



IMAGE RETRIEVAL

Giảng viên: Hoàng Anh Đức

Học phần: Nhập môn trí tuệ nhân tạo

Mã học phần: MAT3508

Học kỳ: Học kỳ 1 - 2025-2026

Ngày nộp: 04/12/2025

OUR TEAM

 Họ và tên 	 Mã sinh viên 	 Tên GitHub 
Nguyễn Ngọc Quân	23001552	NguyenQuan1811
Nguyễn Ngọc Trường	23001566	nntruong-ds
Đinh Văn Thiêm	23001560	thiemhus
Đỗ Đình Tuyến	23001568	tuyendodinh
Nguyễn Văn Bách	23001499	I34cH-VN



OVERVIEWS

01

MỞ ĐẦU

02

PHƯƠNG PHÁP

03

TRIỂN KHAI

04

KẾT QUẢ

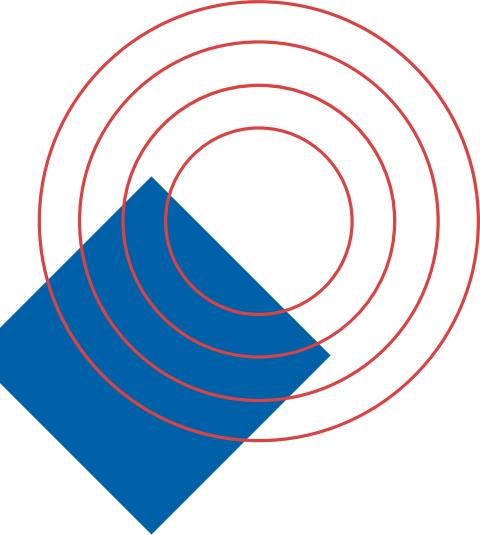
05

KẾT LUẬN, HƯỚNG PHÁT TRIỂN

06

HT WEB + AI

CHƯƠNG 1: MỞ ĐẦU



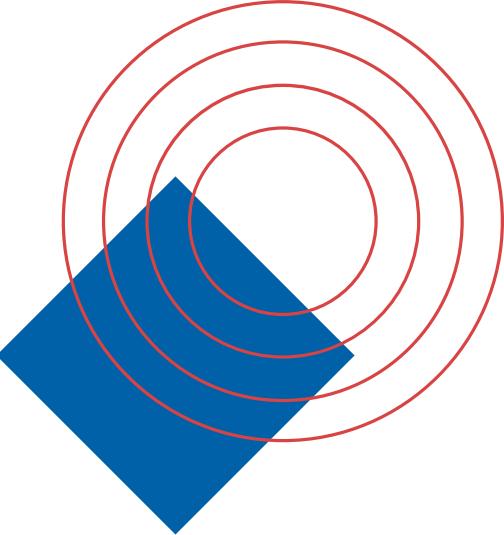
1.1 Tóm tắt dự án

1.2 Bài toán đặt ra

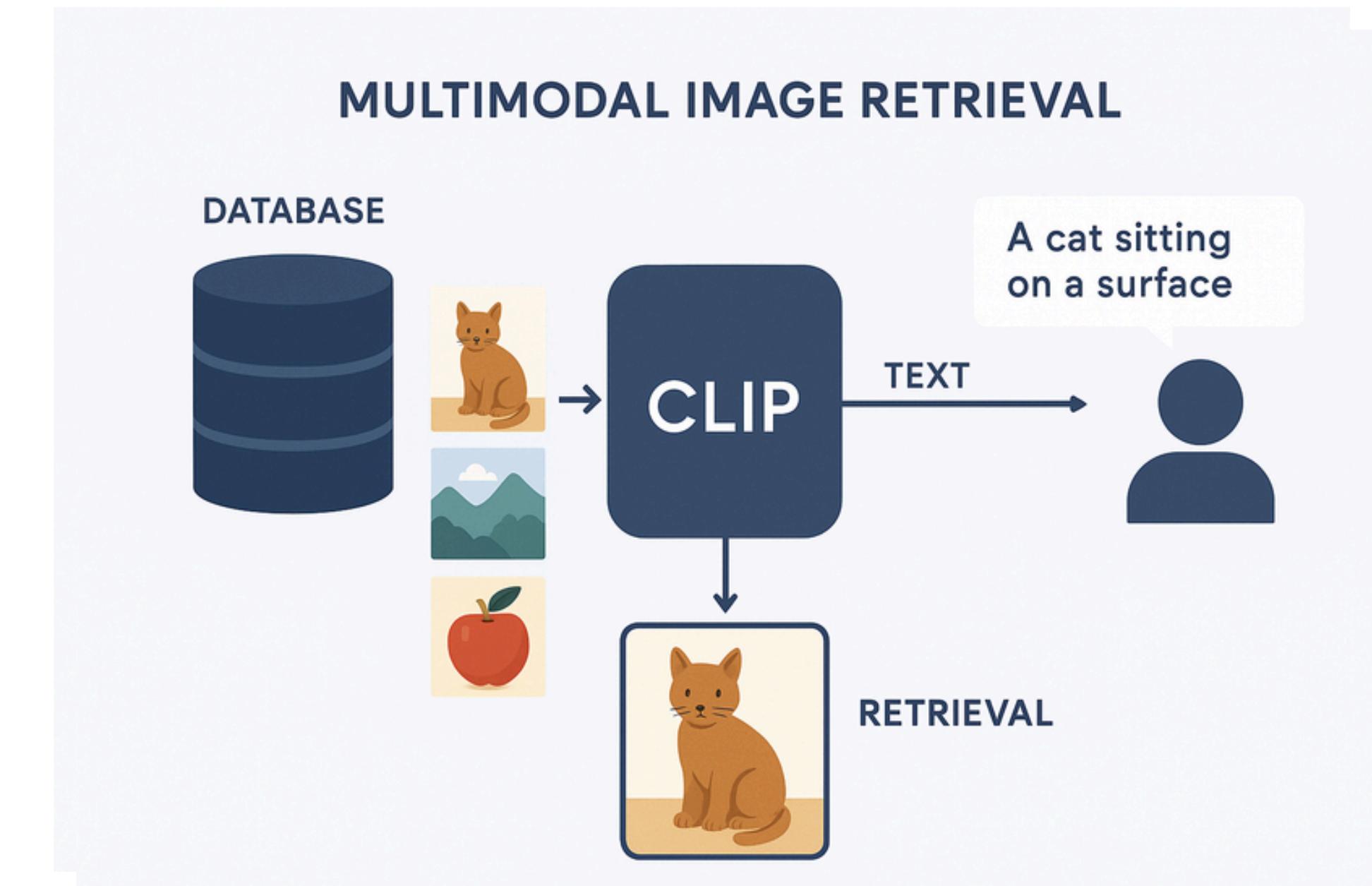
1.3 Tiền xử lý dữ liệu



1.1 Tóm tắt dự án



**Dự án tập trung xây dựng
hệ thống Truy vấn Hình Ảnh
Đa Phương Tiện (Multimodal
Image Retrieval) dựa trên
mô hình học sâu hiện đại
CLIP**



1.1 Tóm tắt dự án

Dữ liệu nghiên cứu:

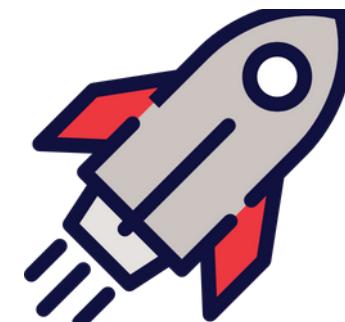
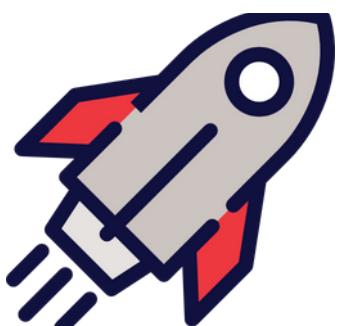
Tập dữ liệu Flickr8k nguồn Kaggle:

- **Quy mô:** Chứa 8.092 hình ảnh.
- **Mô tả:** Mỗi ảnh được gán 5 mô tả văn bản độc lập do con người viết.
- **Tổng số cặp (ảnh, mô tả):** Khoảng 40.460.
- **Mục đích:** Xây dựng các mô hình có thể hiểu và mô tả chính xác nội dung trong ảnh.

The Kaggle logo is displayed in a stylized blue font, consisting of the word "kaggle" in lowercase letters.

Mục tiêu chính của dự án:

- **Tinh chỉnh (fine-tune) mô hình CLIP trên tập Flickr8k nhằm cải thiện khả năng liên kết ảnh-văn bản.**
- **Triển khai hệ thống tìm kiếm dựa trên độ tương đồng cosine cho phép truy vấn hình ảnh bằng ảnh và văn bản.**



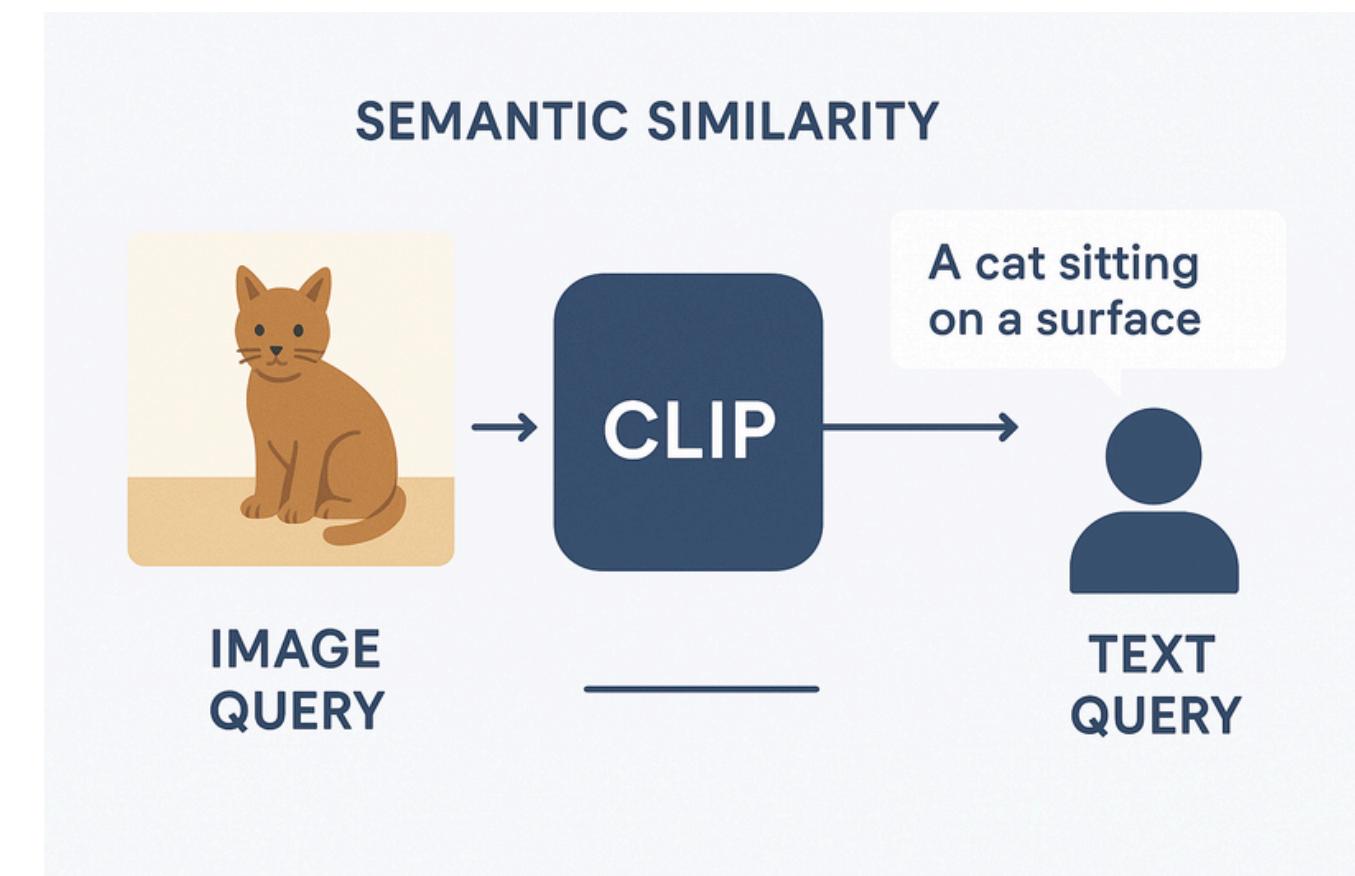
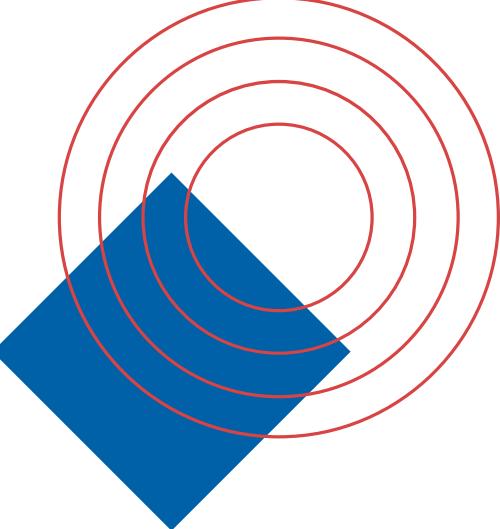
1.2 Bài toán đặt ra

Vấn đề: Khoảng cách Ngữ nghĩa và Bùng nổ Dữ liệu Hình ảnh.

Giải pháp đề xuất

Dự án sử dụng mô hình CLIP đã fine-tune để học các biểu diễn ngữ nghĩa giàu thông tin.

- Đo độ tương đồng ngữ nghĩa giữa ảnh-ảnh hoặc text-ảnh hoặc ảnh, text-ảnh
- Truy vấn ảnh dựa trên mô tả văn bản
- Tìm kiếm theo ý định thay vì theo từ khóa cứng



1.3 Tiền xử lý dữ liệu

1. Mục Tiêu Chính

Đảm bảo ảnh đầu vào có kích thước vuông
cố định (224 X 224 hoặc 256 X 256) và đã
được chuẩn hóa để tương thích với Kiến trúc CLIP.

2. Các Bước Tiền Xử Lý Ảnh cho CLIP

- Chuyển đổi Định dạng: Mở ảnh và chuyển sang hệ màu RGB (3 kênh).
- Tạo Ảnh Nền Vuông : Xác định cạnh vuông lớn nhất: Cạnh = $\max(h, w)$
- Căn Giữa: Tính toán Offset O_x, O_y và chèn ảnh gốc vào trung tâm ảnh nền vuông.
- Thay đổi Kích thước : Thay đổi kích thước ảnh vuông về kích thước cố định của mô hình (ví dụ: 224 X 224).
- Chuẩn hóa: Chuẩn hóa giá trị pixel (0 -> 1) và áp dụng mean và std tiêu chuẩn của mô hình CLIP





CHƯƠNG 2. PHƯƠNG PHÁP

01

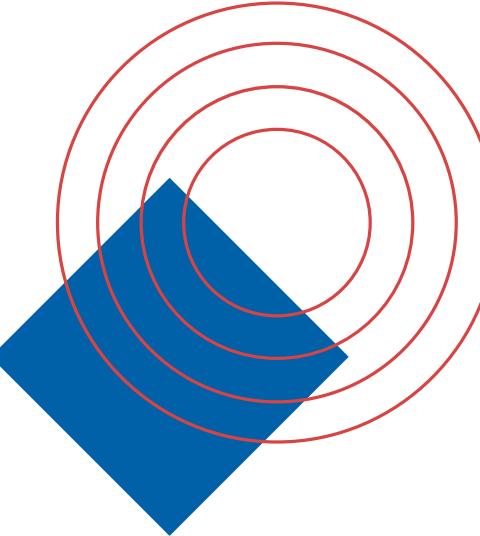
Biểu diễn text, image thành vector
(embedding) bằng CLIP

02

So sánh độ giống nhau giữa vector query
và vector ảnh trong database

03

Chọn ra Top-K ảnh có độ tương đồng
cao nhất





1

Biểu diễn text (ảnh) thành vector (embedding) bằng CLIP

1.1

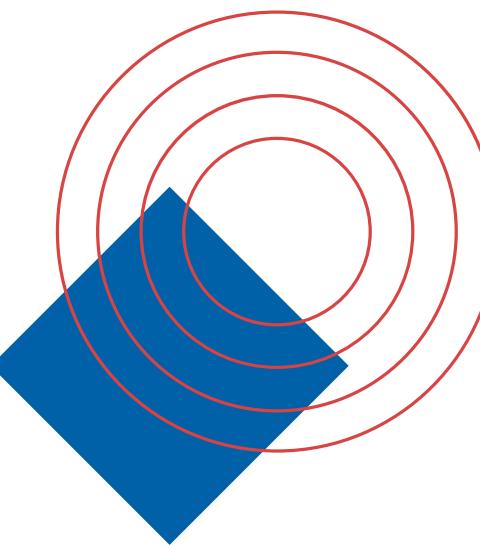
CLIP là gì?

1.2

A. Bộ mã hóa text của CLIP

1.3

B. Quá trình biến ảnh thành embedding của CLIP



1.1

CLIP là gì?



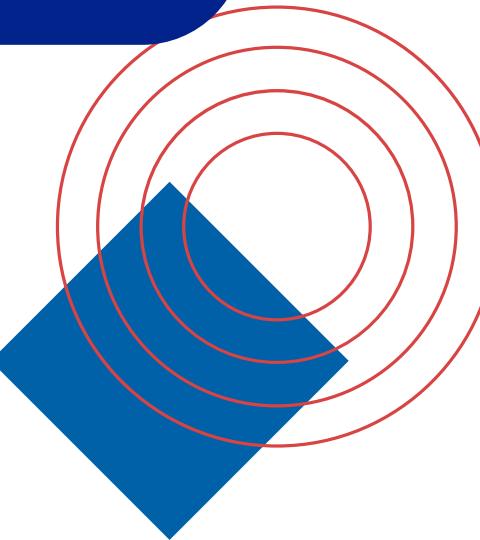
Huấn luyện bằng Contrastive Learning để đưa ảnh và văn bản có cùng ý nghĩa vào gần nhau trong không gian vector 512 chiều.

Nghĩa là:

• **Ảnh “một con mèo”**

• **Văn bản “a cat”**

→ phải có embedding gần nhau.



1.1

CLIP là gì?



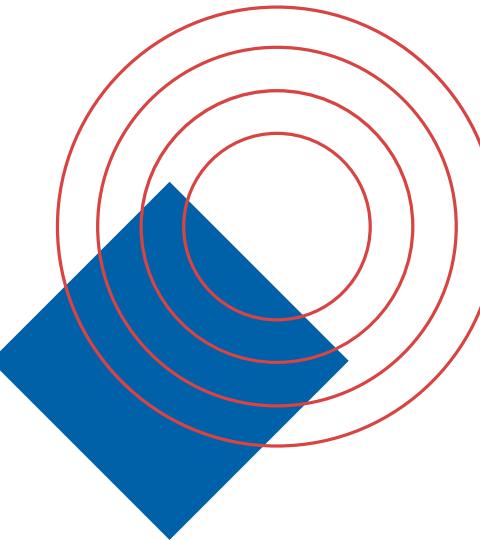
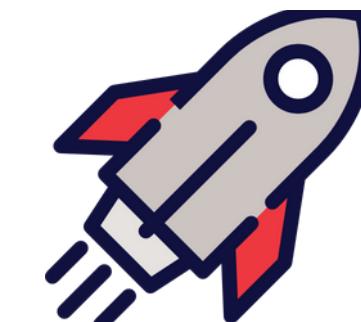
Công thức Contrastive Loss:

$$\mathcal{L} = -\log \frac{\exp(\text{sim}(v_i, t_i)/\tau)}{\sum_j \exp(\text{sim}(v_i, t_j)/\tau)}$$

Trong đó:

- v_i = vector ảnh
- t_i = vector văn bản
- $\text{sim}(v, t)$ = cosine similarity
- τ = hệ số nhiệt (temperature)

=> CLIP học được điều này từ 400
triệu cặp ảnh-văn bản



1.2

A. Bộ mã hóa text của CLIP



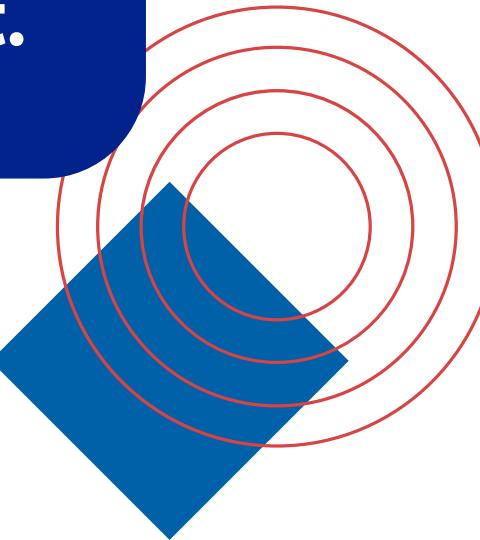
Text Transformer

- Tokenize câu thành 77 token
- Mỗi token → vector 512 chiều
- Thêm BOS/EOS
- Cộng positional encoding
- Transformer xử lý chuỗi token bằng Self-Attention



Chuỗi văn bản mặc định dài 77 token

- Câu luôn được mở rộng/pad thành 77 token
- Văn bản có 77 token để transformer xử lý thống nhất.



1.2

B. Bộ mã hóa ảnh của CLIP



Vision Transformer ViT-B/32

- Chia ảnh thành patch 32×32
- Mỗi patch được biến thành một vector
- Transformer xử lý chuỗi các patch

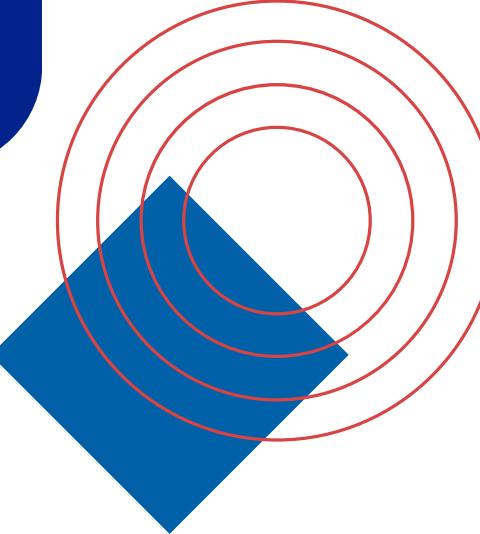


Kích thước ảnh mặc định là 224×224 .

Ảnh được chia thành:

$$(224/32)^2 = 7*7 = 49 \text{ patch}$$

Ảnh giống như 49 từ trong câu → transformer xử lý chúng bằng Self-Attention.



1.3

A. Quá trình biến text thành embedding của CLIP



Bước 1: Chuẩn hóa text

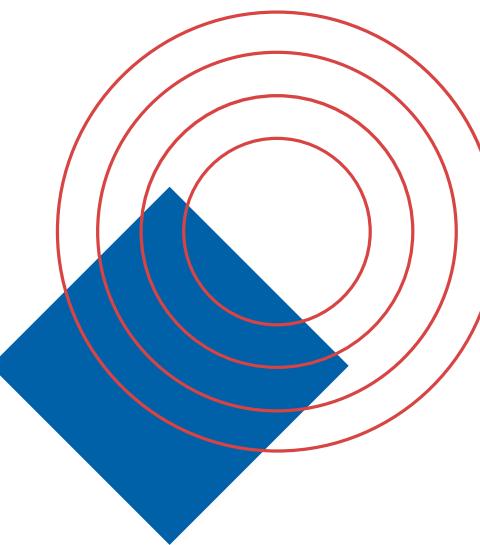
Bước 2: Tokenize câu → chuyển thành chuỗi token

Bước 3: Biểu diễn mỗi token thành vector embedding

Bước 4: Thêm token đặc biệt “SOS / EOS”

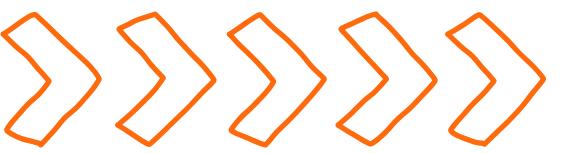
Bước 5: Cộng positional encoding

Bước 6: Transformer Encoder



1.3

B. Quá trình biến ảnh thành embedding của CLIP



Bước 1: Chuẩn hóa ảnh

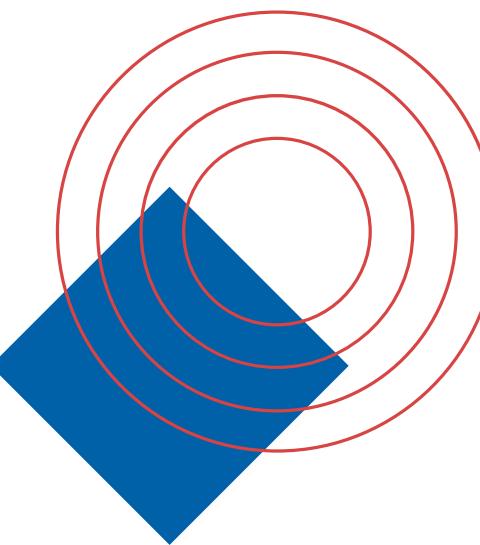
Bước 2: Chia ảnh thành patch

Bước 3: Biến mỗi patch(token) thành vector

Bước 4: Thêm Token đặc biệt “CLS”

Bước 5: Cộng positional encoding

Bước 6: Transformer Encoder





2

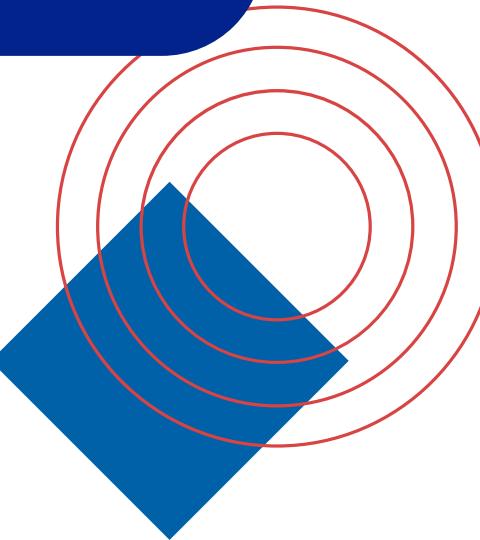
So sánh độ giống nhau giữa vector query và vector ảnh trong database

vector đặc trưng (F) được tính toán được áp dụng chuẩn hóa L2

$$F_{norm} = \frac{F}{\|F\|_2}$$

Vector đặc trưng đều có độ dài bằng 1:

- loại bỏ ảnh hưởng của độ lớn vector
- tập trung vào hướng của vector trong không gian





2

So sánh độ giống nhau giữa vector query và vector ảnh trong database

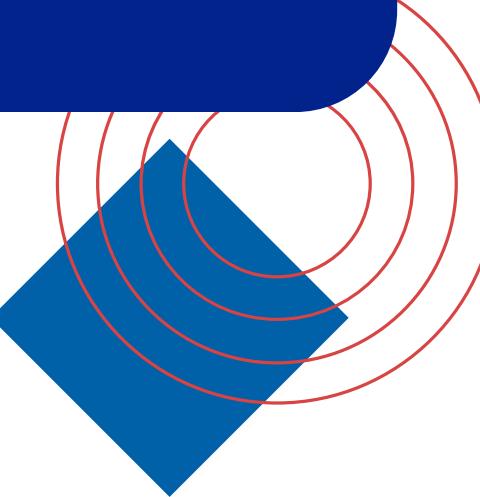
Tất cả ảnh đều được ánh xạ vào cùng một không gian vector (embedding space)

Cosine Similarity:

$$\text{cosine}(u, v) = \frac{u \cdot v}{\|u\| \|v\|}$$

Cosine similarity $\in [-1, 1]$:

- 1.0 \rightarrow 2 vector gần như trùng nhau \rightarrow ảnh rất giống
- 0.7 - 0.9 \rightarrow giống mạnh (high similarity)
- 0.3 - 0.6 \rightarrow giống vừa
- 0.0 \rightarrow không liên quan
- < 0 \rightarrow đối ngược trong không gian semantic



3

Chọn ra Top-K ảnh có độ tương đồng cao nhất

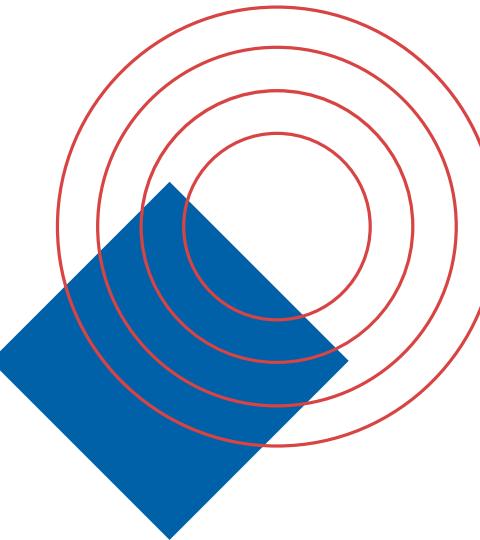


Top-K retrieval trong không gian embedding

- Mỗi ảnh = 1 điểm trong không gian 512 chiều.
 - Ảnh query cũng là 1 điểm trong không gian đó.
 - Cosine similarity đo mức độ gần nhau giữa các điểm.
- Ảnh nào nằm gần query nhất trong không gian vector → ảnh giống nhất.

Thuật toán thực hiện:

1. Tính khoảng cách từ query đến tất cả các điểm (vector database).
2. Sắp xếp các điểm theo độ gần.
3. Chọn K điểm gần nhất.



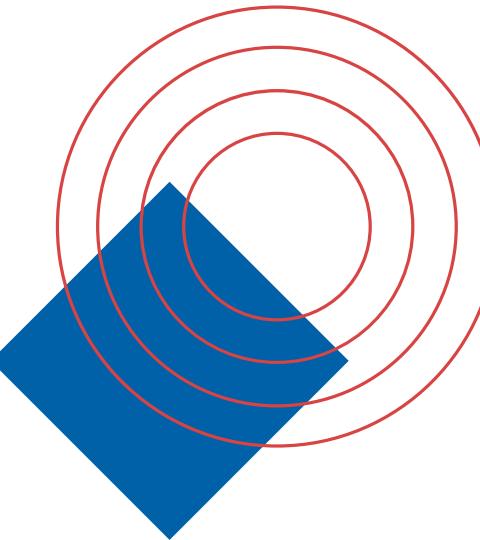
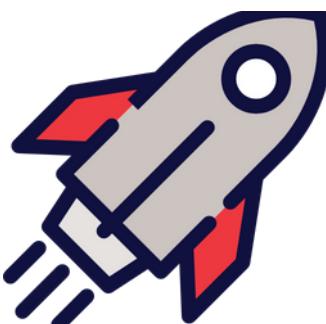
CHƯƠNG 3: TRIỂN KHAI

1 FINE-TUNE CLIP

1. Giải thích Hàm load_flickr8k_captions

Mục đích: Đọc tệp chú thích và chuyển thành Python Dictionary dễ sử dụng.

```
# captions_dict = {
#     "image_file_1.jpg": ["caption 1a", "caption 1b", "caption 1c", "caption 1d", "caption 1e"],
#     "image_file_2.jpg": ["caption 2a", "caption 2b", "caption 2c", "caption 2d", "caption 2e"],
#     # ...
# }
```



CHƯƠNG 3: TRIỂN KHAI

1 FINE-TUNE CLIP

2. Lớp Flickr8kDataset

2.1 Tổ chức Dữ liệu cho Quá trình Tinh chỉnh Mô hình CLIP

Thách thức Flickr8k
Mỗi hình ảnh (Image) trong
tập dữ liệu Flickr8k có **5
chú thích (Caption)** khác
nhau.

Vấn đề: Mô hình cần các
cặp 1-1 (Ảnh, Caption) duy
nhất để học khớp chính
xác.

Giải pháp: Cơ chế Mở rộng Dữ liệu
Hàm `__init__` chuyển đổi từ cấu trúc (Ảnh, [C1, C2, C3, C4, C5])
sang danh sách phẳng [(Ảnh, C1), (Ảnh, C2), ...].
→ Tổng số mẫu = 5 lần số lượng ảnh gốc.

CHƯƠNG 3: TRIỂN KHAI

1 FINE-TUNE CLIP

2. Lớp Flickr8kDataset

Tải Ảnh & Xử lý Lỗi

Sử dụng **Try...Except** để xử lý lỗi tệp bị hỏng/thiếu; tự động gọi mẫu tiếp theo.

CLIPProcessor

Dùng self.processor để xử lý **đồng thời** cả Ảnh (Resize, Normalize) và Văn bản (Tokenization, Padding, Max Length=77).

Đầu ra Cuối cùng

Sử dụng **.squeeze(0)** cho từng Tensor đầu ra để loại bỏ chiều Batch size = 1.
Điều này cho phép DataLoader tự động gom các mẫu thành Batch lớn hơn.

CHƯƠNG 3: TRIỂN KHAI

1 FINE-TUNE CLIP

3. Khởi tạo mô hình huấn luyện

1. Tải CLIPProcessor

Vai trò: Cung cấp **Tokenizer** (cho văn bản) và **Feature Extractor** (cho ảnh), đảm bảo tiền xử lý đồng nhất với mô hình CLIP gốc.



2. Khởi tạo Dataset và DataLoader

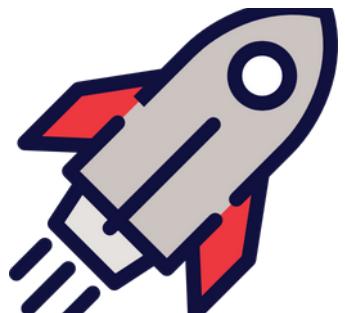
1. Dataset (Flickr8kDataset)

Tạo instance của lớp Dataset tùy chỉnh, truyền Processor và từ điển chú thích.

2. DataLoader

Batch Size = 16: Tải dữ liệu theo lô 16 cặp.

Shuffle = True: Trộn ngẫu nhiên dữ liệu mỗi Epoch để tránh bias và cải thiện hội tụ.



CHƯƠNG 3: TRIỂN KHAI

1 FINE-TUNE CLIP

4. Vòng lặp tinh chỉnh



1. Khởi tạo Tối ưu hóa

Sử dụng AdamW với **LR nhỏ ($1e-6$)** để tinh chỉnh nhẹ các trọng số đã được huấn luyện sẵn của mô hình.



2. Vòng lặp Dữ liệu

Duyệt qua 3 Epochs. Dữ liệu được tải theo lô (batch) và chuyển lên GPU trước khi đưa vào mô hình.



3. Kiểm tra Hiệu suất

Quan sát thấy **Lỗi trung bình giảm dần** qua các Epoch ($0.0738 \rightarrow 0.0438 \rightarrow 0.0357$), cho thấy mô hình học thành công.

CHƯƠNG 3: TRIỂN KHAI

2. PHƯƠNG PHÁP TRIỂN KHAI

2.1 Chuẩn hóa ảnh trong CLIP

Bộ tham số chuẩn hóa của CLIP

Bộ thống kê do OpenAI cung cấp:
Mean (R, G, B): 0.481 - 0.458 - 0.408
Std (R, G, B): 0.269 - 0.261 - 0.276

Quy trình chuẩn hóa ảnh

Bước 1 - Chuyển ảnh sang Tensor
Chuyển ảnh RGB từ 0-255 → 0-1.
Đây là dạng mà mạng nơ-ron xử lý tốt.

Bước 2 - Chuẩn hóa từng kênh
Ảnh được đưa về trung bình 0, độ lệch chuẩn 1.
Giúp giảm chênh lệch ánh sáng giữa các ảnh.



CHƯƠNG 3: TRIỂN KHAI

2. PHƯƠNG PHÁP TRIỂN KHAI

2.2 TRÍCH XUẤT VECTO

Mục tiêu: Các vector nằm cùng một không gian → có thể đo độ tương đồng cosine.

Quy trình trích xuất vector ảnh

1. Đầu vào ảnh

- Ảnh được đọc và xử lý với transform CLIP FT.
- Chuyển sang tensor chuẩn hóa.

2. Đưa vào Image Encoder

- Mô hình tạo vector đặc trưng 512 chiều.
- Thể hiện nội dung ảnh: vật thể, hành động, ngữ cảnh...

3. Chuẩn hóa L2

- Vector được chuẩn hóa để so sánh cosine chính xác.
- Giúp embedding ổn định, giảm sai lệch giữa các ảnh.

Quy trình trích xuất vector văn bản

1. Tokenize văn bản bằng Processor

- Tách từ, chuyển thành token ID
- Padding & tạo attention mask

2. Đưa vào Text Encoder của CLIPFT

- Tạo vector văn bản 512 chiều
- Mã hóa ý nghĩa câu mô tả

3. Chuẩn hóa L2

- Bắt buộc để khớp với vector ảnh
- Hỗ trợ tính cosine similarity chính xác



CHƯƠNG 3: TRIỂN KHAI

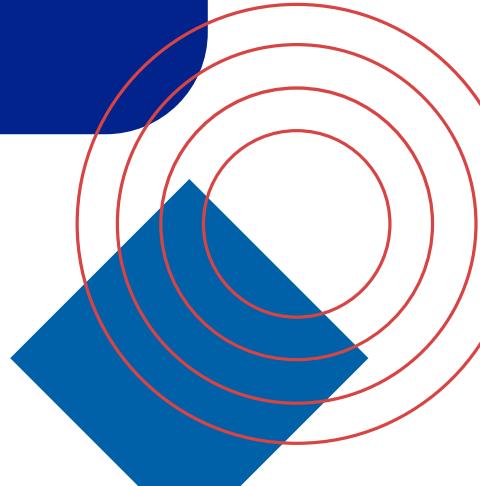
2. PHƯƠNG PHÁP TRIỂN KHAI

2.3 Hàm so sánh độ tương đồng



```
def cosine_similarity(a, b):  
    a = np.array(a).squeeze()  
    b = np.array(b).squeeze()  
    return np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))
```

- Giá trị gần 1 => Vector có hướng gần như nhau (rất giống nhau về mặt ngữ nghĩa).
- Giá trị gần 0 => Vector gần như độc lập (không liên quan).



CHƯƠNG 3: TRIỂN KHAI



2. PHƯƠNG PHÁP TRIỂN KHAI

2.4 Quy trình tìm kiếm ảnh

2.4.1 Tiềng xử lý Ảnh Truy vấn

- **Đổi ảnh sang định dạng màu**
- **Tạo một ảnh vuông kích thước bằng cạnh lớn nhất của ảnh gốc**
- **Tính toán các offset (y_offset, x_offset) để chèn ảnh gốc**
- **Chèn ảnh**
- **Resize ảnh vuông**

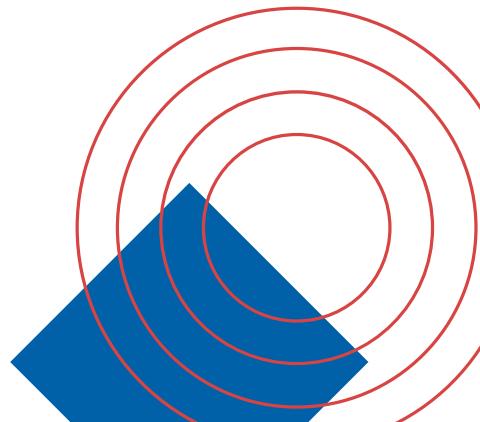


```
# Mở và chuyển đổi ảnh sang định dạng RGB
image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
h, w = image.shape[:2]

# tạo ảnh nền vuông có cạnh bằng max(h, w)
size = max(h, w)
square = np.zeros((size, size, 3), dtype=np.uint8)

# tính offset để căn giữa
y_offset = (size - h) // 2
x_offset = (size - w) // 2

# chèn ảnh gốc vào
square[y_offset:y_offset+h, x_offset:x_offset+w] = image
# resize
inputs = cv2.resize(square, (224, 224))
```



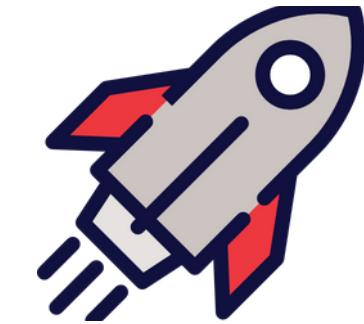


CHƯƠNG 3: TRIỂN KHAI

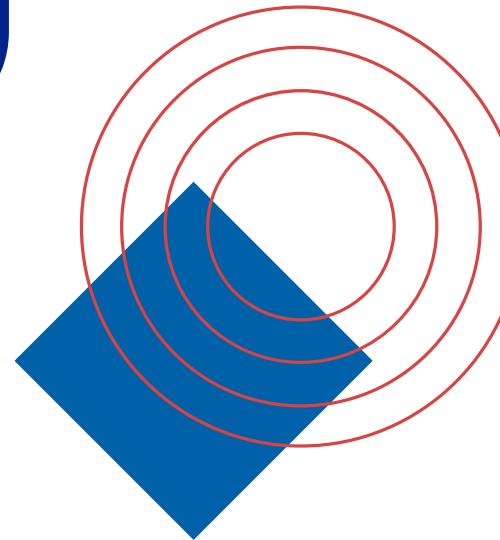
2. PHƯƠNG PHÁP TRIỂN KHAI

2.4 Quy trình tìm kiếm ảnh

2.4.2 Trích xuất Vector Đặc trưng



- **Vector Văn bản (text_vec):** Sử dụng hàm `extract_text_features`
- **Vector Ảnh(image_vec):** Ảnh đã được tiền xử lý (Bước1) được đưa vào hàm `extract_image_features`



CHƯƠNG 3: TRIỂN KHAI



2. PHƯƠNG PHÁP TRIỂN KHAI

2.4 Quy trình tìm kiếm ảnh

2.4.3 Kết hợp Vector Truy vấn

- Đầu vào: `text_vec`, `image_vec`, và trọng số `alpha = 0.6`
- Công thức kết hợp:

`combined = alpha * text_vec + (1 - alpha) * image_vec`

_ `alpha` là trọng số của vector văn bản.

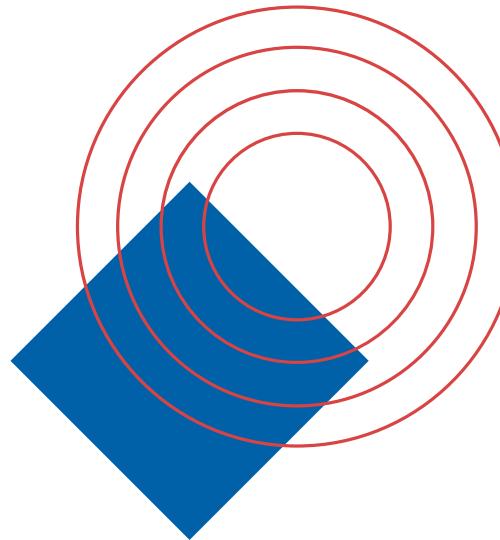
_ `1-alpha` là trọng số của vector ảnh

```
def combine_query(text_vec=None, image_vec=None, alpha=0.6):
    """
    alpha = trọng số text
    (1-alpha) = trọng số ảnh

    Nếu chỉ có text hoặc chỉ có ảnh → tự xử lý.
    """

    if text_vec is not None and image_vec is not None:
        combined = alpha * text_vec + (1 - alpha) * image_vec
    elif text_vec is not None:
        combined = text_vec
    else:
        combined = image_vec

    combined = combined / np.linalg.norm(combined)
    return combined
```



CHƯƠNG 3: TRIỂN KHAI

2. PHƯƠNG PHÁP TRIỂN KHAI

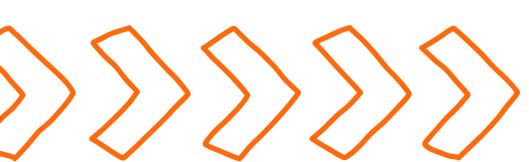
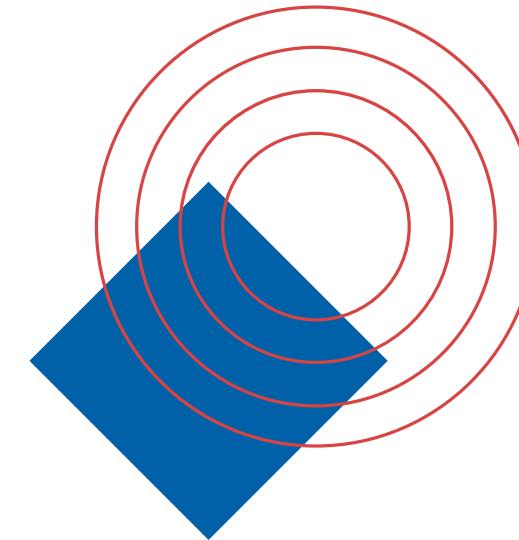
2.4 Quy trình tìm kiếm ảnh

2.4.4 Tính toán Độ tương đồng và Xếp hạng

- **Lặp qua cơ sở dữ liệu:** Với mỗi ảnh trong cơ sở dữ liệu:
 1. Lấy vector đặc trưng ảnh (img_vec).
 2. Tính toán độ tương đồng bằng Cosine Similarity:
`score = cosine_similarity(img_vec, combined_vec)`
 3. Lưu trữ cặp (img_path, score) vào danh sách scores.
- **Sắp xếp:** Danh sách scores

```
text_query = "đây là text_query"
text_vec = extract_text_features("A boy and dog")
path_img = "đây là đường dẫn ảnh query"
image_vec = extract_image_features(path_img)
combined_vec = combine_query(text_vec, image_vec)
scores = []
for img_path, img_vec in loaded.items():
    img_vec = img_vec.reshape(1, -1)
    score = float(cosine_similarity([img_vec], [combined_vec]))
    scores.append((img_path, score))

scores.sort(key=lambda x: x[1], reverse=True)
top_k = 5
for img, s in scores[:top_k]:
    print(img, s)
```



CHƯƠNG 4: KẾT QUẢ

1 Kết quả Truy vấn Ảnh theo Ảnh

Ảnh Query:



Ví dụ kết quả:

0.8996



0.7985



0.7610

CHƯƠNG 4: KẾT QUẢ

2. Kết quả truy vấn ảnh theo văn bản

Ví dụ kết quả:



0.3547

Truy vấn được sử dụng:

“A boy jumping in a fountain”

(Một cậu bé đang nhảy trong đài phun nước)



0.3493



0.3371

CHƯƠNG 4: KẾT QUẢ

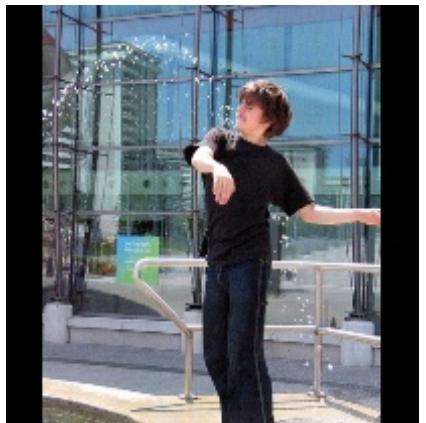
3. Kết quả Truy vấn Kết hợp Ảnh và Văn bản

Text query: “A boy and dog”

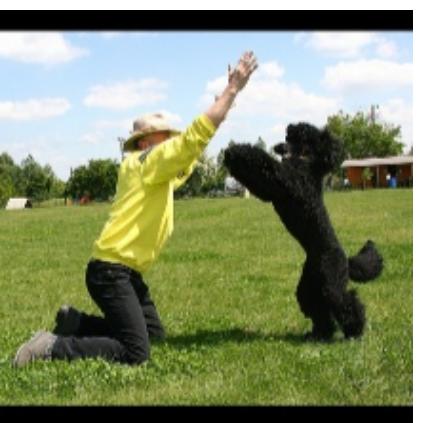
Ảnh truy vấn:



Kết quả:



0.7845



0.6281



0.612



0.6121

CHƯƠNG 4: KẾT QUẢ

Điểm mạnh của mô hình

1. Nhận dạng tốt các đối tượng chính trong ảnh
2. Hiểu tốt ngữ cảnh và hành động
3. Tốc độ tìm kiếm rất nhanh

Hạn chế của mô hình

1. Khó xử lý ảnh chứa nhiều đối tượng phức tạp
2. Ảnh bị nhiễu (ánh sáng, góc chụp)
→ embedding không ổn định
3. Text query dài hoặc mô tả chi tiết quá → hiệu quả giảm

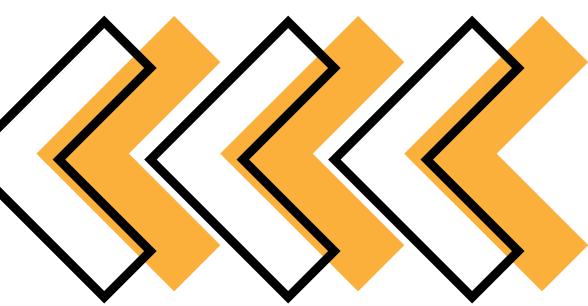
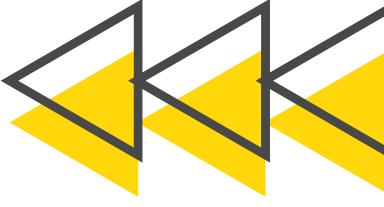
CHƯƠNG 5: KẾT LUẬN & HƯỚNG PHÁT TRIỂN

Kết luận

- Xây dựng bộ pipeline truy vấn hoàn chỉnh
- Huấn luyện mô hình CLIP cho bài toán tìm kiếm
- Hỗ trợ các chế độ truy vấn
- Tích hợp FAISS để mở rộng hệ thống

Hướng phát triển trong tương lai

- Sử dụng mô hình CLIP lớn hơn
- Mở rộng tập dữ liệu Flickr8k
- Fine-tune sâu hơn với chiến lược cải tiến
- Tích hợp FAISS GPU FAISS CPU
- Xây dựng giao diện Web/Ứng dụng
- Hỗ trợ truy vấn nâng cao



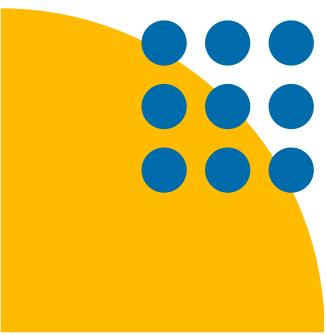
CHƯƠNG 6: HT WEB + AI

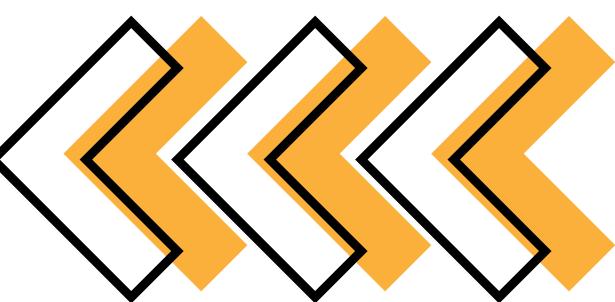
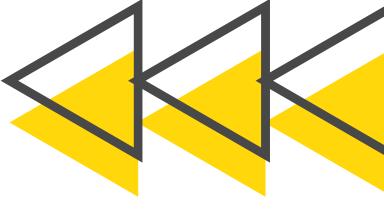
01

FRONTEND

02

BACKEND





01

FRONTEND

1.1

Tổng quan

1.2

Công nghệ sử dụng

1.3

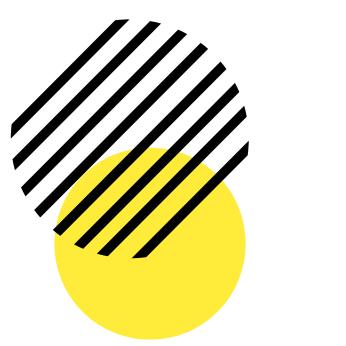
Kiến trúc

1.4

Chức năng & giao diện của hệ thống

1.5

Kết luận & hướng phát triển





1.1

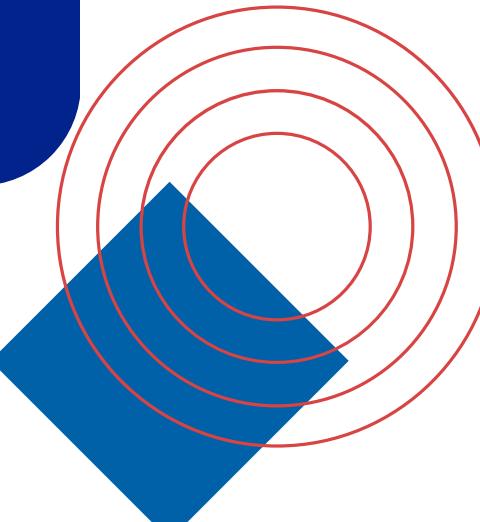
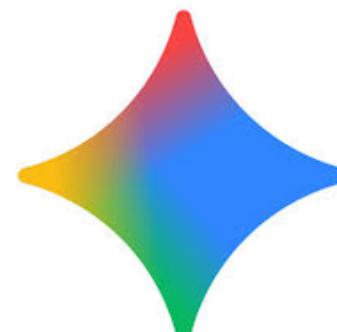
TỔNG QUAN



Frontend của dự án đóng vai trò là cầu nối tương tác giữa người dùng và mô hình AI . Thiết kế theo phong cách chat bot

Mục tiêu chính:

- Tạo trải nghiệm chat mượt mà, độ trễ thấp.
- Hỗ trợ đa phương thức (Multimodal): Xử lý input gồm text và file ảnh.
- Đảm bảo tính thẩm mỹ với chế độ Dark/Light mode





1.2

Công nghệ sử dụng

Kiến trúc chủ đạo: Vanilla JavaScript (JS thuần)

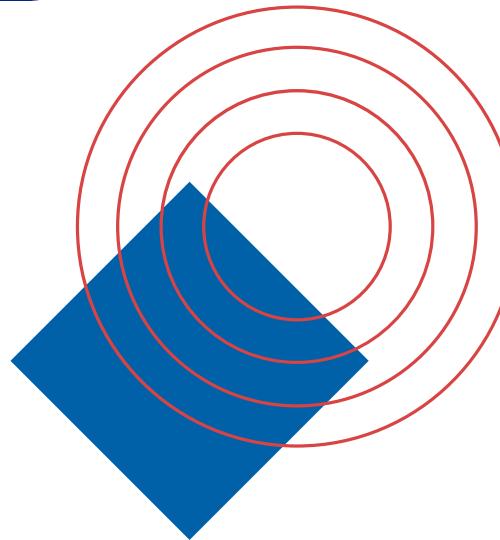
Ngôn ngữ cốt lõi

- HTML5: Cấu trúc Semantic Web, đảm bảo tính tiếp cận.
- CSS3: Responsive (Flexbox/Grid) & Theme động (CSS Variables).
- JavaScript (ES6+): Xử lý bất đồng bộ (Async/Await), quản lý trạng thái.



Thư viện hỗ trợ

- Marked.js: Render văn bản Markdown từ AI.
- Swiper: Hiển thị slider hình ảnh.
- Google Fonts: Font Inter tối ưu hiển thị.



 index.html

- **Vai trò:** Xương sống của ứng dụng.
- **Header:** Công cụ tiện ích (Theme Toggle, Clear History).
- **Main (#chatBox):** Vùng hiển thị hội thoại động.
- **Footer:** Nhập liệu đa năng (Textarea co giãn + Preview ảnh).
- **Thành phần ẩn:** Lightbox (Xem ảnh phóng to) tải sẵn.

 style.css

- **Vai trò:** Trải nghiệm thị giác (Visual Design).
- **Theming:** Dùng CSS Variables (:root) để đổi màu (Dark/Light) tức thì.
- **Responsive:** Media Queries tối ưu hiển thị trên Mobile.
- **Micro-interactions:** Hiệu ứng Hover, Transition và Animation tin nhắn mượt mà.

 script.js

- **Vai trò:** Điều phối luồng dữ liệu & Tương tác.
- **State Management:** Dùng localStorage lưu Lịch sử chat & Theme.
- **Xử lý ảnh:** Client-side với FileReader (Base64) -> Instant Preview (Xem trước ngay lập tức).
- **Async Communication:** Fetch API + FormData xử lý gửi/nhận dữ liệu bất đồng bộ.



1.4

Chức năng & giao diện của hệ thống

01

Tương tác & Tìm kiếm

- Multimodal Query: Hỗ trợ nhập liệu đa phương thức (Văn bản + Hình ảnh) giúp AI hiểu rõ ngữ cảnh.
- Smart Response: Hiển thị kết quả thông minh.
- Feedback System: Trạng thái "Đang nhập..." giúp người dùng nhận biết tiến trình xử lý.

02

Quản lý đầu vào

- Clipboard Support: Dán ảnh nhanh từ bộ nhớ tạm (Ctrl+V) không cần lưu file.
- InstantPreview : Xem trước ảnh thu nhỏ ngay lập tức.
- FileManagement: Thêm/Xóa ảnh trong danh sách chờ gửi dễ dàng.

03

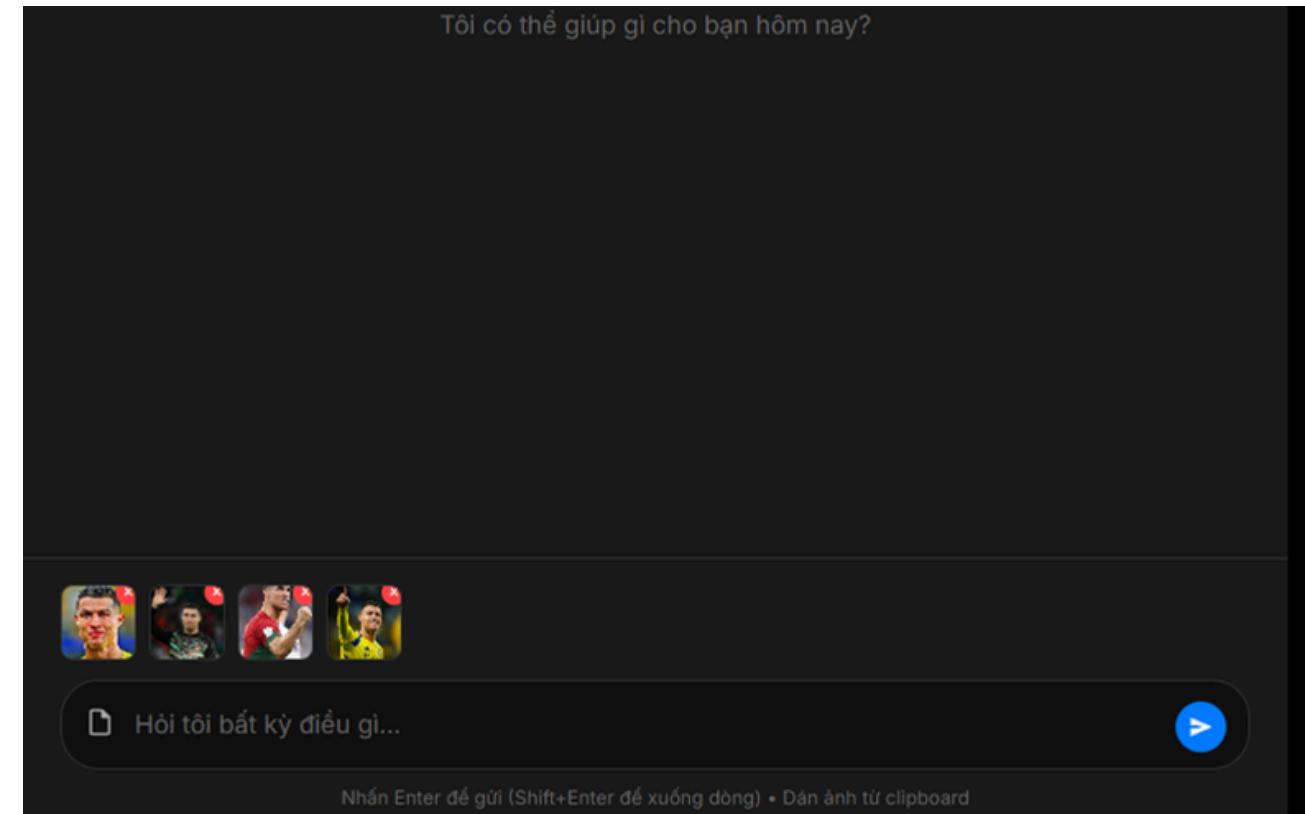
Tiện ích

- Theme: Chuyển đổi Dark/Light Mode linh hoạt.
- History: Tự động lưu lịch sử chat vào trình duyệt.
- Lightbox: Xem ảnh phóng to.
- Quick Copy: Sao chép câu trả lời chỉ với 1 cú click.

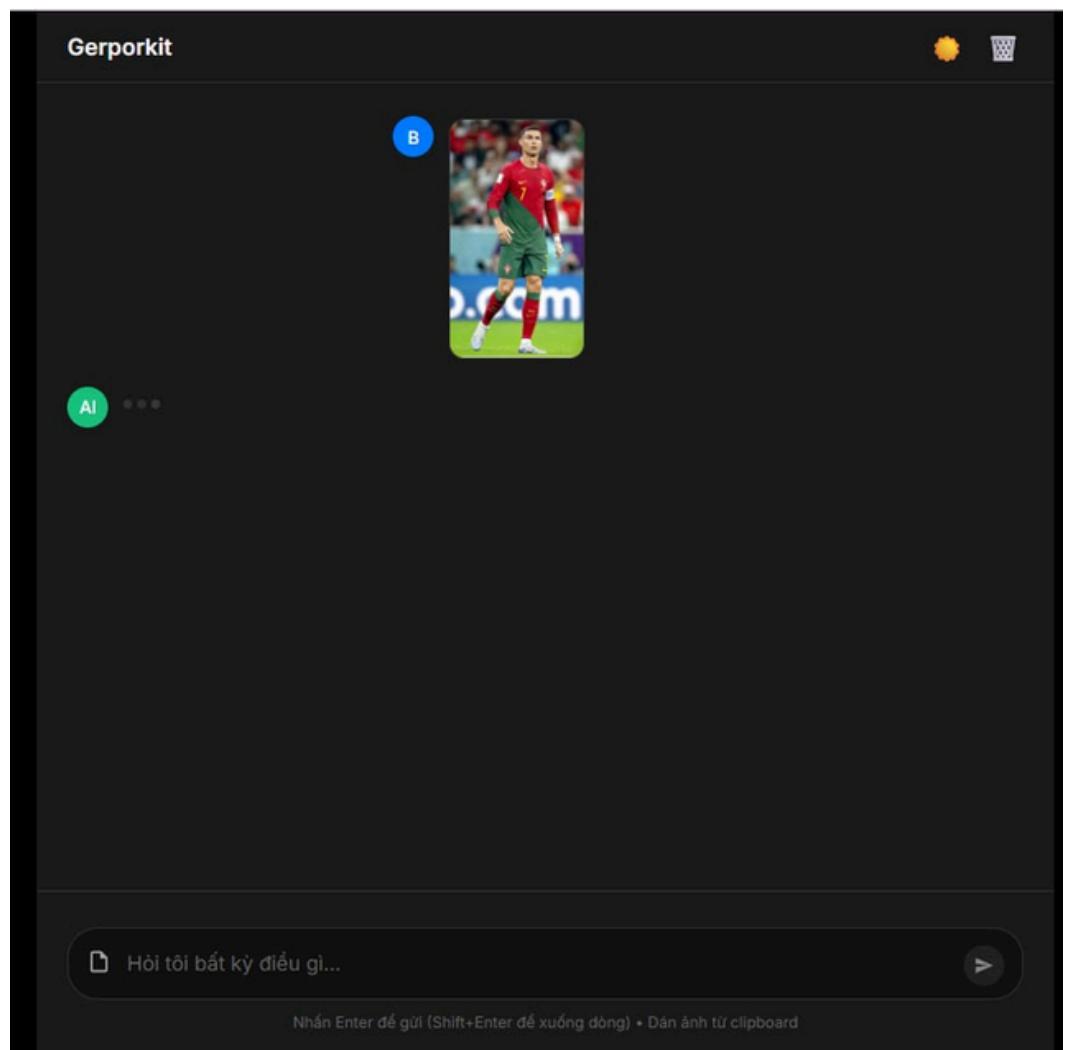
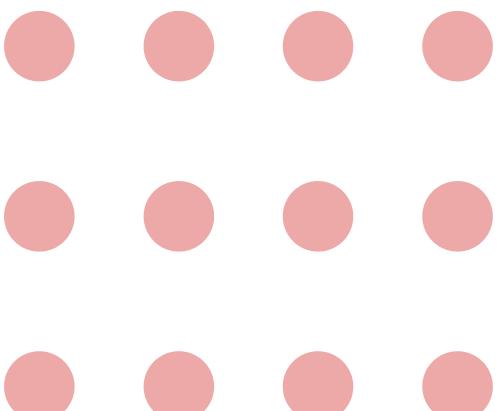


1.4

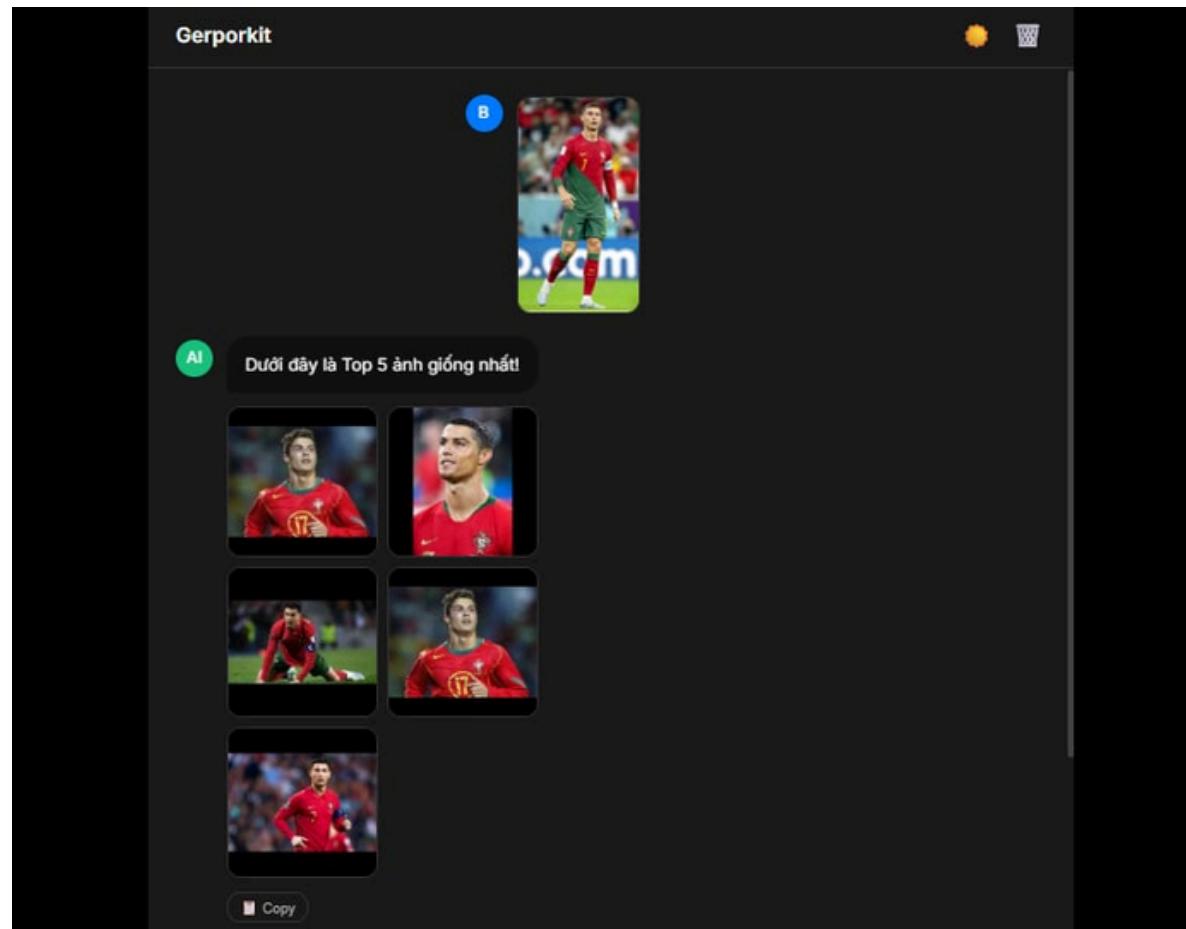
Chức năng & giao diện của hệ thống



xem trước ảnh



chatbot đang phản hồi

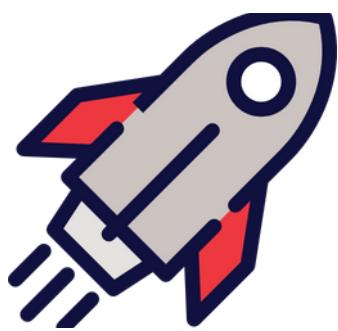


kết quả hình ảnh trả về



1.5

Kết luận & hướng phát triển

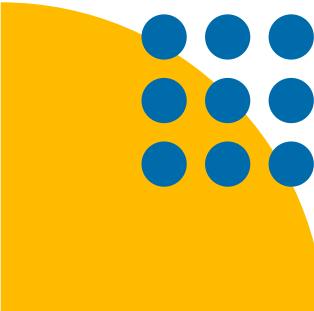


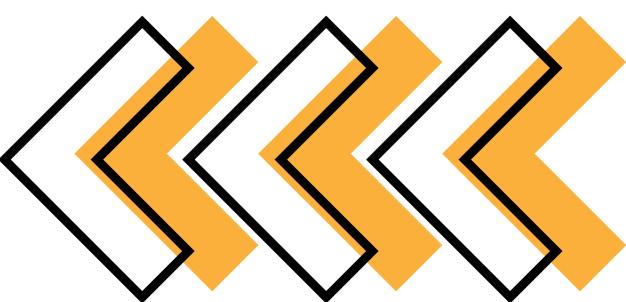
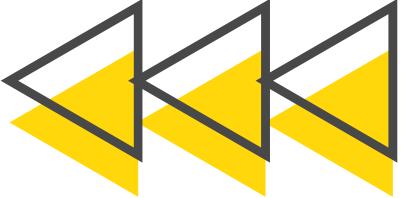
Kết quả đạt được

- Tính tương tác cao : trải nghiệm mượt
- Tối ưu hiệu năng
- Mã nguồn :dễ bảo trì và mở rộng.

Hướng phát triển

- Voice Input: Tích hợp nhập liệu bằng giọng nói (Speech-to-Text).
- Streaming Response: Hiển thị hiệu ứng gõ chữ từng từ (Typewriter effect) để giảm cảm giác chờ đợi.





02

BACKEND

2.1

Giới thiệu chung

2.2

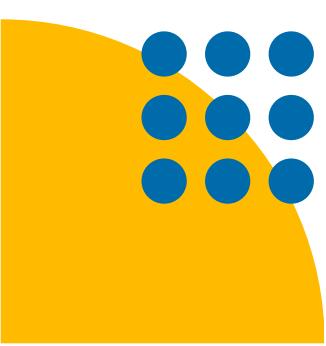
CÔNG NGHỆ

2.3

Cấu trúc chính

2.4

Kết luận & hướng phát triển





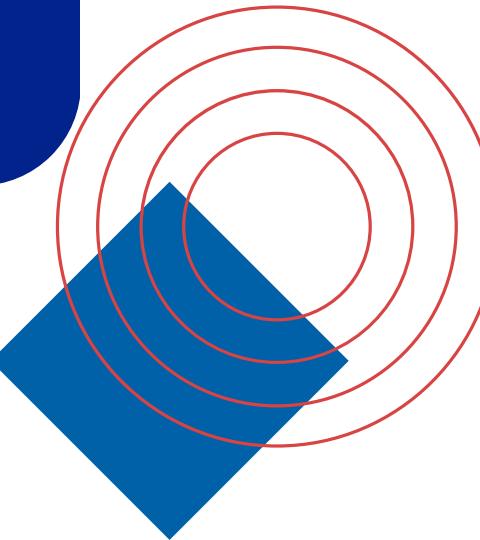
2.1

GIỚI THIỆU CHUNG

Backend của dự án đóng vai trò là trung tâm xử lý logic, nhận dữ liệu từ người dùng chuyển tiếp qua cho mô hình AI xử lý, rồi sau đó tiếp nhận kết quả trả về từ mô hình và đưa kết quả đó đến với người dùng.

Nhiệm vụ chính:

- Phục vụ Ứng dụng Web (Web Serving)
- Quản lý API & Điều phối (API Gateway)
- Xử lý Nghiệp vụ & Kết nối AI (Business Logic & Bridge)
- Chuẩn hóa dữ liệu đầu ra (Data Formatting)



2.2

Công nghệ



Ngôn ngữ chính: Python

Framework : FastAPI

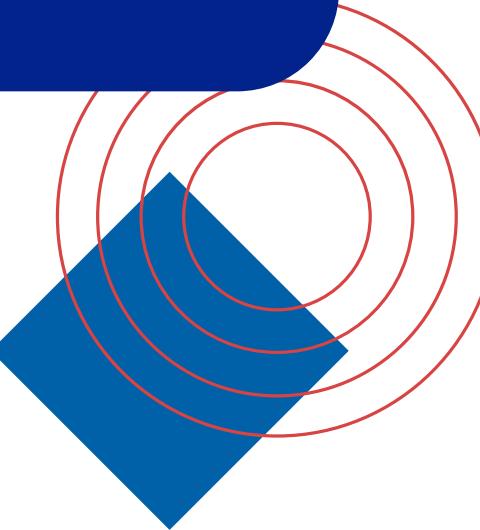
- Hiện đại, hiệu năng cao cho phát triển web
- Uvicorn: Một máy chủ ASGI siêu nhanh, xử lý các kết nối HTTP đồng bộ

Thư viện hỗ trợ

- Requests: Thư viện HTTP client phổ biến nhất của Python.
- Python-multipart: Thư viện hỗ trợ xử lý form data.
- Pillow (PIL): Thư viện xử lý ảnh.

Hệ tầng mạng & Kết nối

- Google Colab: Cung cấp phần cứng miễn phí để chạy các tác vụ AI nặng
- Ngrok: Mở một cổng public an toàn (HTTPS) để Local Backend có thể kết nối tới server AI đang chạy ở trong Google Colab.



2.3

Cấu trúc chính

main.py

- **Khởi động ứng dụng FastAPI.**
- **Cấu hình CORS:** Cho phép trình duyệt truy cập API mà không bị chặn bảo mật.
- **Phục vụ Static Files:** trực tiếp cung cấp các file giao diện cho người dùng truy cập.
- **Đăng ký Router:** Kết nối các file định tuyến vào ứng dụng chính.

image_route.py

- **Định nghĩa địa chỉ API (Endpoint).**
- **Validation (Kiểm tra):** Đóng vai trò "bảo vệ", đảm bảo người dùng gửi lên ít nhất là văn bản hoặc hình ảnh. Nếu gửi request rỗng, nó trả lỗi ngay lập tức.
- **Điều hướng:** Gọi sang image_service để xử lý logic nghiệp vụ.

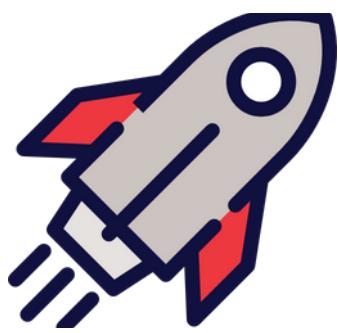
image_service.py

- **Xử lý File:** Đọc dữ liệu nhị phân từ file ảnh upload, xử lý con trỏ file để tránh lỗi đọc file rỗng.
- **Đóng gói (Payload):** Chuẩn bị dữ liệu theo định dạng multipart/form-data để gửi đi.
- **Giao tiếp:** Thực hiện cuộc gọi HTTP POST tới Server AI thông qua URL Ngrok.
- **Chuẩn hóa dữ liệu:** Nhận kết quả thô từ AI trả về cho Frontend.



2.4

Kết luận & hướng phát triển

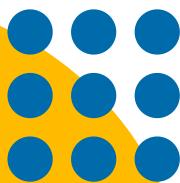


Kết quả đạt được

- Mô hình Hybrid Search mạnh mẽ
- Tận dụng tài nguyên thông minh
- Kiến trúc tinh gọn, mã nguồn dễ bảo trì và mở rộng.

Hướng phát triển

- Caching (Redis): Nếu người dùng tìm kiếm cùng một từ khóa "con mèo" nhiều lần, lưu kết quả vào Redis cache để trả về ngay lập tức mà không cần gọi sang AI Colab.
- Rate Limiting: giới hạn request



**THANK
YOU**