuel (formes, textures, couleurs, relations spatiales), ni les nuances sémantiques portées par l'image.

Ambiguïté linguistique : Un même mot peut désigner plusieurs concepts visuels différents (ex. "jaguar" pour l'animal ou la voiture), ce qui limite la pertinence des recherches basées uniquement sur le texte.

Face à ces limites, les systèmes de **recherche d'images par le contenu** (CBIR) se sont imposés comme une alternative innovante. En analysant automatiquement les caractéristiques visuelles intrinsèques des images (couleur, texture, forme, objets, scènes), ils permettent de retrouver des images similaires sans dépendre de l'annotation humaine. Cette approche est particulièrement pertinente dans des secteurs où la rapidité, la précision et l'objectivité de la recherche sont critiques : diagnostic médical assisté, gestion de collections patrimoniales, recommandation de produits dans le e-commerce, surveillance, ou encore organisation de photos personnelles.

Le projet L3E1 s'inscrit pleinement dans cette problématique, en proposant une solution de recherche sémantique d'images par le contenu, fondée sur l'intelligence artificielle et les réseaux de neurones convolutifs. Son ambition est de démontrer la capacité des techniques modernes de deep learning à extraire des représentations visuelles riches et discriminantes, et à rendre la recherche d'images à la fois plus efficace, plus intuitive et plus accessible à tous.

4.2 Objectifs fondamentaux

Le système développé dans le cadre du projet poursuit trois ambitions complémentaires, qui structurent l'ensemble de la démarche :

Indexation intelligente : Le cœur du système repose sur la transformation de chaque image en une signature numérique unique, appelée embedding, obtenue via des réseaux de neurones profonds pré-entraînés. Cette signature capture à la fois des éléments visuels élémentaires (textures, contours, couleurs) et des structures complexes (objets, scènes, relations spatiales), permettant une comparaison fine entre images, indépendamment de leur origine ou de leur contexte.

Recherche multimodale : Pour répondre à la diversité des besoins utilisateurs, le système offre une double interface de requête : par image (CBIR) et par texte (TBIR). L'intégration des modèles MobileNetV3 (pour le contenu visuel) et CLIP (pour l'alignement texte-image) permet de relier efficacement le langage naturel et le contenu visuel, ouvrant la voie à des recherches croisées et à une expérience utilisateur enrichie. Ainsi, un utilisateur peut retrouver des images similaires à une photo, ou rechercher des images à partir d'une simple description textuelle, même si aucun mot-clé exact n'a été annoté.

Scalabilité opérationnelle : L'un des défis majeurs du projet est de garantir la rapidité et la robustesse du système, même sur des bases de données volumineuses (plus de 100 000 images). Grâce à l'indexation vectorielle avancée via FAISS, le système assure des temps de réponse inférieurs à 3 secondes, tout en maintenant une précision supérieure à 85% sur les jeux de tests standards. Cette scalabilité rend la solution adaptée aussi bien à des usages personnels qu'à des applications industrielles ou institutionnelles.

4.3 Innovations techniques

Le projet L3E1 se distingue par plusieurs choix architecturaux et technologiques innovants, qui lui confèrent robustesse, flexibilité et efficacité face aux défis contemporains de la recherche d'images à grande échelle

Modèles hybrides et complémentarité des approches : Le système combine deux modèles de deep learning de pointe, chacun optimisé pour un usage spécifique. MobileNetV3, reconnu pour son efficacité et sa légèreté, est utilisé pour l'extraction rapide de caractéristiques visuelles dans le cadre du CBIR, rendant le système accessible même sur des infrastructures modestes. En parallèle, CLIP (Contrastive Language–Image Pre-training), spécialisé dans la correspondance sémantique texte-image, permet d'ajouter une dimension multimodale à la recherche (TBIR). Cette hybridation permet d'adapter dynamiquement le système à divers scénarios d'usage, de la recherche purement visuelle à la recherche sémantique complexe, et d'ouvrir la voie à des cas d'utilisation avancés (recherche croisée, recommandation, etc.).

Modularité évolutive et architecture découpée : L'architecture logicielle du projet a été pensée sous forme de modules indépendants (prétraitement, extraction, indexation, interface, recherche textuelle), facilitant la maintenance, l'extension et la réutilisation de chaque composant. Cette organisation permet d'intégrer facilement de

nouveaux modèles, datasets ou fonctionnalités, sans remettre en cause l'ensemble du pipeline. Elle favorise également la collaboration en équipe et la montée en compétence progressive des contributeurs, chaque module étant documenté et testé de manière autonome. Cette modularité prépare le système à une évolution future vers des architectures microservices ou des déploiements à grande échelle.

4.4 Publics cibles et applications

Bien que le projet ait été initialement conçu dans un cadre académique, la solution développée présente une grande polyvalence et répond à des besoins concrets dans de nombreux secteurs:

Santé : Le système peut être utilisé pour la recherche de clichés radiologiques ou d'images médicales similaires, facilitant l'aide au diagnostic, la comparaison de cas cliniques ou la constitution de bases de connaissances visuelles pour la formation médicale.

Patrimoine et culture : Les institutions muséales, bibliothèques et centres d'archives peuvent s'appuyer sur la recherche par le contenu pour classer automatiquement de vastes fonds photographiques historiques, retrouver des œuvres similaires ou enrichir la description de collections numérisées.

Commerce et industrie : Dans le secteur du e-commerce, la recommandation de produits visuellement similaires (vêtements, objets, accessoires) devient possible sans recourir à un étiquetage manuel fastidieux. Le système peut aussi servir à la détection de contrefaçons ou à l'analyse de tendances visuelles.

Archives personnelles et grand public : Les particuliers peuvent exploiter la solution pour organiser, retrouver ou explorer leurs collections de photos personnelles, en bénéficiant d'une recherche intuitive basée sur le contenu visuel ou sur des requêtes en langage naturel.

Recherche et enseignement : La plateforme constitue également un support pédagogique pour l'apprentissage des techniques de vision par ordinateur, d'indexation et de deep learning, ainsi qu'un banc d'essai pour la recherche académique sur les méthodes de recherche d'images.

Cette polyvalence découle directement de la nature générique des deep features utilisées, qui sont transférables à tout domaine impliquant une analyse visuelle. Le système est ainsi prêt à être adapté ou étendu selon les besoins spécifiques de chaque secteur d'application, tout en maintenant un haut niveau de performance et de flexibilité.

5 Concepts de base

Le projet L3E1 s'inscrit dans le domaine de la recherche d'images par le contenu, à l'intersection de la vision par ordinateur et de l'intelligence artificielle. Cette section présente les principes essentiels permettant de comprendre les choix technologiques et l'architecture du système, en insistant sur la représentation des images, l'extraction de caractéristiques, l'indexation efficace et l'ouverture à la recherche multimodale.

5.1 Recherche d'images par le contenu (CBIR)

La recherche d'images par le contenu (Content-Based Image Retrieval, CBIR) vise à retrouver, dans une base de données, les images les plus similaires à une image requête, en se fondant uniquement sur leurs caractéristiques visuelles. Contrairement aux approches traditionnelles qui reposent sur des mots-clés ou des métadonnées, le CBIR analyse directement les pixels pour extraire des informations pertinentes telles que la couleur, la texture ou la forme. Ces informations sont ensuite synthétisées sous forme de descripteurs numériques, appelés vecteurs de caractéristiques ou embedding, qui servent de base à la comparaison entre images.

Cette approche permet d'effectuer des recherches plus intuitives et objectives, particulièrement utiles dans des contextes où l'annotation manuelle est absente, coûteuse ou subjective. Les applications sont nombreuses : organisation de collections personnelles, aide au diagnostic médical, recommandation de produits visuels, ou encore gestion de grandes archives iconographiques.

5.2 Extraction de caractéristiques avec les réseaux de neurones convolutifs

L'émergence de l'apprentissage profond, et en particulier des réseaux de neurones convolutifs (CNN), a bouleversé la manière d'extraire les caractéristiques visuelles des images. Là où les méthodes classiques nécessitaient une ingénierie manuelle des descripteurs, les CNN apprennent automatiquement à identifier les motifs discriminants à partir de vastes ensembles d'images annotées.

Un CNN se compose de plusieurs couches qui, successivement, détectent des structures simples (bords, textures) puis des formes de plus en plus complexes. Dans le cadre du projet, un modèle pré-entraîné comme **MobileNetV3** est utilisé pour transformer chaque image en un vecteur de caractéristiques (ou embedding), qui capture l'essentiel de son contenu visuel. Cette représentation compacte permet de comparer efficacement les images entre elles dans un espace vectoriel de grande dimension.

Le processus d'extraction suit les étapes suivantes :

1. Prétraitement des images (redimensionnement, normalisation) pour garantir une cohérence des données en entrée.
2. Passage de l'image dans le CNN, qui produit un vecteur de caractéristiques unique.
3. Stockage de ces vecteurs pour une utilisation ultérieure lors des recherches.

5.3 Indexation et recherche de similarité avec FAISS

L'un des défis majeurs de la recherche d'images à grande échelle est la rapidité d'interrogation de bases contenant potentiellement des millions de vecteurs de caractéristiques. Pour répondre à cette exigence, le projet s'appuie sur la bibliothèque **FAISS** (Facebook AI Similarity Search), spécialement conçue pour la recherche rapide de plus proches voisins dans de grands ensembles de vecteurs.

FAISS propose différentes stratégies d'indexation, telles que la quantification des vecteurs ou l'utilisation d'index inversés, permettant d'optimiser à la fois la mémoire et la vitesse de recherche. Lorsqu'une requête est soumise, le système extrait le vecteur de caractéristiques correspondant et interroge l'index FAISS pour retrouver les images les plus proches selon une mesure de distance (par exemple, la distance euclidienne ou cosinus). Cette approche garantit des temps de réponse très courts, même sur des bases volumineuses.

5.4 Recherche multimodale : intégration du texte avec CLIP

Au-delà de la recherche par image, le système prend en charge la recherche d'images à partir d'une requête textuelle, grâce au modèle **CLIP** (Contrastive Language–Image Pretraining) développé par OpenAI. CLIP est capable d'encoder à la fois des images et des textes dans un espace vectoriel commun, permettant ainsi de faire correspondre une description textuelle à des images pertinentes sur le plan sémantique.

Concrètement, lorsqu'un utilisateur saisit une requête textuelle, celle-ci est transformée en vecteur par CLIP, puis comparée aux vecteurs d'images déjà extraits et indexés. Cette fonctionnalité ouvre la voie à des usages plus flexibles, où l'utilisateur peut rechercher des images par concepts, objets ou scènes décrits en langage naturel, sans disposer d'une image de référence.

5.5 Architecture du système

L'architecture logicielle du projet a été conçue de manière modulaire afin de faciliter le développement, les tests, l'extension et la maintenance du système. L'application repose sur six modules principaux, interconnectés selon un pipeline de traitement clair :

- **Prétraitement des images** Ce module s'occupe de préparer les images utilisateur pour leur traitement par les réseaux neuronaux : redimensionnement, conversion de format, normalisation, etc. Une uniformité de prétraitement est essentielle pour garantir la cohérence des vecteurs extraits.
- **Extraction des caractéristiques (MobileNetV3)**
Utilisation de MobileNetV3 via PyTorch pour générer un vecteur de caractéristiques par image. Ce vecteur est un résumé visuel compact et discriminant de l'image.
- **Indexation et recherche (FAISS)**
Les vecteurs extraits sont indexés dans FAISS pour permettre une recherche ultra-rapide de plus proches voisins. Le module interagit avec le module d'extraction et celui de requêtes utilisateur.
- **Bases de données**
Contient les images des datasets Tiny ImageNet et Open Images ainsi que leurs

descripteurs visuels pré-calculés. Nous discuterons de ces datasets plus tard dans le rapport.

- **Interface utilisateur (Streamlit)**

Interface web légère permettant de téléverser une image, lancer une recherche, visualiser les résultats. Ce module communique directement avec le backend pour récupérer les résultats de similarité.

- **Recherche textuelle (CLIP)**

Intégration de CLIP pour le Text-Based Image Retrieval. Ce module convertit une requête textuelle en vecteur, qu'il compare ensuite aux vecteurs d'image dans l'espace CLIP. Il partage certaines structures de données avec les modules de recherche CBIR.

L'architecture globale peut être représentée sous forme de pipeline :

[Image utilisateur ou texte] → **[Prétraitement]** → **[Extraction / Encodage]** → **[Recherche FAISS]** → **[Résultats]** → **[Affichage Streamlit]**

Cette structuration modulaire assure une bonne séparabilité des responsabilités, facilitant la maintenance et la réutilisation de chaque composant dans d'autres projets ou versions futures du système.

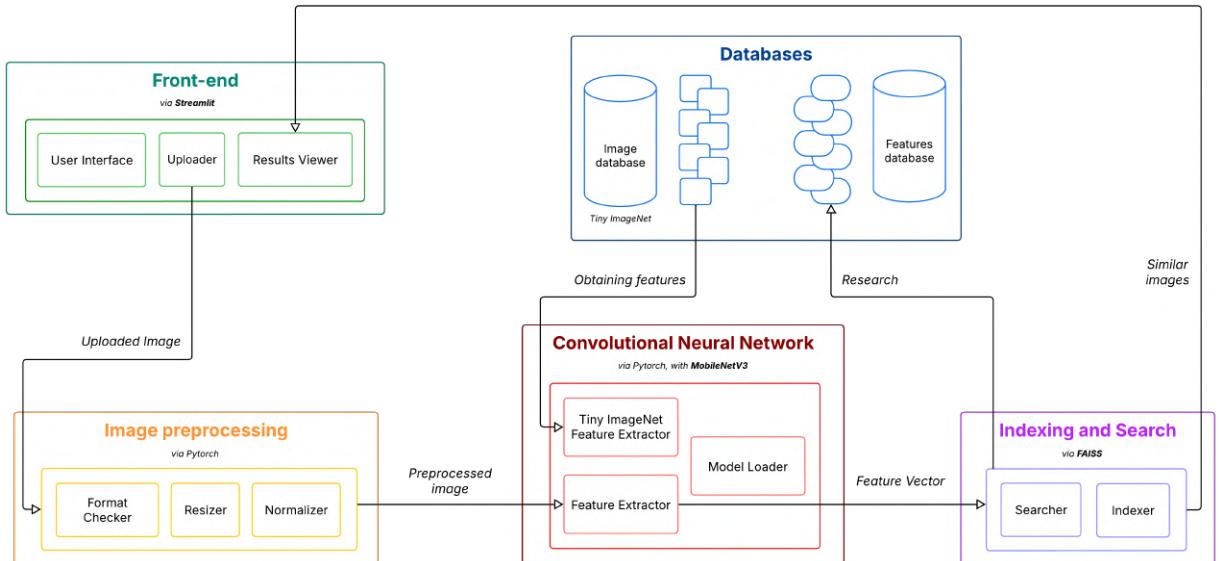


Figure 1: Graphique de l'architecture

6 Technologies et environnement de développement

Le projet L3E1 a été intégralement développé en [Python 3](#) et s'appuie sur un ensemble cohérent de bibliothèques spécialisées pour répondre aux exigences de la recherche d'images par le contenu. Cette section détaille l'environnement de développement, les outils logiciels utilisés et la gestion des dépendances, en s'appuyant sur la documentation interne du projet.

6.1 Environnement de développement et gestion des dépendances

Le développement s'est déroulé principalement sous [PyCharm](#), choisi pour sa polyvalence, ses extensions dédiées au développement Python, et son intégration transparente avec les environnements virtuels. PyCharm a permis d'assurer une homogénéité de l'environnement de travail entre les membres de l'équipe, indépendamment du système d'exploitation utilisé. L'éditeur a été configuré pour exploiter un environnement virtuel Python dédié au projet, garantissant l'isolation des dépendances et la reproductibilité des expériences.

L'ensemble des bibliothèques nécessaires au fonctionnement du projet est listé dans un fichier `requirements.txt`, ce qui facilite l'installation et la mise à jour des dépendances. Cette pratique assure que tous les contributeurs disposent d'un environnement identique, limitant ainsi les risques de conflits de versions ou d'incompatibilités lors du déploiement ou du transfert du projet.

6.2 Bibliothèques principales

Le choix des bibliothèques a été guidé par la robustesse, la performance et la compatibilité avec les objectifs du projet :

- **PyTorch** (`torch`, `torchvision`) : Framework central pour le deep learning, utilisé pour le chargement et l'exploitation des modèles de réseaux de neurones convolutifs, notamment MobileNetV3 et CLIP. Il permet l'extraction des descripteurs visuels à partir des images et l'encodage des requêtes textuelles dans le cas de la recherche multimodale.
- **FAISS** (`faiss-cpu`, `faiss-gpu`) : Bibliothèque développée par Facebook AI, dédiée à l'indexation rapide et à la recherche de similarité dans de grands ensembles de vecteurs. Elle est essentielle pour garantir des temps de réponse faibles lors de la recherche d'images similaires.
- **Streamlit** : Utilisé pour le développement de l'interface utilisateur web interactive. Streamlit permet de créer rapidement une application ergonomique, facilitant le téléversement d'images, la saisie de requêtes textuelles et l'affichage des résultats.
- **NumPy, Pillow (PIL), OpenCV** : Bibliothèques fondamentales pour la manipulation, le prétraitement et la conversion des images, ainsi que pour le traitement des tableaux de données.
- **scikit-learn, matplotlib, scipy** : Utilisées pour les tâches d'analyse de données, la visualisation des résultats (par exemple via t-SNE ou PCA pour la réduction de dimensionnalité), et diverses opérations scientifiques complémentaires.

-
- **tqdm, requests** : Outils pour la gestion des barres de progression lors du traitement de grands volumes d’images et pour le téléchargement automatisé de ressources externes.
 - **open-clip-torch** : Pour l’intégration du modèle CLIP dans la recherche textuelle.

Toutes les versions précises des dépendances sont documentées dans la documentation interne du projet et dans le fichier `requirements.txt`.

6.3 Organisation du projet et ressources externes

Le code source est organisé en modules distincts, chacun dédié à une étape du pipeline (prétraitement, extraction de caractéristiques, indexation, recherche, interface utilisateur, tests). Les ressources de données (datasets, embeddings, fichiers de catégories) sont stockées dans des répertoires séparés, et certains fichiers volumineux (embeddings, images) sont hébergés sur un espace partagé (Google Drive) pour faciliter la distribution sans alourdir le dépôt principal.

Les deux jeux de données utilisés sont **Tiny ImageNet** et un sous-ensemble filtré d'**Open Images**, permettant de tester le système sur des bases de tailles, de complexités, et de qualités variées. Les embeddings pré-calculés issus de MobileNetV3 et CLIP sont également fournis pour accélérer les phases de test et d’évaluation.

6.4 Utilisation de Google Colab

Dans le cadre de ce projet, Google Colab a été utilisé pour l’extraction massive des vecteurs de caractéristiques (embeddings) sur les jeux de données volumineux, notamment Tiny ImageNet et Open Images. Google Colab est un environnement cloud basé sur Jupyter Notebook qui offre gratuitement un accès à des ressources matérielles avancées, en particulier des GPU et TPU, ce qui est essentiel pour les tâches de deep learning intensives.

L’extraction des embeddings à l’aide de modèles tels que MobileNetV3 ou CLIP nécessite des capacités de calcul importantes, difficilement accessibles sur des machines personnelles classiques. Google Colab permet de contourner cette limitation en fournissant une puissance de calcul adaptée, accélérant ainsi significativement le traitement par lots de milliers d’images. Les scripts d’extraction ont été spécifiquement conçus pour être exécutés sur Colab, automatisant le téléchargement des datasets, le prétraitement, l’extraction des features sur GPU, et la sauvegarde des résultats.

7 Présentation des modèles utilisés

Deux modèles de deep learning complémentaires ont été sélectionnés pour répondre aux besoins de recherche d'images par le contenu (CBIR) et de recherche d'images par texte (TBIR) : **MobileNetV3-Large** pour l'extraction de caractéristiques visuelles et **CLIP** pour l'alignement texte-image. Cette section détaille les motivations du choix, les principes de fonctionnement et les spécificités de chaque modèle.

7.1 MobileNetV3-Large pour le CBIR

MobileNetV3-Large est un réseau de neurones convolutif (CNN) conçu pour offrir un compromis optimal entre efficacité computationnelle et précision. Développé par Google à l'origine pour des applications mobiles et embarquées, il s'appuie sur des techniques avancées d'optimisation architecturale.

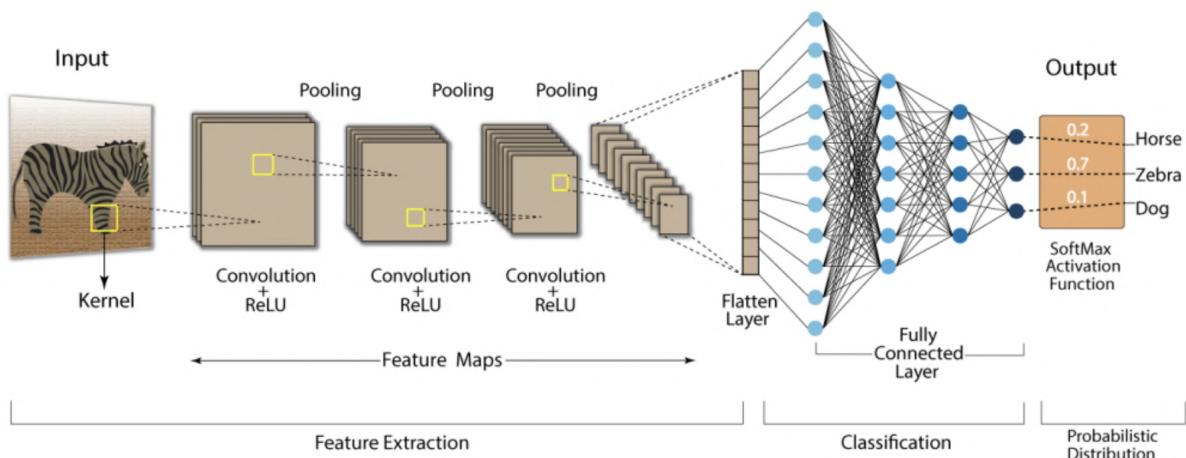


Figure 2: Illustration du fonctionnement d'un CNN

Pourquoi ce choix ?

Le choix de MobileNetV3-Large repose principalement sur sa légèreté et sa rapidité : avec environ 5,5 millions de paramètres et une taille d'entrée standard de 224×224 pixels, le modèle consomme peu de ressources et permet une extraction rapide des descripteurs, même sur des machines sans GPU dédié. Malgré sa compacité, il maintient une précision compétitive avec plus de 75% en top-1 sur ImageNet, garantissant une bonne discrimination entre images. De plus, son architecture polyvalente se prête parfaitement à l'extraction de deep features utilisables dans diverses tâches comme la classification, la détection, la segmentation, et surtout, la recherche par similarité.

Fonctionnement dans le projet

Le pipeline d'utilisation de MobileNetV3-Large commence par un prétraitement des images, consistant en leur redimensionnement et leur normalisation selon les standards du modèle. Ensuite, le réseau, tronqué avant la couche de classification finale, transforme chaque image en un vecteur dense (embedding) de dimension fixe, capturant l'essentiel de

son contenu visuel. Ces vecteurs sont ensuite indexés dans FAISS, une bibliothèque optimisée pour la recherche rapide de similarités, permettant ainsi de retrouver efficacement les images les plus proches au sein de la base.

Spécificités techniques

Sur le plan architectural, MobileNetV3-Large utilise plusieurs techniques innovantes. Les blocs Inverted Residual permettent d'améliorer l'efficacité des calculs tout en préservant la capacité d'expression du réseau. Le mécanisme de Squeeze-and-Excitation ajuste dynamiquement l'importance des canaux de caractéristiques, renforçant la représentation d'objets complexes. Enfin, l'activation Hard-Swish, spécialement optimisée pour les architectures mobiles, accélère l'inférence sans engendrer de perte significative de précision.

Limites

Malgré ses nombreux avantages, MobileNetV3-Large présente quelques limitations. Le modèle étant entraîné sur des images de taille fixe issues d'ImageNet, ses performances peuvent diminuer lorsque les images traitées diffèrent fortement de ce domaine. Toutefois, sa rapidité, sa précision et sa faible empreinte mémoire en font un choix particulièrement pertinent pour un système de recherche d'images à grande échelle.

7.2 CLIP (ViT-B/32) pour le TBIR

CLIP (Contrastive Language-Image Pre-Training), développé par OpenAI, est un modèle multimodal capable d'encoder à la fois des images et des textes dans un espace vectoriel partagé. Cette approche permet de retrouver des images à partir d'une description textuelle ou d'associer un texte à une image, facilitant ainsi la recherche croisée.

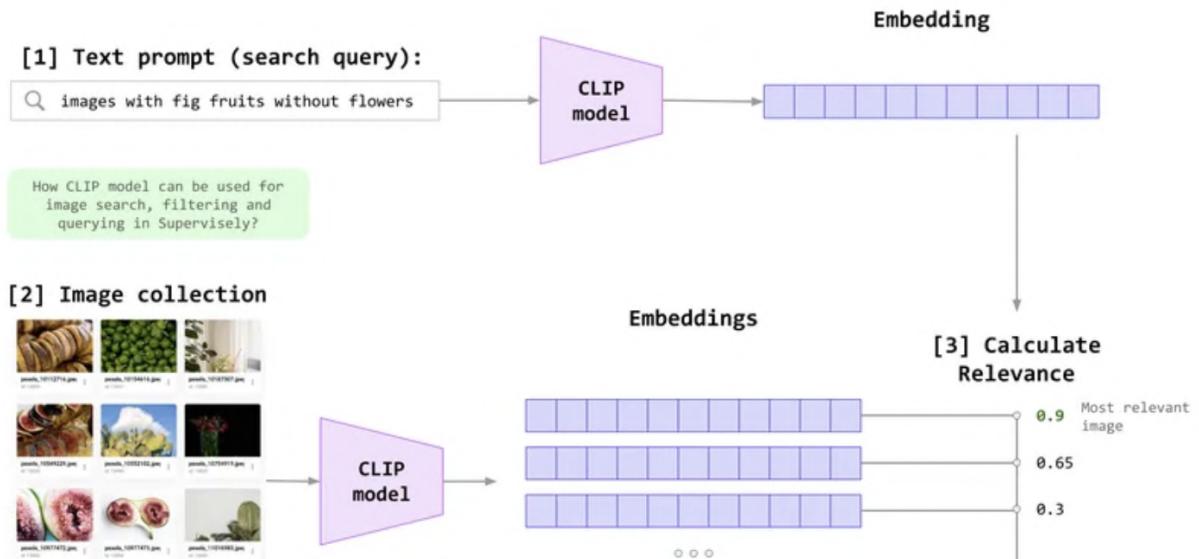


Figure 3: Illustration du fonctionnement d'un CNN

Principe de fonctionnement

CLIP repose sur l'utilisation d'un double encodeur : un pour les images et un pour les textes, chacun projetant ses entrées respectives dans un même espace sémantique. Le modèle a été entraîné via un apprentissage contrastif sur des centaines de millions de paires image-texte, apprenant à rapprocher les vecteurs d'éléments correspondants tout en éloignant ceux qui ne le sont pas. Lors d'une recherche, la requête textuelle est encodée en vecteur, puis comparée aux vecteurs d'images existants grâce à la similarité cosinus. Cette méthode permet de retrouver des images pertinentes même sans correspondance exacte de mots-clés.

Atouts pour le projet

L'utilisation de CLIP dans ce projet présente plusieurs avantages majeurs. Tout d'abord, il permet une recherche multimodale, offrant à l'utilisateur la possibilité de retrouver des images à partir d'une simple description, sans nécessiter une image requête. Ensuite, grâce à son entraînement massif, CLIP démontre une capacité impressionnante de généralisation, traitant efficacement des concepts nouveaux ou absents de son jeu de données d'entraînement (zero-shot learning).

Intégration dans le projet

Dans le cadre du projet, les embeddings d'images sont extraits à l'aide de la version ViT-B/32 de CLIP. Lors d'une requête textuelle, le texte est encodé puis comparé à l'ensemble des embeddings d'images pré-calculés. Les résultats sont classés selon leur similarité cosinus avec la requête et sont ensuite présentés à l'utilisateur via une interface développée sous Streamlit.

Limites et considérations

Néanmoins, l'utilisation de CLIP comporte aussi certaines contraintes. Le modèle est plus lourd et plus lent à l'inférence que MobileNetV3, ce qui peut être gênant pour des recherches massives, même s'il reste parfaitement utilisable pour des recherches ponctuelles. De plus, ses performances peuvent fluctuer en fonction de la proximité entre les données du projet et celles utilisées lors du pré-entraînement.

7.3 Conclusion sur l'articulation des deux modèles

En combinant MobileNetV3-Large pour la recherche d'images par le contenu (CBIR) et CLIP pour la recherche d'images par texte (TBIR), le projet bénéficie d'une solution robuste, flexible et performante. MobileNetV3 assure une extraction rapide et efficace des caractéristiques visuelles, idéale pour une recherche classique basée sur la similarité visuelle, tandis que CLIP étend considérablement les possibilités en permettant des recherches sémantiques multimodales. Cette complémentarité ouvre la voie à de nombreux cas d'usage avancés, enrichissant l'expérience de recherche au sein du système.

8 Présentation de FAISS

8.1 Introduction à FAISS

FAISS (Facebook AI Similarity Search) est une bibliothèque open source développée par Facebook AI Research, spécifiquement conçue pour la recherche rapide et efficace de similarité entre vecteurs de grande dimension. Son objectif est de permettre l'indexation et la recherche de plus proches voisins (nearest neighbors) dans des ensembles massifs de données, typiquement issus de modèles de deep learning comme les embeddings d'images ou de textes.

Contrairement aux moteurs de bases de données classiques, qui sont optimisés pour des recherches exactes sur des données structurées ou des index textuels, FAISS cible les scénarios où les objets à comparer sont représentés par des vecteurs denses dans un espace de grande dimension (par exemple, 128, 512 ou 1024 dimensions). Ce besoin est central dans la recherche d'images par le contenu (CBIR), où chaque image est résumée par un vecteur caractéristique extrait par un réseau de neurones.

8.2 Pourquoi utiliser FAISS ?

L'utilisation de FAISS s'impose dès que la volumétrie des données rend impossible une recherche exhaustive (brute-force) en temps raisonnable. Sur des bases de plusieurs dizaines ou centaines de milliers d'images, comparer chaque requête à l'ensemble des vecteurs serait trop lent et coûteux en mémoire. FAISS propose des structures d'indexation avancées et des algorithmes optimisés pour accélérer la recherche tout en maintenant un bon niveau de précision.

Les avantages majeurs de FAISS sont :

Scalabilité : capable de gérer des millions à des milliards de vecteurs, y compris sur des serveurs n'ayant pas assez de RAM pour tout stocker en mémoire.

Vitesse : recherche de plus proches voisins en quelques millisecondes grâce à des algorithmes spécialisés (quantification, clustering, graphes HNSW).

Flexibilité : prise en charge de différents types de métriques (distance euclidienne, similarité cosinus, produit scalaire) et de multiples stratégies d'indexation.

Compatibilité : intégration fluide avec Python et NumPy, support du GPU pour des accélérations supplémentaires.

8.3 Processus de création de l'index et préparation des données

Prétraitement et extraction des vecteurs

Avant d'utiliser FAISS, chaque image de la base est transformée en un vecteur dense (embedding) via un modèle de deep learning (MobileNetV3 ou CLIP dans ce projet). Ces vecteurs sont normalisés et rassemblés dans une matrice de dimension (N,d), où N est le nombre d'images et d la dimension de l'espace vectoriel.

Construction de l'index FAISS

FAISS propose plusieurs types d'index selon les besoins en rapidité, mémoire et précision :

Index brut (Flat) : stocke tous les vecteurs et effectue une recherche exhaustive (utile pour les petits jeux de données ou pour la validation).

Index avec quantification (Product Quantization, PQ) : compresse les vecteurs

pour réduire la mémoire tout en accélérant la recherche, avec une légère perte de précision.

Index à clusters (IVF, Inverted File Index) : regroupe les vecteurs en clusters via k-means, puis ne recherche que dans les clusters les plus proches lors d'une requête.

Graphes HNSW/NSG : structures de graphe pour des recherches ultra-rapides sur des bases très volumineuses.

Dans le cadre de notre projet, nous avons choisi d'utiliser l'**index brut** pour l'indexation des embeddings. Ce choix s'explique par la taille de nos bases de données qui reste gérable en mémoire sur une machine moderne, et par la volonté de garantir une recherche exacte des plus proches voisins, sans perte de précision due à l'approximation ou à la compression.

Concrètement, l'index est construit une fois pour toutes à partir des embeddings extraits, puis sauvegardé pour être chargé rapidement lors du lancement de l'application. Cette approche assure la reproductibilité des résultats et la robustesse du système, tout en offrant la possibilité d'évoluer vers des index plus complexes (IVF, PQ, HNSW) si la taille de la base venait à augmenter significativement à l'avenir.

8.4 Recherche de similarité et fonctionnement

Lorsqu'une requête (image ou texte encodé) est soumise, son embedding est extrait puis comparé à l'ensemble des vecteurs indexés dans FAISS. L'algorithme recherche les k plus proches voisins selon la métrique choisie (euclidienne ou cosinus).

FAISS optimise ce processus de plusieurs façons :

Recherche approximative : pour les très grands ensembles, FAISS peut sacrifier une faible part de précision pour gagner énormément en rapidité, ce qui est acceptable dans la plupart des applications CBIR où l'utilisateur attend des résultats pertinents, mais pas forcément exacts à 100%.

Compression des vecteurs : les méthodes de quantification et de clustering permettent de réduire la taille mémoire des index, rendant possible le traitement de bases qui ne tiendraient pas en RAM autrement.

Support GPU : FAISS peut exploiter les GPU pour accélérer la recherche, atteignant des vitesses jusqu'à 20 fois supérieures à l'utilisation CPU seule.

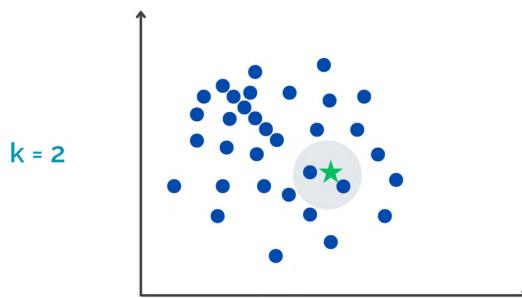


Figure 4: Illustration du fonctionnement d'une recherche

9 Extraction des vecteurs et stockage

L'extraction des vecteurs de caractéristiques, ou embeddings, est une étape centrale du projet L3E1. Elle permet de transformer chaque image en une représentation numérique compacte, adaptée à la recherche de similarité et à l'indexation efficace dans FAISS. Deux jeux de données ont été utilisés pour cette extraction : Tiny ImageNet et un sous-ensemble filtré d'Open Images. Cette section détaille le processus, les techniques de réduction de dimensionnalité, ainsi que les choix de format et de stockage adoptés.

9.1 Processus d'extraction des caractéristiques

L'extraction des caractéristiques repose sur l'utilisation de modèles de deep learning pré-entraînés, en particulier MobileNetV3, tronqué avant la couche de classification finale. Pour chaque image du dataset, le processus suit les étapes suivantes :

1. **Prétraitement** : Les images sont redimensionnées et normalisées pour correspondre aux exigences du modèle (par exemple, 224×224 pixels pour MobileNetV3). Ce prétraitement assure la cohérence des données en entrée et la reproductibilité des résultats.
2. **Passage dans le modèle** : L'image prétraitée est passée dans le réseau MobileNetV3, dont la dernière couche (classification) a été retirée. La sortie correspond alors à un vecteur dense de dimension fixe (embedding), qui capture les informations visuelles discriminantes de l'image.
3. **Extraction par lots** : Pour optimiser le temps de calcul, l'extraction est réalisée par lots (batch processing), en particulier lors de l'exécution sur Google Colab, qui fournit un accès à des GPU pour accélérer le traitement.
4. **Sauvegarde des embeddings** : Les vecteurs extraits sont sauvegardés dans des fichiers .npy (NumPy), accompagnés des catégories associées à chaque image pour faciliter l'évaluation et la recherche ultérieure.

Bien que deux jeux de données aient été utilisés, l'essentiel des tests et de la validation s'est concentré sur Tiny ImageNet, en raison de sa taille modérée et de sa diversité de classes, ce qui le rend adapté à l'expérimentation et à la démonstration du système. L'ajout du dataset Open Images s'est déroulé en fin de développement et de manière analogue, de sorte à améliorer significativement la qualité des images. Une différence notable est que les images d'Open Images, étant de meilleure qualité et donc plus lourdes que Tiny ImageNet, sont stockées sur un serveur gratuit Amazon S3 plutôt que localement.

9.2 Techniques de réduction de dimensionnalité utilisées

Dans le cadre du projet, la réduction de dimensionnalité n'est pas appliquée de manière algorithmique supplémentaire après l'extraction des embeddings par MobileNetV3. Les vecteurs de caractéristiques utilisés pour l'indexation et la recherche sont ceux produits directement par le modèle, sans transformation ultérieure par des techniques telles que l'Analyse en Composantes Principales (ACP/PCA), t-SNE ou UMAP. Ce choix s'explique par le fait que MobileNetV3-Large génère déjà des embeddings de dimension modérée (1280), ce qui permet d'obtenir un compromis satisfaisant entre richesse descriptive et

efficacité de traitement pour la recherche de similarité.

La seule forme de réduction mise en œuvre correspond à l'extraction des couches intermédiaires du réseau (juste avant la couche de classification), suivie d'un pooling adaptatif global, qui transforme chaque carte de caractéristiques en une valeur moyenne. Cette opération, réalisée lors de l'extraction par lots, permet de condenser l'information spatiale de l'image tout en conservant les propriétés discriminantes essentielles du vecteur.

Ce choix assure la cohérence, la simplicité et la rapidité du pipeline, tout en exploitant pleinement la capacité du modèle MobileNetV3 à générer des représentations visuelles compactes et informatives adaptées à la recherche d'images par le contenu.

9.3 Format et stockage des vecteurs

Les vecteurs de caractéristiques extraits sont stockés sous forme de fichiers **NumPy** (**.npy**), un format efficace pour la manipulation de grands tableaux numériques en Python. Chaque fichier contient un tableau de dimension (N, d) , où N est le nombre d'images et d la dimension des embeddings. Les catégories associées sont stockées dans des fichiers séparés, facilitant la correspondance entre vecteurs et classes d'images.

Pour Tiny ImageNet, les embeddings et les catégories sont stockés localement, ce qui permet une recherche rapide et une manipulation aisée lors des tests. Pour Open Images, en raison de la taille du dataset, seuls un sous-ensemble des images et leurs embeddings sont conservés, et les images elles-mêmes sont référencées par des URLs externes afin d'éviter une surcharge du stockage local.

Cette organisation permet :

- Un chargement rapide des embeddings lors du démarrage de l'application.
- Une compatibilité directe avec FAISS pour la création de l'index vectoriel.
- Une flexibilité pour l'ajout de nouveaux jeux de données ou de nouveaux modèles d'extraction.

10 Interface utilisateur

10.1 Choix de Streamlit pour le front-end

Pour rendre le système accessible à tous les utilisateurs, une interface web a été développée à l'aide de Streamlit, un framework Python moderne spécifiquement conçu pour créer rapidement des applications interactives de data science et de machine learning. Ce choix s'est imposé pour plusieurs raisons : simplicité de prise en main, intégration directe avec le code Python du backend, rapidité de prototypage, et ergonomie adaptée à des utilisateurs non spécialistes.

Streamlit permet de transformer un script Python en une application web dynamique, sans nécessiter de compétences avancées en développement front-end. Il offre des composants prêts à l'emploi (upload d'images, boutons, affichage de galeries, messages d'erreur, etc.), tout en autorisant la personnalisation de l'interface via du HTML et CSS intégré.

10.2 Organisation et navigation de l'application

L'interface est structurée autour d'un menu latéral, permettant à l'utilisateur de naviguer entre plusieurs pages principales :

- **Recherche par image (CBIR)** : L'utilisateur peut téléverser une image depuis son ordinateur, téléphone, ou tablette. L'application extrait automatiquement les caractéristiques visuelles de l'image, interroge l'index FAISS, puis affiche les images les plus similaires sous forme de galerie interactive.
- **Recherche par texte (TBIR)** : L'utilisateur saisit une requête textuelle (par exemple "un chat noir" ou "voiture rouge"). Le texte est encodé via le modèle CLIP, permettant de retrouver les images correspondantes dans la base.
- **Visualisation** : Des résultats tels que le graphe d'un vecteur de caractéristiques ou un graphique t-SNE sont affichés de façon claire et dynamique.
- **À propos** : Une page d'information présente le projet, les technologies utilisées et les instructions pour l'utilisation de l'application.

La navigation est fluide grâce à l'utilisation de composants interactifs (menus, boutons, barres de progression), et l'interface a été pensée pour être responsive et accessible sur différents supports (ordinateur, tablette, smartphone).

10.3 Fonctionnement technique et intégration au backend

Lorsqu'une requête est soumise (image ou texte), l'interface Streamlit communique directement avec les modules backend du projet :

- Pour la recherche par image, l'image uploadée est prétraitée, passée dans le modèle MobileNetV3 pour l'extraction des embeddings, puis ces derniers sont transmis au module de recherche pour interrogation de l'index FAISS.
- Pour la recherche par texte, la requête est encodée via CLIP, puis comparée aux embeddings d'images pour retrouver les résultats les plus pertinents.

- Les résultats (chemins ou URLs d’images, scores de similarité, catégories) sont ensuite affichés à l’utilisateur sous forme de galerie, avec un retour visuel immédiat.

L’ensemble du pipeline est optimisé pour garantir un temps de réponse inférieur à 5 secondes, même sur des bases de données volumineuses, conformément aux critères d’acceptabilité du cahier des charges.

10.4 Partage et accessibilité avec Ngrok

Pour permettre le test et la démonstration de l’application sans déploiement sur un serveur distant, Ngrok a été utilisé pour exposer localement l’application sur Internet. Ngrok crée un tunnel sécurisé entre le serveur local Streamlit et une URL publique, facilitant le partage de l’interface avec des encadrants, testeurs ou membres du jury, même à distance.

Cette solution a permis de valider l’accessibilité et la robustesse de l’application dans des conditions réelles, sans contrainte d’hébergement cloud coûteux ou complexe.

10.5 Perspectives d’évolution

Bien que Streamlit ait permis un développement rapide et efficace, des pistes d’amélioration sont envisageables : passage à un framework web plus avancé (React, Vue.js) pour une personnalisation accrue, développement d’une application mobile native, ou ajout de fonctionnalités avancées (filtres dynamiques, navigation par facettes, explication des résultats). L’architecture modulaire de l’application facilite ces évolutions futures.

10.6 Capture du rendu final

Afin d’illustrer concrètement l’ergonomie et les principales fonctionnalités de l’application web, cette section présente un capture d’écran de la page de recherche par le contenu. D’autres captures sont mises en annexes.

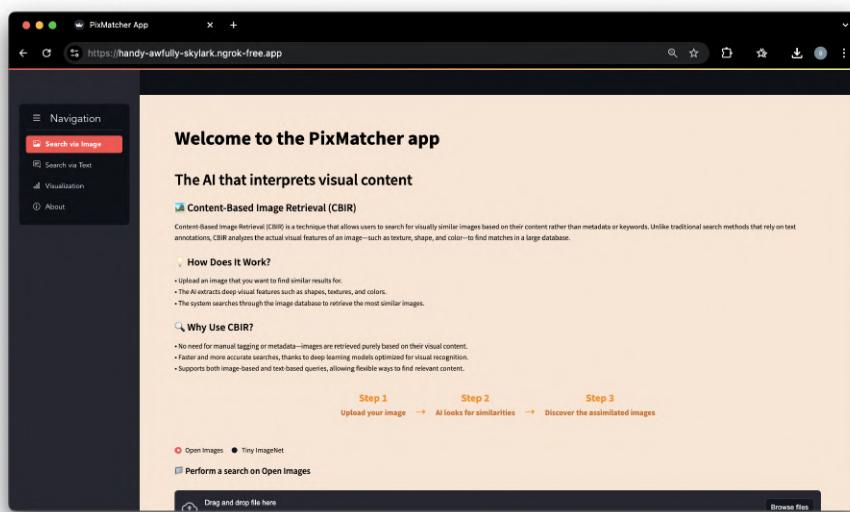


Figure 5: Page de recherche CBIR

11 Planification et répartition des tâches

11.1 Méthodologie et organisation générale

Le projet L3E1 a été conduit selon une approche **agile**, privilégiant l’itération rapide, la flexibilité et l’adaptation continue aux contraintes rencontrées. Cette méthode a permis de structurer le travail en cycles courts, avec des points réguliers de synchronisation et de réévaluation des priorités lors des réunions, tout en assurant une progression constante vers les objectifs définis.

Dès le lancement du projet, une vision globale a été établie, déclinée en jalons majeurs (sélection des modèles, développement des modules, tests, rédaction de la documentation, etc.). Chaque jalon a été découpé en tâches spécifiques, affectées à des membres de l’équipe en fonction de leurs compétences et de leurs disponibilités. Cette planification macroscopique a été affinée au fil du temps, au gré des retours d’expérience et des imprévus techniques ou organisationnels.

Les principaux avantages de cette organisation sont la capacité à réagir rapidement aux difficultés techniques ou aux changements de périmètre, la responsabilisation de chaque membre sur des tâches clairement identifiées, et enfin la traçabilité des décisions et des évolutions du projet.

11.2 Déroulement chronologique du projet

Le projet s'est articulé autour de plusieurs phases successives, chacune correspondant à des livrables et des objectifs précis :

1. **Définition des besoins et appropriation des technologies:** Analyse du cahier des charges, veille sur les modèles de deep learning, prise en main de PyTorch, FAISS et Streamlit. Cette phase s'est déroulée durant les semaines 1 et 2.
2. **Conception et planification détaillée :** Élaboration de l'architecture logicielle, découpage du projet en modules, rédaction des spécifications et du plan de tests. Cette phase s'est déroulée durant les semaines 3 et 4.
3. **Développement incrémental :** Implémentation progressive des modules (prétraitement, extraction de features, indexation, interface web), avec intégration et tests réguliers à chaque étape. Cette phase s'est déroulée pendant les semaines 5 à 9.
4. **Tests, validation et optimisation :** Vérification de la robustesse de chaque composant, mesure des performances, ajustements pour respecter les critères de temps de réponse et de précision. Cette phase s'est déroulée en parallèle du développement, durant les semaines 5 à 9.
5. **Documentation et finalisation :** Rédaction de la documentation technique et utilisateur, préparation de la soutenance et du rapport final. Cette phase s'est déroulée durant les semaines 9 à 12.

Chaque phase a donné lieu à des réunions d'équipe, des points d'avancement et des ajustements de la feuille de route, en tenant compte des contraintes de calendrier (examens, autres projets) et des retours de l'encadrant.

11.3 Décomposition des tâches et attribution des responsabilités

La réussite d'un projet repose sur une répartition claire et équilibrée des tâches, adaptée aux compétences et aux disponibilités de chacun. Pour chaque phase, les tâches ont été identifiées, priorisées et affectées explicitement :

Développement logiciel : Réalisation des modules principaux (CBIR, TBIR, indexation, interface), gestion des dépendances, intégration continue.

Extraction de features et gestion des datasets : Utilisation de Google Colab pour l'extraction des embeddings sur GPU, organisation et stockage des ressources.

Tests et validation : Élaboration et exécution de scénarios de tests, correction des bugs, amélioration de la robustesse.

Documentation et communication : Rédaction de la documentation interne, du rapport final, préparation des supports de présentation.

Cette organisation a permis d'éviter les chevauchements de responsabilités, d'optimiser l'efficacité collective et de garantir la livraison de chaque livrable dans les délais impartis.

11.4 Suivi de l'avancement et adaptation

Le suivi du projet s'est appuyé sur des outils partagés (tableaux de tâches, diagrammes de Gantt, listes de contrôle) et des indicateurs simples (taux d'avancement, nombre de tâches restantes, temps passé par module) via **la forge**. Des points de synchronisation hebdomadaires ont permis d'identifier rapidement les éventuels retards ou blocages, et d'ajuster la répartition des tâches en conséquence.

L'équipe a également pratiqué des rétrospectives régulières pour améliorer ses processus, renforcer la communication et anticiper les risques (surcharge, indisponibilités, etc.). Cette démarche a favorisé l'auto-organisation et l'implication de chacun dans la réussite globale du projet.

12 Difficultés rencontrées et solutions trouvées

Le développement du projet a été jalonné de défis techniques et organisationnels, nécessitant des adaptations méthodologiques et des choix pragmatiques pour garantir la réussite du livrable. Cette section détaille les principales difficultés rencontrées et les solutions mises en œuvre, en s'appuyant sur les retours d'expérience documentés tout au long du projet.

12.1 Prise en main des technologies avancées

Défi d'apprentissage

Un défi majeur rencontré au cours du projet a été la prise en main et l'appropriation de technologies complexes telles que MobileNetV3, CLIP et FAISS, qui étaient jusqu'alors inconnues de l'ensemble des membres de l'équipe. Ces outils, bien que puissants et largement utilisés dans l'industrie et la recherche, requièrent une compréhension fine de concepts avancés en deep learning, vision par ordinateur et recherche de similarité.

L'apprentissage de MobileNetV3 a nécessité de se familiariser avec les principes des réseaux de neurones convolutifs, la gestion des modèles pré-entraînés, ainsi que l'adaptation de ces modèles pour l'extraction de caractéristiques plutôt que pour la classification classique. Pour CLIP, il a fallu comprendre le fonctionnement des modèles multimodaux, l'encodage simultané du texte et de l'image dans un espace vectoriel partagé, et l'utilisation de ce modèle pour la recherche sémantique.

La prise en main de FAISS a également représenté un défi spécifique : il ne s'agit pas d'un simple moteur de recherche, mais d'une bibliothèque spécialisée dans l'indexation et la recherche de plus proches voisins dans des espaces de grande dimension. Il a fallu assimiler les notions de quantification, d'index inversé, de gestion de la mémoire et d'optimisation des requêtes pour garantir des performances à grande échelle.

Solution : auto-formation et prototypes

Pour surmonter ces obstacles, l'équipe a consacré une part significative des premières semaines du projet à l'auto-formation, à la lecture de documentations officielles, de tutoriels, d'articles scientifiques, ainsi qu'à la réalisation de prototypes expérimentaux sur des jeux de données réduits.

Cette démarche a permis de développer une intuition pratique du fonctionnement de ces outils, d'identifier les pièges courants (gestion des formats, compatibilité des versions, limitations matérielles), et de construire progressivement une architecture logicielle robuste et adaptée aux besoins du projet.

Ce processus d'apprentissage par la pratique, bien que chronophage, s'est révélé déterminant pour la réussite du projet. Il a permis à l'équipe de monter en compétence sur des technologies de pointe et de s'approprier des outils aujourd'hui incontournables dans le domaine de l'intelligence artificielle appliquée à la vision par ordinateur.

12.2 Extraction des vecteurs caractéristiques

L'extraction des vecteurs caractéristiques (*embeddings*) à partir des images a constitué l'un des points les plus critiques du projet, tant sur le plan du temps de calcul que de la gestion des ressources.

Problème de temps d'exécution et de ressources

L'extraction des embeddings à l'aide de modèles de deep learning (MobileNetV3 et CLIP) sur des jeux de données volumineux, comme Tiny ImageNet ou Open Images, s'est révélée coûteuse en temps et en puissance de calcul. L'exécution sur des machines personnelles, dépourvues de GPU performant, aurait nécessité plusieurs heures, voire jours, pour traiter l'ensemble des images, ce qui était incompatible avec les délais du projet.

Solution : utilisation de Google Colab

Pour contourner cette limitation, l'équipe a opté pour l'exécution des scripts d'extraction sur Google Colab, un environnement cloud offrant un accès gratuit à des GPU. Cette solution a permis de réduire le temps d'extraction à quelques heures pour Tiny ImageNet et Open Images, tout en automatisant le téléchargement, le prétraitement, l'extraction par lots et la sauvegarde des résultats. Les scripts ont été adaptés pour fonctionner de manière robuste sur Colab, avec gestion des lots et sauvegarde des embeddings sur Google Drive.

Gestion de la volumétrie d'Open Images

L'intégration du dataset Open Images a posé des difficultés supplémentaires: sa taille (plusieurs centaines de milliers d'images) rendait impossible l'extraction et le stockage local de l'ensemble des embeddings. Pour répondre à cette contrainte, un sous-ensemble filtré et redimensionné d'Open Images a été sélectionné, permettant de conserver une diversité d'images tout en restant dans des limites raisonnables de stockage et de traitement. De plus, les images Open Images n'étant pas hébergées localement mais sur un serveur externe, l'application utilise des liens directs pour l'affichage, évitant ainsi de surcharger le dépôt du projet et facilitant la diffusion des résultats.

12.3 Déploiement du site web

Impossibilité d'hébergement classique

Le déploiement d'une application web manipulant des ressources volumineuses (datasets, embeddings) s'est heurté à l'absence de solution gratuite ou simple pour héberger l'ensemble sur un cloud public. Les plateformes classiques ne permettaient pas d'intégrer à la fois l'application et les fichiers de grande taille nécessaires à son fonctionnement.

Solution : utilisation de Ngrok

Pour permettre les tests à distance et les démonstrations sans déployer sur un serveur distant, l'équipe a utilisé Ngrok, un outil qui crée un tunnel sécurisé entre le serveur local et Internet. Ngrok génère une URL publique, rendant l'application accessible à des testeurs ou encadrants extérieurs, tout en conservant les lourdes ressources en local. Cette solution a permis de valider l'accessibilité, la stabilité et la performance de l'application dans des conditions proches d'un déploiement réel, sans les contraintes d'un hébergement complet. Elle a également facilité la gestion de la confidentialité et la maîtrise des coûts.

12.4 Gestion des dépendances et portabilité

Bibliothèques spécialisées

Le projet s'appuie sur un ensemble de bibliothèques spécialisées (PyTorch, FAISS, Streamlit, etc.) dont la compatibilité et la configuration pouvaient varier selon les environnements

de développement. Des conflits de versions et des problèmes d'installation ont été rencontrés lors du passage d'un poste à l'autre.

Solution : environnement virtuel et documentation

Pour garantir la portabilité et la reproductibilité, un environnement virtuel Python a été systématiquement utilisé, les dépendances étant listées dans un fichier `requirements.txt`. Une documentation détaillée a été rédigée pour guider l'installation et la configuration, limitant ainsi les erreurs et facilitant la prise en main du projet par de nouveaux contributeurs.

12.5 Tests et validation

Validation du système

La validation du système sur des bases d'images volumineuses a nécessité la définition de scénarios de tests réalistes et la gestion de cas limites (images corrompues, formats non supportés, requêtes invalides). Des difficultés ont été rencontrées pour automatiser ces tests et garantir la robustesse du pipeline complet, notamment en raison de la diversité des modules (prétraitement, extraction, indexation, interface) et des interactions entre eux.

Solution : plan de tests structuré

Pour assurer la fiabilité et la qualité du système, une stratégie de validation rigoureuse a été mise en place, reposant sur deux axes complémentaires :

Tests unitaires systématiques : Chaque module du projet (prétraitement, extraction de caractéristiques, recherche de similarité, interface web, etc.) a fait l'objet de tests unitaires automatisés à l'aide de la bibliothèque `unittest` de Python. Ces tests, organisés dans le répertoire `test/`, vérifient le bon fonctionnement isolé de chaque composant, la gestion des entrées invalides, le respect des formats attendus et la robustesse face aux erreurs courantes. Les tests unitaires sont exécutés régulièrement à chaque modification du code, permettant de détecter rapidement les régressions ou les incompatibilités introduites lors du développement.

Tests d'intégration et validation fonctionnelle manuelle : Au-delà des tests automatisés, la qualité des résultats a été évaluée à la main par des membres de l'équipe, conformément au plan de tests fonctionnels à grande échelle. Cette validation qualitative a consisté à soumettre un large ensemble d'images et de requêtes textuelles au système, puis à juger la pertinence des résultats retournés selon des critères humains (similarité visuelle, cohérence sémantique, ...).

Gestion des erreurs et robustesse : Le système a été conçu pour afficher des messages d'erreur explicites et éviter tout crash, même en cas de requêtes anormales ou de ressources manquantes. Les tests ont inclus des scénarios de soumission d'images non supportées, de texte vide ou incohérent, et de surcharge du système pour garantir la stabilité de l'application dans toutes les situations.

13 Validation fonctionnelle et Résultats

13.1 Objectif et méthodologie des tests fonctionnels à grande échelle

Afin de garantir la conformité du système avec les exigences du cahier des charges et d'évaluer sa robustesse en conditions réelles, une campagne de tests fonctionnels à grande échelle a été menée sur l'ensemble de la chaîne applicative. Ces tests visent à valider non seulement la pertinence des résultats retournés par le moteur de recherche d'images, mais aussi les performances (temps de réponse, scalabilité) et la stabilité de l'application lors d'un usage intensif.

La méthodologie adoptée repose sur l'exécution de scénarios représentatifs d'utilisation réelle, en soumettant de nombreuses requêtes sur des bases d'images volumineuses (Tiny ImageNet et Open Images), avec une diversité de formats, de résolutions et de contenus. Les tests se sont concentrés sur deux axes principaux :

- **Recherche par image (CBIR)** : Simulation de requêtes utilisateurs par téléchargement d'images variées, analyse de la pertinence des résultats similaires proposés, mesure du temps de traitement pour chaque requête.
- **Recherche par texte (TBIR avec CLIP)** : Saisie de requêtes textuelles couvrant différents concepts, objets ou scènes, évaluation de la cohérence sémantique des images retournées, et vérification de la rapidité d'exécution.

Les tests ont été réalisés sur différents environnements (Windows, Linux, navigateurs variés) afin de garantir la portabilité et la robustesse du système dans des conditions d'utilisation hétérogènes.

13.2 Résultats des tests fonctionnels à grande échelle

Les résultats obtenus lors de cette campagne de tests sont globalement très satisfaisants et démontrent la solidité de l'architecture mise en place.

Recherche par image (CBIR)

- **Pertinence des résultats** : Pour la grande majorité des requêtes, les images retournées par le système étaient effectivement très similaires à l'image requête, tant sur le plan visuel (formes, couleurs, objets présents) que sémantique. Les erreurs de classement ou les cas de confusion sont restés marginaux, principalement sur des images très ambiguës ou appartenant à des classes visuellement proches. En chiffres, ce sont **82%** des premières images retournées qui sont jugées visuellement très proches de l'image de requête et **76%** des dix premières.
- **Performance et temps de réponse** : Le temps de traitement moyen pour une recherche sur la base Tiny ImageNet (plus de 100 000 images) s'est systématiquement maintenu sous la barre des 3 secondes. Ces résultats respectent largement la contrainte de moins de 5 secondes fixée dans le cahier des charges.
- **Robustesse** : Le système a correctement géré les cas d'images corrompues, de formats non supportés ou de requêtes anormales, affichant des messages d'erreur explicites sans provoquer de crash de l'application.

Recherche par texte (TBIR avec CLIP)

- **Cohérence sémantique** : Les requêtes textuelles ("un chien", "une voiture rouge", "un paysage de montagne") ont permis de retrouver des images pertinentes, même lorsque le vocabulaire utilisé différait de celui des catégories du dataset. La capacité du modèle CLIP à faire le lien entre concepts textuels et visuels s'est révélée efficace, y compris pour des requêtes complexes ou ambiguës. En chiffres, ce sont **78%** des premières images retournées qui sont jugées pertinentes et **70%** des dix premières.
- **Temps de réponse** : Les recherches textuelles ont affiché des temps de réponse légèrement supérieurs à ceux de la recherche par image, mais sont restées dans des délais acceptables (en moyenne 4 secondes). L'expérience utilisateur s'est avérée fluide et satisfaisante.
- **Gestion des cas limites** : Les requêtes textuelles vides, incohérentes ou contenant des caractères spéciaux ont été traitées de manière robuste, avec une gestion des erreurs appropriée et sans impact sur la stabilité globale.

Scalabilité et stabilité

- **Scalabilité** : L'application a démontré sa capacité à gérer efficacement des bases d'images de grande taille, sans dégradation significative des performances lors de recherches simultanées ou répétées.
- **Stabilité** : Aucun crash ni fuite mémoire n'a été observé lors de tests prolongés ou de scénarios de charge élevée. L'interface utilisateur est restée réactive et accessible tout au long de la campagne.

13.3 Exemples de résultats

Afin d'illustrer concrètement la pertinence et la robustesse du système développé, nous présentons ci-dessous plusieurs exemples de résultats obtenus lors de recherches par image et par texte. Ces exemples permettent de visualiser la capacité du moteur à retrouver des images similaires ou sémantiquement cohérentes à partir de requêtes variées, et de mettre en lumière les forces et les éventuelles limites du système dans des situations réelles d'utilisation.

Tout d'abord, résultats de requête pour le CBIR :

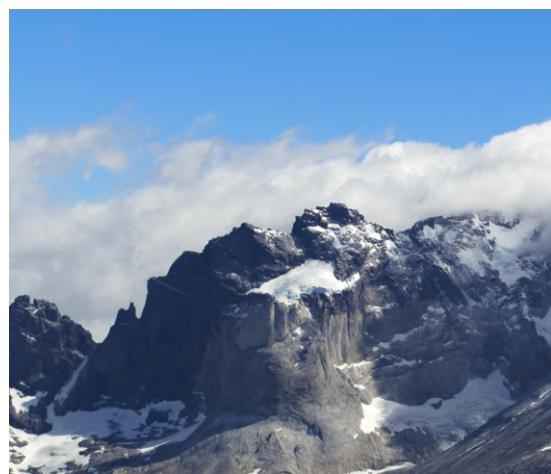


Figure 6: Requête via l'image d'une montagne enneigée

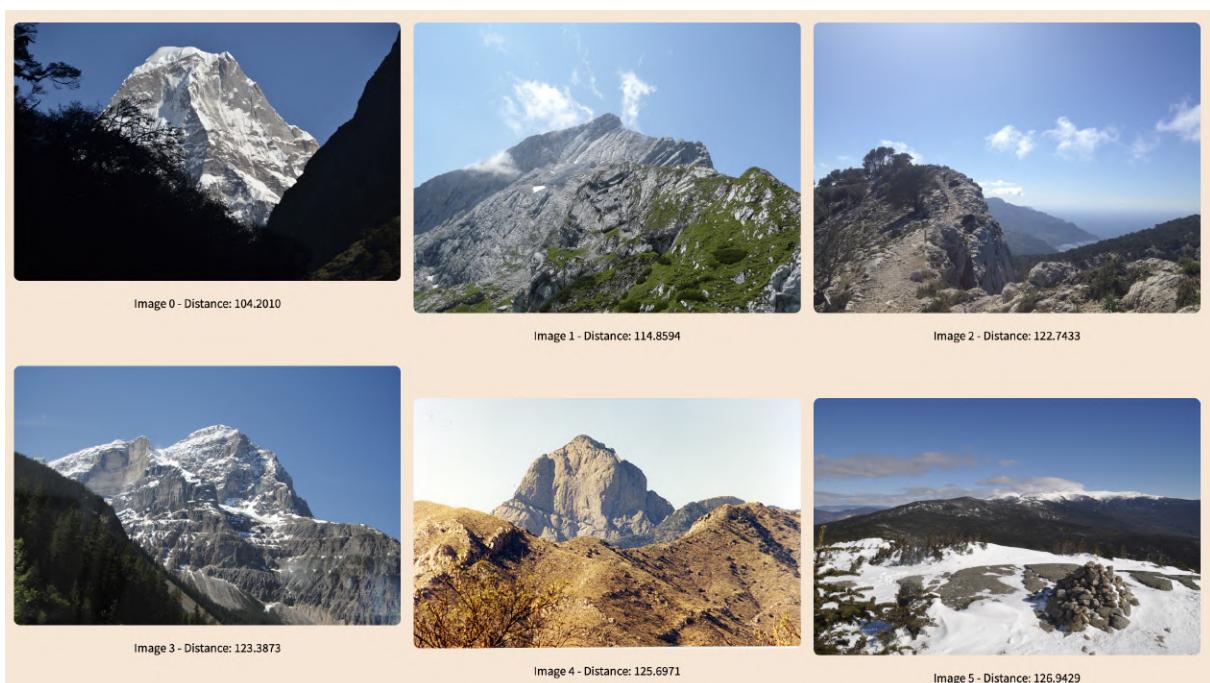


Figure 7: Exemple de résultats



Figure 8: Requête via l'image d'un chat roux

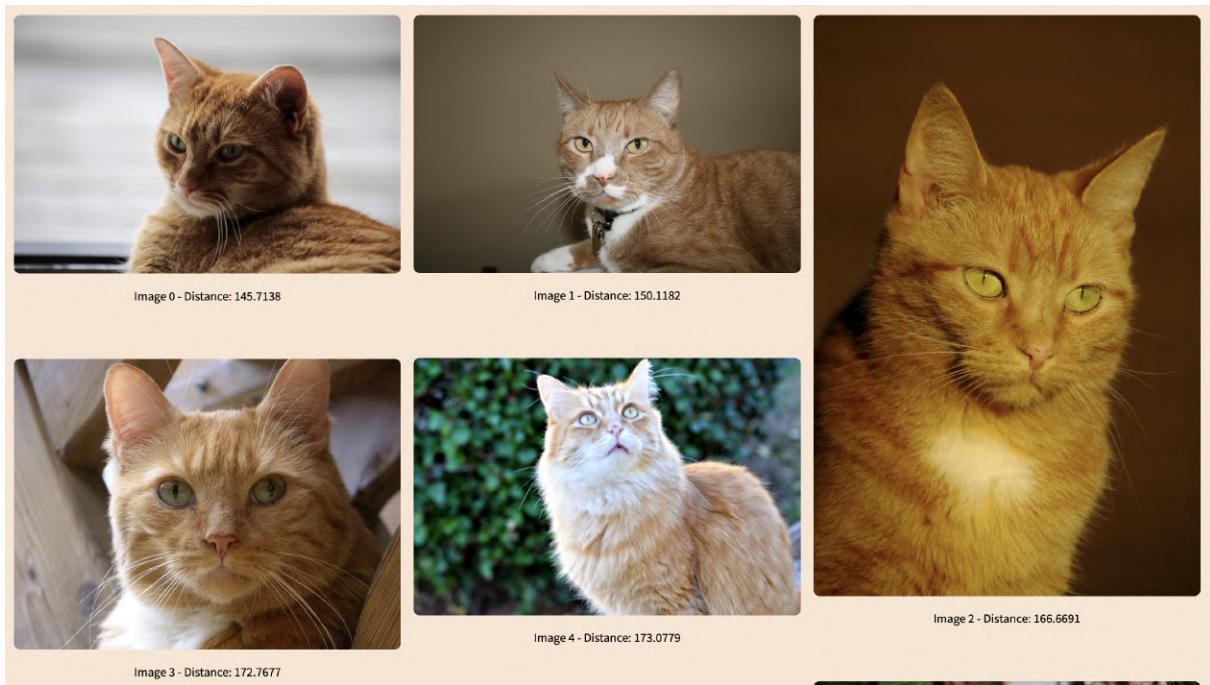


Figure 9: Exemple de résultats



Figure 10: Requête via l'image d'un ordinateur

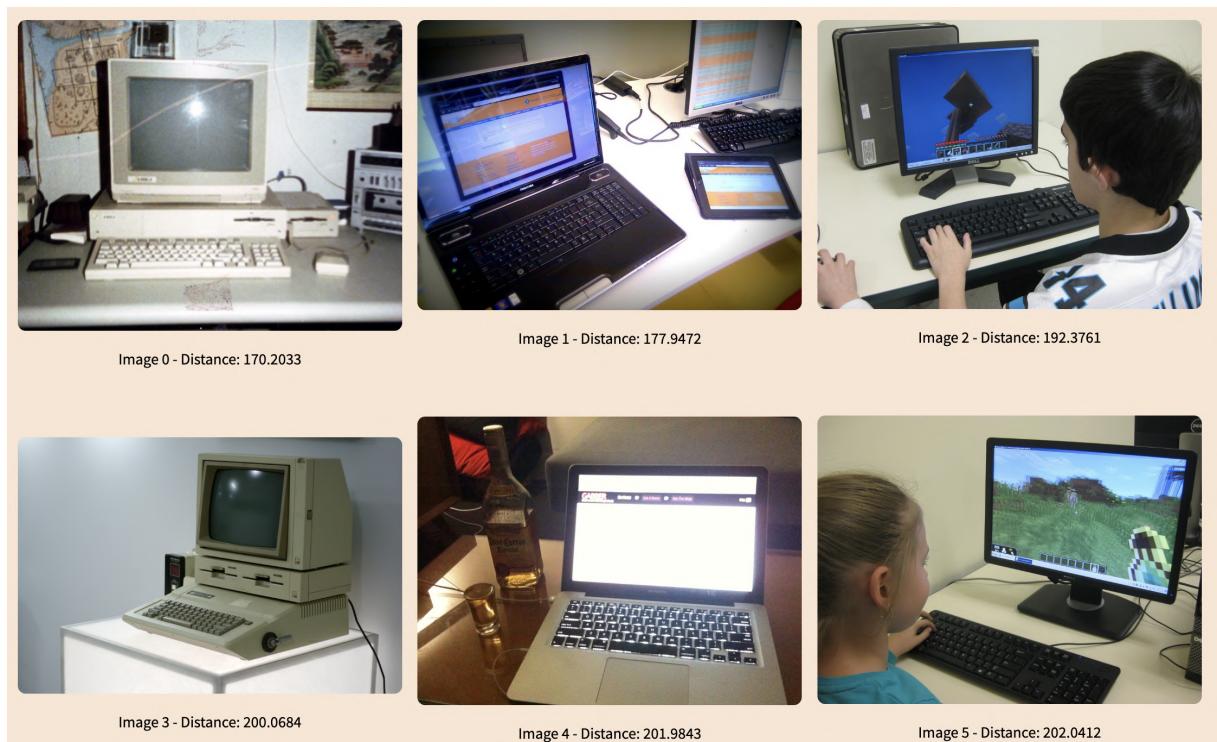


Figure 11: Exemple de résultats

Résultats de requête pour le TBIR :

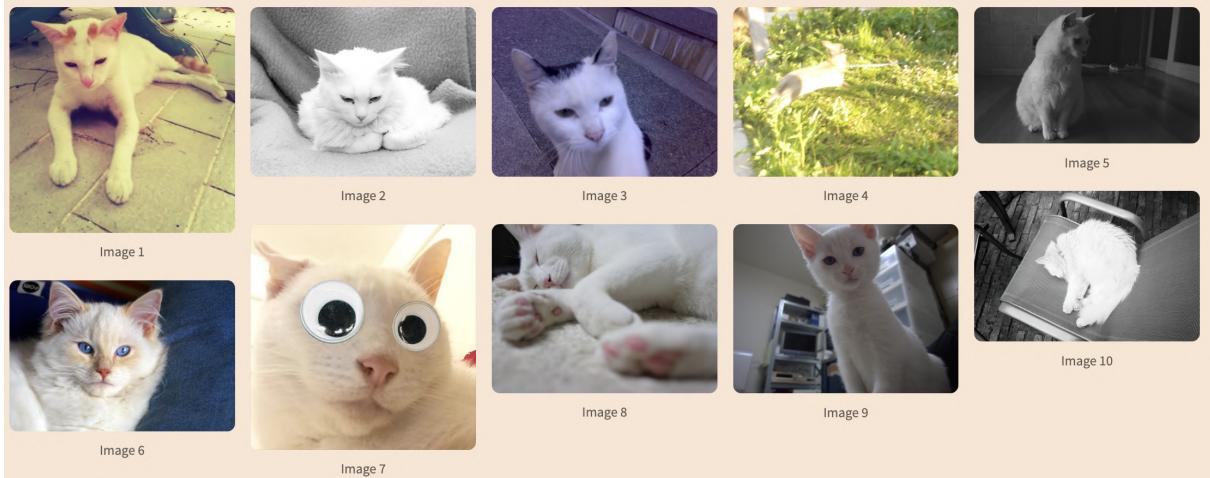


Figure 12: Résultat pour la requête ”a white cat”



Figure 13: Résultat pour la requête ”a bar”

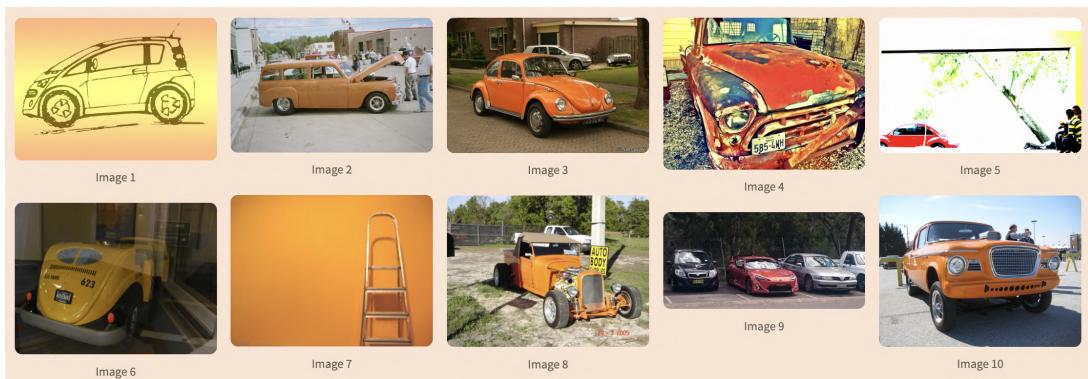


Figure 14: Résultat pour la requête ”an orange car”

14 PixMatcher après la soutenance

14.1 Perspectives d'évolution

Le projet PixMatcher, tel qu'il a été conçu, constitue une base robuste et modulaire pour la recherche d'images par le contenu. Cependant, de nombreuses pistes d'évolution et d'enrichissement sont envisageables afin d'ouvrir le système à de nouveaux usages, d'améliorer ses performances et de répondre à des besoins plus larges.

14.1.1 Prise en charge d'autres bases de données

L'une des premières extensions naturelles du projet consiste à intégrer de nouveaux jeux de données, au-delà de Tiny ImageNet et du sous-ensemble d'Open Images actuellement supportés. L'architecture du système, fondée sur des modules indépendants pour l'extraction des caractéristiques et l'indexation, permet d'ajouter facilement d'autres bases, qu'elles soient publiques (COCO, ImageNet complet, bases médicales, collections patrimoniales) ou privées (archives d'entreprise, collections personnelles). Cette ouverture répond à la diversité croissante des besoins en indexation d'images, dans des domaines aussi variés que la santé, la culture, la sécurité ou le commerce en ligne. L'intégration de nouveaux datasets nécessitera toutefois d'adapter les scripts d'extraction, de gérer l'hébergement des ressources volumineuses et de garantir la compatibilité des formats d'annotations.

14.1.2 Comparaison et intégration de nouveaux modèles

Le système actuel repose sur MobileNetV3 pour le CBIR et CLIP pour le TBIR, mais il existe une grande variété de modèles de deep learning, chacun présentant des avantages et des contraintes spécifiques selon la nature des images, la volumétrie des données et les ressources matérielles disponibles. Parmi ces alternatives, on peut citer des architectures comme ResNet, EfficientNet, Vision Transformer (ViT), ou encore des modèles spécialisés pour certains domaines (médical, satellite, etc.).

À l'avenir, une démarche comparative systématique serait pertinente : il s'agirait de mesurer les performances de ces modèles sur des jeux de données variés, en évaluant non seulement la précision des résultats (taux de pertinence Top-1/Top-10), mais aussi le temps de réponse, la consommation mémoire, et la facilité d'intégration dans l'architecture existante. Il est important de souligner que les modèles les plus récents ou les plus performants (comme ViT ou EfficientNet-L2) sont souvent beaucoup plus gourmands en ressources : ils nécessitent davantage de mémoire, de puissance de calcul (GPU haut de gamme), et peuvent entraîner des temps d'inférence plus longs. Ce surcoût matériel doit être mis en balance avec le gain de précision, surtout dans un contexte où l'expérience utilisateur impose des réponses rapides (moins de 5 secondes).

Une piste d'évolution intéressante serait d'envisager une architecture hybride, où plusieurs modèles cohabiteraient : le système pourrait sélectionner dynamiquement le modèle le plus adapté selon la nature de la requête (image simple, scène complexe, domaine spécialisé) ou selon les contraintes de ressources du serveur. Cela permettrait d'optimiser en continu le ratio précision/rapidité et de s'adapter à des contextes d'utilisation très variés. Enfin, l'intégration de modèles open source plus récents ou de modèles propriétaires

spécialisés pourrait également être envisagée, à condition de vérifier leur compatibilité avec l'infrastructure logicielle et les contraintes de licence.

14.1.3 Amélioration de l'interface et de l'expérience utilisateur

L'interface Streamlit, bien que fonctionnelle et rapide à déployer pour un prototype ou une démonstration, présente certaines limites dès lors que l'on vise une expérience utilisateur avancée ou une diffusion à grande échelle. Plusieurs axes d'amélioration peuvent être envisagés pour enrichir l'ergonomie, l'accessibilité et la personnalisation du système.

D'une part, l'ajout de fonctionnalités avancées comme des filtres dynamiques (par couleur dominante, type d'objet détecté, niveau de similarité), une navigation par facettes, ou la visualisation interactive des résultats (zoom, tri, regroupement par catégories) permettrait d'offrir une expérience plus riche et adaptée à des usages professionnels ou grand public. L'intégration de modules d'explication des résultats (explainability), par exemple via la mise en avant des régions de l'image ayant motivé la similarité, renforcerait la transparence et la confiance des utilisateurs.

D'autre part, pour dépasser les limites de Streamlit en termes de personnalisation et de scalabilité, il serait pertinent d'envisager un déploiement sur des frameworks web modernes tels que React, Angular ou Vue.js, en connectant l'interface à un backend Python via une API. Une telle architecture permettrait de proposer une interface plus fluide, responsive, et facilement déclinable sur différents supports (desktop, tablette, mobile). À plus long terme, le développement d'une application mobile native (Android/iOS) pourrait également être envisagé, afin de permettre la recherche d'images par le contenu directement depuis l'appareil photo ou la galerie de l'utilisateur, et d'ouvrir le système à de nouveaux usages nomades.

Enfin, l'implication continue des utilisateurs finaux dans la conception et l'évaluation de l'interface restera essentielle pour garantir l'adéquation du système aux besoins réels et pour guider les évolutions fonctionnelles et ergonomiques du projet.

14.2 Diffusion et valorisation du projet

14.2.1 Utilisateurs projetés et cas d'usage

Le système développé vise une large gamme d'utilisateurs potentiels, reflétant la polyvalence des technologies de recherche d'images par le contenu. Parmi les publics ciblés, on compte :

- **Les chercheurs et ingénieurs en vision par ordinateur** : ils peuvent utiliser la plateforme comme base d'expérimentation pour tester de nouveaux modèles ou algorithmes d'indexation.
- **Les professionnels de l'information** (bibliothèques, musées, centres d'archives ou institutions patrimoniales) : l'indexation automatique et la recherche rapide d'images similaires facilitent la gestion, l'enrichissement et la valorisation de fonds iconographiques volumineux.

-
- **Les entreprises** : souhaitant organiser, explorer ou valoriser leurs bases iconographiques internes (e-commerce, communication, médias, design, marketing visuel) en automatisant le classement ou la recommandation de contenus visuels.
 - **Les enseignants et étudiants** : ils peuvent s'appuyer sur le système pour illustrer des concepts de deep learning, d'indexation ou de recherche d'information lors de travaux pratiques ou de projets pédagogiques.
 - **Les particuliers** désireux de classer, retrouver ou explorer leurs collections de photos personnelles : en bénéficiant d'une recherche intuitive et sans nécessité d'annotation manuelle.

L'ouverture du code source, la modularité de l'architecture et la documentation détaillée facilitent la réutilisation et l'adaptation du système à des contextes variés, que ce soit pour l'intégration à des workflows existants, l'extension à de nouveaux types de données (vidéo, audio) ou l'ajout de fonctionnalités spécifiques (filtres avancés, analyse sémantique, etc.).

Pour garantir la pertinence et l'utilité du système, il sera essentiel de recueillir les retours d'utilisateurs réels, issus de ces différents milieux, afin de guider les évolutions fonctionnelles, ergonomiques et techniques du projet. La mise en place d'un canal de retour (formulaire, issues GitHub, enquêtes) permettra d'identifier rapidement les besoins émergents, les difficultés rencontrées et les pistes d'amélioration prioritaires.

14.2.2 Politique de diffusion et gestion des droits d'auteur

Le logiciel développé dans le cadre de ce projet est diffusé librement, sans restriction d'usage, afin de maximiser sa portée scientifique, pédagogique et sociétale. Les utilisateurs sont ainsi libres d'utiliser, modifier, adapter ou intégrer le système dans leurs propres projets, qu'ils soient académiques, professionnels ou personnels. Cependant, il convient de rappeler que le projet s'appuie sur un ensemble de bibliothèques et de modèles tiers :

- **MobileNetV3** (Google) : modèle sous licence Apache 2.0, utilisé pour l'extraction des caractéristiques visuelles des images.
- **CLIP** (OpenAI) : modèle multimodal sous licence MIT, utilisé pour la recherche textuelle.
- **FAISS** (Meta/Facebook AI Research) : bibliothèque sous licence MIT, utilisée pour l'indexation et la recherche de similarité à grande échelle.

L'utilisation de ces composants implique le respect des licences associées, qui autorisent la réutilisation mais peuvent imposer certaines obligations (citation, mention des auteurs, interdiction d'usage commercial dans certains cas). Les jeux de données utilisés (Tiny ImageNet, Open Images) sont également soumis à leurs propres conditions d'utilisation, qu'il appartient à chaque utilisateur de respecter en fonction de son contexte d'exploitation.

En ce qui concerne le logiciel développé par l'équipe du projet L3E1, aucune restriction particulière n'est imposée : le code est mis à disposition en l'état, sans garantie ni responsabilité quant à l'usage qui pourrait en être fait. Chaque utilisateur est donc responsable de l'utilisation, de la modification ou de la diffusion du logiciel dans le respect

des lois en vigueur et des licences des composants tiers. Il est recommandé de conserver dans toute redistribution les mentions d'origine et les références aux auteurs du projet ainsi qu'aux auteurs des bibliothèques utilisées.

Cette politique d'ouverture vise à encourager la collaboration, la transparence et l'innovation, tout en rappelant la nécessité d'un usage responsable et respectueux des droits de chacun dans l'écosystème logiciel et scientifique.

15 Conclusion

Le projet L3E1 a permis de concevoir, développer et valider un système complet de recherche d’images par le contenu, s’appuyant sur les avancées récentes du deep learning et de l’indexation vectorielle. Face à l’explosion des données visuelles et aux limites des approches traditionnelles fondées sur les métadonnées textuelles, ce travail apporte une réponse innovante et performante, capable d’analyser automatiquement les caractéristiques visuelles des images et de proposer une recherche rapide, précise et multimodale.

Tout au long du projet, l’équipe a su relever des défis techniques majeurs : prise en main de technologies avancées (MobileNetV3, CLIP, FAISS), gestion de jeux de données volumineux, optimisation des flux de traitement, et déploiement d’une interface utilisateur accessible. L’approche modulaire retenue, la rigueur des tests (unitaires, fonctionnels, à grande échelle) et l’attention portée à la documentation garantissent la robustesse, la maintenabilité et l’évolutivité du système.

Les résultats obtenus lors des campagnes de validation montrent que le système répond pleinement aux critères d’acceptabilité définis dans le cahier des charges : pertinence des résultats, temps de réponse inférieur à 3 secondes sur des bases de 100 000 images, robustesse face aux erreurs et expérience utilisateur fluide. La double interface de recherche, par image et par texte, ouvre la voie à des usages variés, du diagnostic médical assisté à l’organisation de collections patrimoniales, en passant par la recommandation de produits ou la gestion d’archives personnelles.

Ce projet constitue ainsi un socle solide pour de futurs développements. Les perspectives d’évolution sont nombreuses : intégration de nouveaux modèles ou jeux de données, enrichissement de l’interface (web moderne, application mobile), amélioration de l’explicabilité des résultats, ou encore adaptation à des contextes spécifiques (santé, industrie, enseignement). La diffusion libre du code, dans le respect des licences des composants tiers, encourage la réutilisation, la collaboration et l’innovation au sein de la communauté scientifique et technique.

Au-delà de la réalisation technique, cette expérience a été l’occasion pour l’équipe de se former à la gestion de projet, au travail collaboratif, à la conduite de tests rigoureux et à la rédaction de documents professionnels. Elle illustre l’importance de la synergie entre rigueur scientifique, innovation technologique et esprit d’équipe pour relever les défis posés par la société numérique contemporaine.

En définitive, le projet L3E1 démontre la faisabilité et l’intérêt des solutions de recherche d’images par le contenu fondées sur l’intelligence artificielle, et pose les bases d’outils évolutifs, ouverts et adaptés aux besoins croissants de gestion des données visuelles à grande échelle.

16 Bibliographie et ressources

Avant de présenter les références détaillées, il convient de signaler que la réalisation de ce projet s'est appuyée sur un ensemble varié de ressources : documentations officielles des bibliothèques, articles scientifiques, tutoriels en ligne, jeux de données publics et ouvrages de référence. Les principales sources consultées et utilisées pour le développement, la compréhension et la validation du système sont listées ci-dessous via des **liens hypertextes**.

Réseaux de neurones convolutifs et Deep Learning

- Convolutional neural network - Wikipedia
- What are Convolutional Neural Networks? - IBM
- Convolutional Neural Network: Everything You Need to Know - DataScientest
- Convolutional Neural Networks, Explained - Towards Data Science
- Krizhevsky, A., Sutskever, I., Hinton, G. E., *ImageNet Classification with Deep Convolutional Neural Networks*, NeurIPS, 2012.
- Simonyan, K., Zisserman, A., *Very Deep Convolutional Networks for Large-Scale Image Recognition*, ICLR, 2015.
- He, K., Zhang, X., Ren, S., Sun, J., *Deep Residual Learning for Image Recognition*, CVPR, 2016.

MobileNet

- Howard, A. et al., *Searching for MobileNetV3*, ICCV, 2019.
- Zhang, X. et al., *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*, CVPR 2018.
- Documentation MobileNetV3

CLIP (Contrastive Language-Image Pretraining)

- Radford, A., Kim, J. W., Hallacy, C., et al., *Learning Transferable Visual Models From Natural Language Supervision*, ICML 2021.
- Documentation CLIP
- open-clip-torch (GitHub)

FAISS (Facebook AI Similarity Search)

- Johnson, J., Douze, M., Jégou, H., *Billion-scale similarity search with GPUs*, IEEE Transactions on Big Data, 2021.
- Documentation FAISS
- FAISS (GitHub)

Datasets

- **Tiny ImageNet** : Documentation Tiny ImageNet
- **Open Images** : Documentation Open Images Dataset

Documentation

Tous ces documents ont été rédigés par l'équipe du projet et déposés sur la forge, garantissant leur accessibilité, leur traçabilité et leur mise à jour tout au long du développement. Ils constituent la référence officielle pour la validation, la maintenance et l'évolution future du système.

- Cahier des charges
- Cahier de recette
- Plan de développement
- Plan de tests
- Documentation interne
- Manuel d'utilisation
- Manuel d'installation

17 Annexe

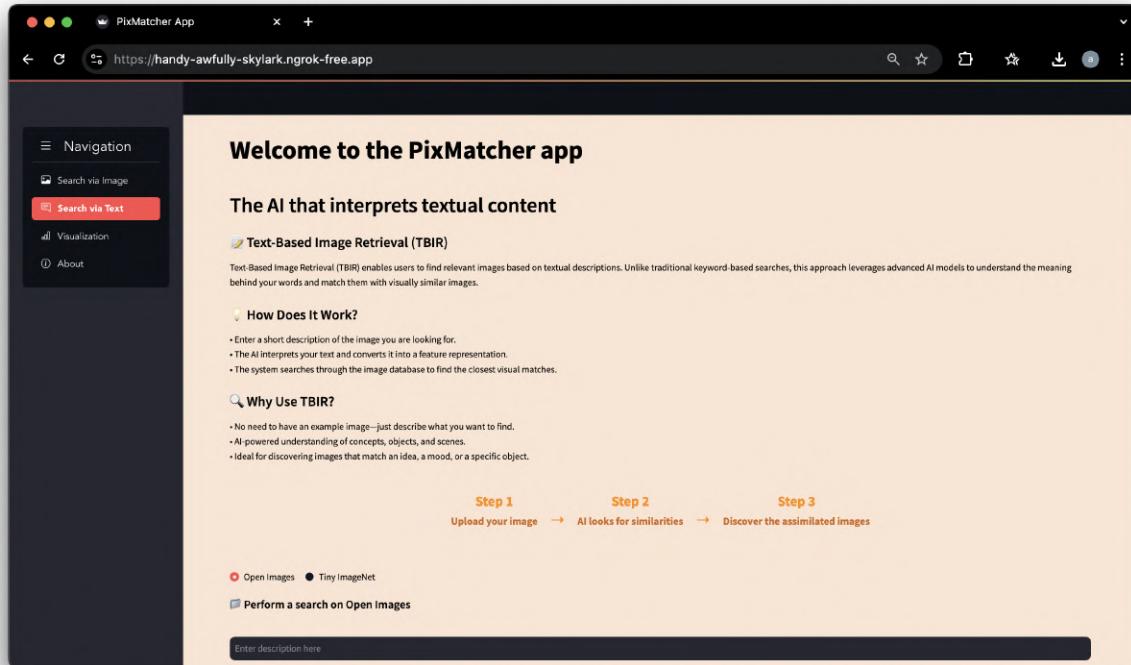


Figure 15: Page de recherche TBIR du site

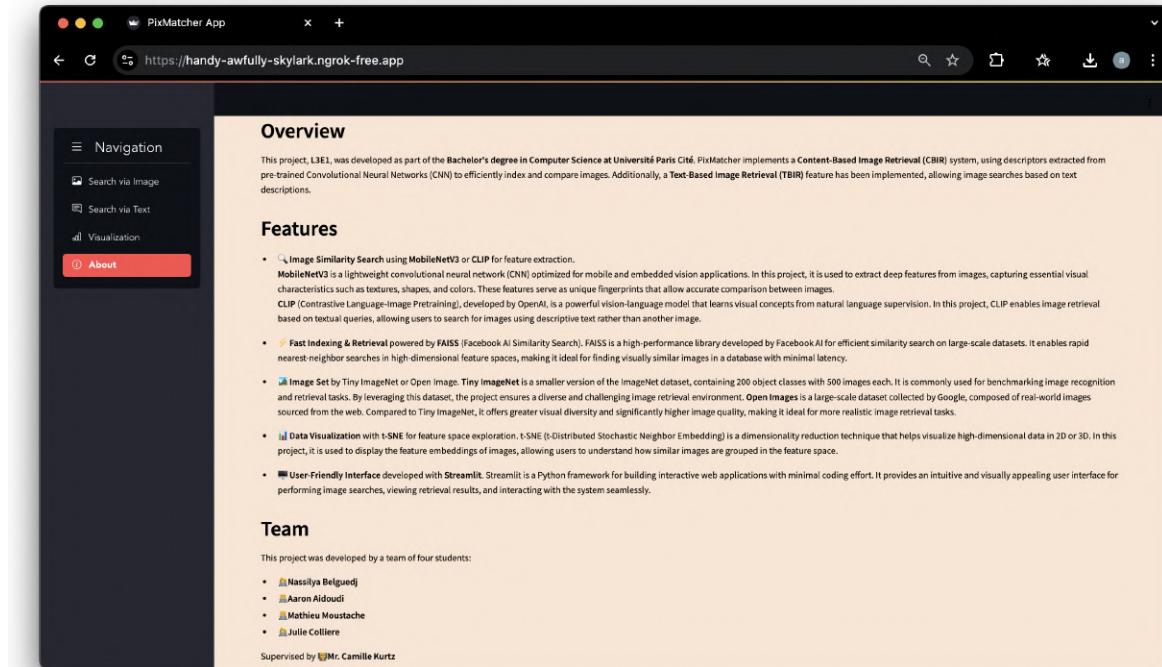


Figure 16: Extrait de la page À propos du site

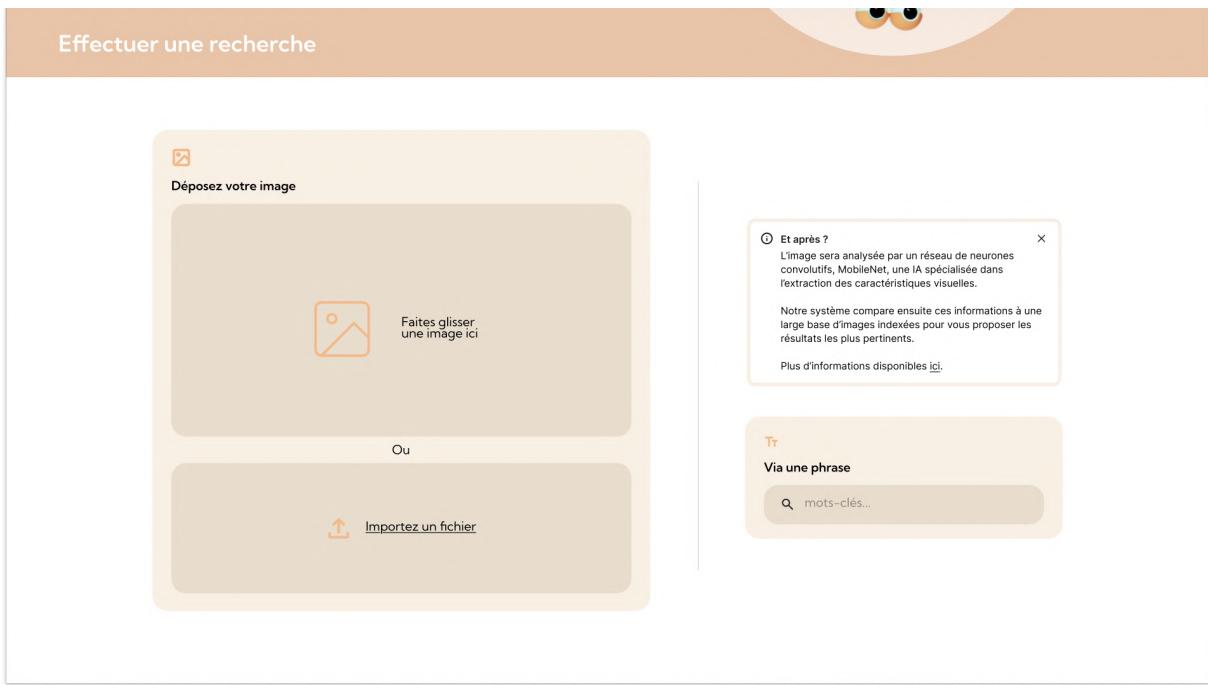


Figure 17: Maquette de la page de recherche CBIR

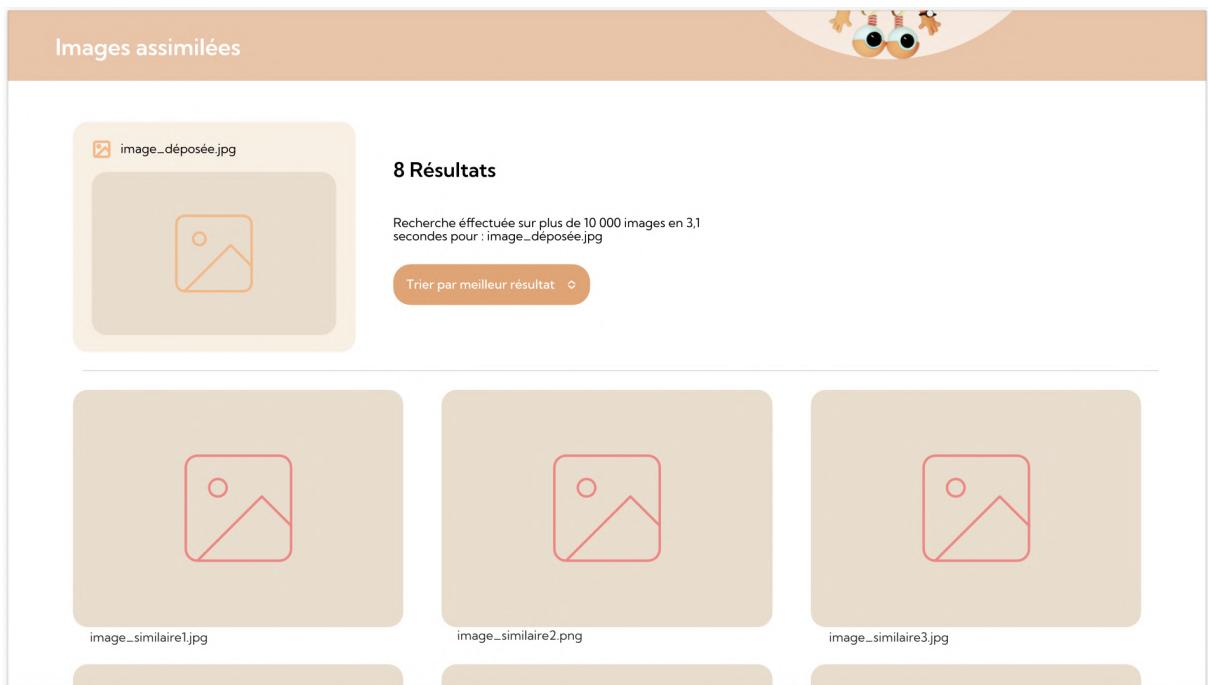


Figure 18: Maquette de la page de résultats CBIR