HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN THỰC TẬP CƠ SỞ

THỰC TẬP CƠ SỞ

ĐỀ TÀI: NHẬN DIỆN BỐI CẢNH QUA ẢNH

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**
Họ và tên sinh viên :**Nguyễn Quang Thiện**
Mã sinh viên :**B19DCCN668**
Lớp :**D19CNPM01**

Hà Nội – 2025

BÁO CÁO ĐỒ ÁN

1. Giới thiệu đề tài

Hệ thống nhận diện ngữ cảnh hình ảnh là một ứng dụng kết hợp giữa mô hình phát hiện vật thể (YOLOv8) và mô hình sinh mô tả ảnh (BLIP), hỗ trợ người dùng hiểu được nội dung và ngữ cảnh của các hình ảnh đầu vào một cách tự động.

1.1. Lý do chọn đề tài

Trong thời đại công nghệ 4.0, lượng thông tin hình ảnh ngày càng nhiều trong các lĩnh vực như nông nghiệp, y tế, giáo dục, giao thông, truyền thông,... Tuy nhiên, việc hiểu và phân tích nội dung ảnh vẫn là một thách thức nếu không có chuyên môn hoặc công cụ hỗ trợ. Vì vậy, việc xây dựng hệ thống tự động nhận diện vật thể và sinh mô tả ngữ cảnh ảnh mang lại giá trị lớn trong hỗ trợ con người tiếp cận thông tin nhanh chóng, đặc biệt trong:

- **Giáo dục**: giúp học sinh học qua hình ảnh trực quan.
- **Y tế**: hỗ trợ bác sĩ phân tích ảnh X-quang, MRI.
- **Người khiếm thị**: mô tả thế giới xung quanh qua hình ảnh.
- **Nông nghiệp thông minh**: nhận diện sâu bệnh, giám sát cây trồng.

2. Mục tiêu

- Phát hiện vật thể trong ảnh bằng YOLOv8
- Sinh mô tả ảnh bằng BLIP
- Tự động dịch mô tả sang tiếng Việt bằng Google Translate
- Giao diện Tkinter hỗ trợ người dùng chọn 1 hoặc nhiều ảnh
- Mở rộng: kết hợp mô tả từ nhiều ảnh để tạo ngữ cảnh tổng hợp

3. Cách hoạt động của YOLOv8

YOLOv8 (You Only Look Once phiên bản 8) là một mô hình phát hiện vật thể thời gian thực tiên tiến do Ultralytics phát triển. Nó kế thừa và cải tiến các đặc điểm của các phiên bản trước như YOLOv5, YOLOv7, với những cải tiến quan trọng như kiến trúc mới, tốc độ cao và khả năng phát hiện chính xác hơn.

Điểm nổi bật của YOLOv8

- Dễ sử dụng với cú pháp ngắn gọn qua thư viện ultralytics
- Hỗ trợ nhiều tác vụ: object detection, segmentation, classification, pose estimation
- Có các phiên bản từ nhỏ đến lớn: yolov8n, yolov8s, yolov8m, yolov8l, yolov8x

Cấu trúc mô hình YOLOv8

Mô hình YOLOv8 bao gồm ba phần chính:

- **Backbone**: Trích xuất đặc trưng từ ảnh đầu vào (sử dụng kiến trúc CSP và Conv)
- **Neck**: Kết hợp thông tin từ nhiều cấp độ đặc trưng (FPN, PAN)
- **Head**: Dự đoán bounding boxes, confidence và class

YOLOv8 sử dụng phương pháp anchor-free để tăng hiệu quả và giảm tính toán.

Dữ liệu đầu vào và đầu ra

- **Input**: Ảnh kích thước cố định (thường là 640x640)
- **Output**: Danh sách các bounding boxes, nhãn lớp và độ tin cậy (confidence)

4. Cấu trúc và hoạt động của BLIP

BLIP (Bootstrapping Language-Image Pretraining) là mô hình kết

hợp giữa xử lý ảnh và ngôn ngữ:

- **Input**: Một ảnh và (tùy chọn) một prompt mô tả
- **ViT (Vision Transformer)**: Mã hóa ảnh thành vector đặc trưng
- **Q-Former**: Học quan hệ giữa ảnh và ngôn ngữ
- **Decoder (GPT-like)**: Sinh chuỗi mô tả ảnh

BLIP cho phép thực hiện 2 nhiệm vụ:

- Image Captioning: sinh mô tả tự động
- VQA (Visual Question Answering): trả lời câu hỏi dựa trên ảnh

5. Chuẩn bị dữ liệu huấn luyện (COCO -> YOLO)

```
import os
import json
import shutil
from tqdm import tqdm
from pycocotools.coco import COCO

def convert_coco_json_to_yolo_txt(json_file_path, output_dir, image_dir):
   coco = COCO(json_file_path)
   os.makedirs(output_dir, exist_ok=True)

   image_id_to_filename = {img['id']: img['file_name'] for img in
coco.dataset['images']}
   categories = coco.loadCats(coco.getCatIds())
   category_id_to_index = {cat['id']: idx for idx, cat in
enumerate(categories)}

   for img_id in tqdm(coco.getImgIds(), desc="Converting"):
       ann_ids = coco.getAnnIds(imgIds=img_id)
       anns = coco.loadAnns(ann_ids)
       img_info = coco.loadImgs(img_id)[0]

       img_w = img_info['width']
       img_h = img_info['height']
       filename = os.path.splitext(img_info['file_name'])[0]
       label_file = os.path.join(output_dir, f"{filename}.txt")

       with open(label_file, "w") as f:
           for ann in anns:
               if ann.get("iscrowd", 0) == 1:
                   continue

               x, y, w, h = ann['bbox']
               x_center = (x + w / 2) / img_w
```

```
                y_center = (y + h / 2) / img_h
                w /= img_w
                h /= img_h

                category_id = ann['category_id']
                class_id = category_id_to_index[category_id]

                f.write(f"{class_id} {x_center:.6f} {y_center:.6f} {w:.6f}
{h:.6f}\n")

    return coco

def copy_images(image_ids, image_dir, save_dir, coco):
    os.makedirs(save_dir, exist_ok=True)
    for img_id in tqdm(image_ids, desc=f"Copying images to {save_dir}"):
        file_name = coco.loadImgs(img_id)[0]['file_name']
        src_path = os.path.join(image_dir, file_name)
        dst_path = os.path.join(save_dir, file_name)
        if os.path.exists(src_path):
            shutil.copy2(src_path, dst_path)

if __name__ == "__main__":
    # === Đường dẫn thư mục dữ liệu COCO ===
    root = "datasetscoco"
    annotations = os.path.join(root, "annotations")
    images = os.path.join(root, "images")

    # === Train set ===
    train_json = os.path.join(annotations, "instances_train2017.json")
    train_img_dir = os.path.join(images, "train2017")
    train_label_output = os.path.join("datasets", "labels", "train")
    train_image_output = os.path.join("datasets", "images", "train")

    train_coco = convert_coco_json_to_yolo_txt(train_json,
train_label_output, train_img_dir)
    copy_images(train_coco.getImgIds(), train_img_dir, train_image_output,
train_coco)

    # === Validation set ===
    val_json = os.path.join(annotations, "instances_val2017.json")
    val_img_dir = os.path.join(images, "val2017")
    val_label_output = os.path.join("datasets", "labels", "val")
    val_image_output = os.path.join("datasets", "images", "val")

    val_coco = convert_coco_json_to_yolo_txt(val_json, val_label_output,
val_img_dir)
    copy_images(val_coco.getImgIds(), val_img_dir, val_image_output,
val_coco)

    print("✅ Hoàn tất chuyển đổi COCO -> YOLO và sao chép ảnh!")
```

6. Tập tin YAML cấu hình dữ liệu huấn luyện YOLO

```
path: ./datasetsmaill  # thư mục chứa 'images' và 'labels'
train: images/train     # KHÔNG có dấu '/' ở đầu
val: images/val

nc: 80  # số lượng lớp

names:
 [
   'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
   'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign',
   'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep',
   'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella',
   'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard',
   'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard',
   'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork',
   'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange',
   'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair',
   'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv',
   'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave',
   'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase',
   'scissors', 'teddy bear', 'hair drier', 'toothbrush'
 ]
```

7. Huấn luyện mô hình YOLOv8

```
from ultralytics import YOLO

# 1. Load mô hình gốc YOLOv8n
model = YOLO('yolov8n.pt')  # Bạn có thể thay bằng yolov8s.pt nếu muốn mô hình lớn hơn

# 2. Huấn luyện mô hình bằng CPU
results = model.train(
   data='coco128.yaml',  # Đường dẫn file YAML chứa dữ liệu
   epochs=50,
   imgsz=640,
   batch=8,         # giảm batch size khi dùng CPU để tránh tràn RAM
   workers=0,       # dùng 1 luồng để tránh treo máy yếu
   device=0# ⚠️ CHẠY TRÊN CPU
)

# 3. Đánh giá mô hình sau huấn luyệnD:\pythonProject1\venv\Scripts\python.exe
```

```
D:\pythonProject1\train_model.py
# Ultralytics 8.3.146  Python-3.10.9 torch-2.7.0+cpu
# Traceback (most recent call last):
#   File "D:\pythonProject1\train_model.py", line 7, in <module>
#     results = model.train(
#   File
"D:\pythonProject1\venv\lib\site-packages\ultralytics\engine\model.py",
line 791, in train
#     self.trainer = (trainer or
self._smart_load("trainer"))(overrides=args, _callbacks=self.callbacks)
#   File
"D:\pythonProject1\venv\lib\site-packages\ultralytics\engine\trainer.py",
line 121, in __init__
#     self.device = select_device(self.args.device, self.args.batch)
#   File
"D:\pythonProject1\venv\lib\site-packages\ultralytics\utils\torch_utils.py"
, line 201, in select_device
#     raise ValueError(
# ValueError: Invalid CUDA 'device=0' requested. Use 'device=cpu' or pass
valid CUDA device(s) if available, i.e. 'device=0' or 'device=0,1,2,3' for
Multi-GPU.
#
# torch.cuda.is_available(): False
# torch.cuda.device_count(): 0
# os.environ['CUDA_VISIBLE_DEVICES']: None
# See https://pytorch.org/get-started/locally/ for up-to-date torch install
instructions if no CUDA devices are seen by torch.
metrics = model.val(device=0)

# 4. In hiệu năng
print("\n📊 Hiệu năng mô hình:")
print(f"mAP@0.5         : {metrics.box.map50:.4f}")
print(f"mAP@0.5:0.95    : {metrics.box.map:.4f}")
print(f"Precision       : {metrics.box.precision:.4f}")
print(f"Recall          : {metrics.box.recall:.4f}")

# 5. Tính và in sai số (error)
error_map50 = 1 - metrics.box.map50
error_precision = 1 - metrics.box.precision

print("\n❌ Sai số:")
print(f"Sai số mAP@0.5       : {error_map50:.4f}")
print(f"Sai số Precision     : {error_precision:.4f}")
```

8. Tạo giao diện Tkinter xử lý ảnh

Hệ thống giao diện được xây dựng bằng thư viện Tkinter, hỗ trợ chọn

1 hoặc nhiều ảnh, hiển thị kết quả mô tả từ BLIP sau khi phát hiện vật thể với YOLO.

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
from transformers import BlipProcessor, BlipForConditionalGeneration
from googletrans import Translator
from ultralytics import YOLO
import torch

# Tải mô hình BLIP
blip_processor =
BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
blip_model =
BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-caption
ing-base")

# Tải mô hình YOLOv8
yolo_model = YOLO("yolo11n.pt")

selected_files = []  # Danh sách các ảnh đã chọn

# Hàm mô tả ảnh bằng BLIP
def generate_caption(image_path):
    raw_image = Image.open(image_path).convert('RGB')
    inputs = blip_processor(raw_image, return_tensors="pt")
    out = blip_model.generate(**inputs)
    return blip_processor.decode(out[0], skip_special_tokens=True)

# Hàm dịch tiếng Anh sang tiếng Việt
def translate_to_vietnamese(text):
    translator = Translator()
    result = translator.translate(text, src='en', dest='vi')
    return result.text

# Hàm xử lý mô tả ảnh
def process_image(file_path):
    # Hiển thị ảnh
    img = Image.open(file_path)
    img_resized = img.resize((400, 300))
    img_tk = ImageTk.PhotoImage(img_resized)
    panel.configure(image=img_tk)
    panel.image = img_tk

    # Hiển thị trạng thái
    status_label.config(text="⏳ Đang xử lý...")
    root.update()

    try:
        # Nhận diện đối tượng bằng YOLO
```

```
        results = yolo_model(file_path)
        objects = results[0].names
        detected = list(set([objects[int(cls)] for cls in
results[0].boxes.cls]))
        detected_objects = ", ".join(detected) if detected else "Không phát
hiện đối tượng nào"

        # Mô tả bằng BLIP
        caption_en = generate_caption(file_path)

        # Dịch sang tiếng Việt
        caption_vi = translate_to_vietnamese(caption_en)

        # Hiển thị kết quả
        result_text.delete("1.0", tk.END)
        result_text.insert(tk.END, f"🔍 Các đối tượng phát hiện:
{detected_objects}\n")
        result_text.insert(tk.END, f"📝 Mô tả (EN): {caption_en}\n")
        result_text.insert(tk.END, f"📝 Mô tả (VI): {caption_vi}")
    except Exception as e:
        result_text.delete("1.0", tk.END)
        result_text.insert(tk.END, f"❌ Lỗi: {e}")
    finally:
        status_label.config(text="✅ Hoàn tất.")

# Chọn 1 ảnh và xử lý luôn
def choose_single_image():
    file_path = filedialog.askopenfilename(title="Chọn 1 ảnh",
filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")])
    if file_path:
        dropdown_menu.pack_forget()
        process_image(file_path)

# Chọn nhiều ảnh và hiển thị menu chọn
def choose_multiple_images():
    global selected_files
    selected_files = list(filedialog.askopenfilenames(title="Chọn nhiều
ảnh", filetypes=[("Image files", "*.jpg *.jpeg *.png *.bmp")]))
    if not selected_files:
        return

    # Hiện menu để chọn 1 ảnh để xử lý
    selected_var.set(selected_files[0])
    dropdown_menu['menu'].delete(0, 'end')
    for path in selected_files:
        display_name = path.split("/")[-1] if "/" in path else
path.split("\\")[-1]
        dropdown_menu['menu'].add_command(label=display_name, command=lambda
p=path: on_dropdown_select(p))
    dropdown_menu.pack(pady=5)

# Khi người dùng chọn ảnh từ menu
```

```
def on_dropdown_select(file_path):
    selected_var.set(file_path)
    process_image(file_path)

# ======= GIAO DIỆN =======
root = tk.Tk()
root.title("📷 Nhận diện ngữ cảnh bằng YOLOv8 + BLIP + Translate")

panel = tk.Label(root)
panel.pack()

# Hai nút: Chọn 1 ảnh / nhiều ảnh
btn_single = tk.Button(root, text="🖼️ Chọn 1 ảnh",
command=choose_single_image, height=2, width=20, bg='lightgreen')
btn_single.pack(pady=5)

btn_multi = tk.Button(root, text="🖼️ Chọn nhiều ảnh",
command=choose_multiple_images, height=2, width=20, bg='orange')
btn_multi.pack(pady=5)

# Menu chọn ảnh nếu nhiều
selected_var = tk.StringVar(root)
dropdown_menu = tk.OptionMenu(root, selected_var, "")

status_label = tk.Label(root, text="", font=("Arial", 10), fg="blue")
status_label.pack()

result_text = tk.Text(root, height=10, width=60, wrap=tk.WORD)
result_text.pack(padx=10, pady=10)

root.mainloop()
```

9. Dịch mô tả ảnh sang tiếng Việt

Sử dụng thư viện googletrans:

```
def translate_to_vietnamese(text):
    translator = Translator()
    result = translator.translate(text, src='en', dest='vi')
    return result.text
```

10. Kết quả thực nghiệm

- Đầu vào: ảnh chứa nhiều vật thể
- YOLOv8 phát hiện và phân loại các vật thể chính xác
- BLIP sinh mô tả dựa trên kết quả ảnh

- Mô tả được dịch sang tiếng Việt dễ hiểu

11. Thư viện cần cài đặt

```
pip install ultralytics
pip install pycocotools
pip install googletrans==4.0.0-rc1
pip install torch torchvision
pip install tqdm
pip install transformers
pip install opencv-python
pip install pillow
```

12. Tổng kết

- Hệ thống đã hoàn thiện các chức năng chính: phát hiện vật thể, mô tả ảnh, dịch sang tiếng Việt và giao diện người dùng
- Ứng dụng có thể được mở rộng để mô tả video, gợi ý nội dung tự động, hỗ trợ người khiếm thị.

13. Hướng phát triển

- Kết hợp mô tả từ nhiều ảnh để tạo ra bối cảnh tổng thể
- Tăng cường độ chính xác mô hình bằng dữ liệu chuyên biệt
- Chuyển hệ thống sang ứng dụng web hoặc mobile

14. Kết luận và Đánh giá hệ thống

14.1. Kết luận

Hệ thống nhận diện ngữ cảnh từ hình ảnh sử dụng kết hợp hai mô hình hiện đại là YOLOv8 (phát hiện vật thể) và BLIP (sinh mô tả ảnh) đã cho thấy hiệu quả trong việc tự động phân tích và mô tả nội dung hình ảnh. Thông qua giao diện Tkinter thân thiện, người dùng có thể

dễ dàng chọn nhiều ảnh, nhận kết quả mô tả bằng tiếng Anh và bản dịch tiếng Việt, giúp tiếp cận thông tin trực quan, đặc biệt hữu ích trong giáo dục, du lịch, và hỗ trợ người khiếm thị.

14.2. Đánh giá hệ thống

- **Ưu điểm**:
    - Tích hợp nhiều công nghệ hiện đại trong xử lý ảnh và ngôn ngữ tự nhiên.
    - Giao diện thân thiện, dễ sử dụng.
    - Có thể xử lý nhiều ảnh cùng lúc, phù hợp với nhu cầu thực tế.
    - Kết quả mô tả tương đối chính xác, phù hợp với nội dung hình ảnh.
- **Nhược điểm**:
    - Hiệu năng phụ thuộc vào tài nguyên phần cứng (vì sử dụng mô hình deep learning).
    - Mô tả chưa thật sự sâu hoặc có thể bị lệch khi ảnh có nhiều đối tượng không rõ ràng.
    - Chất lượng dịch phụ thuộc vào độ chính xác của API dịch tự động.
- **Khả năng ứng dụng thực tế**:
    - Trong giáo dục, giúp học sinh hiểu nội dung hình ảnh trực quan.
    - Trong y tế, có thể hỗ trợ mô tả ảnh chụp X-quang, CT.
    - Trong lĩnh vực hỗ trợ người khuyết tật, đặc biệt là người khiếm thị.

15. Tài liệu tham khảo

1. Ultralytics YOLOv8: https://docs.ultralytics.com

2. BLIP Model - Salesforce: https://github.com/salesforce/BLIP

3. Googletrans API: https://py-googletrans.readthedocs.io/en/latest/

4. COCO Dataset: https://cocodataset.org

5. Tkinter GUI Documentation: https://docs.python.org/3/library/tkinter.html