PROJECT REPORT

December 31, 2025

Báo cáo cuối kỳ môn học: PYTHON CHO KHOA HỌC DỮ LIỆU

Lớp 23TTH, Khoa Toán - Tin học, Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Đề tài thực hiện:

USING DEEP LEARNING TO CLASSIFY ANIMAL AND HUMAN IMAGES

Giảng viên hướng dẫn: ThS. Hà Văn Thảo

Danh sách thành viên nhóm:

1. 23110114 - Nguyễn Thị Hồng Thắm
2. 23110123 - Lê Huỳnh Yến Vy
3. 23110132 - Trần Nhật Anh

0.1 GIỚI THIỆU

Object detection là một trong những chủ đề "nóng" trong deep learing bởi tính ứng dụng cao trong thực tiễn và nguồn dữ liệu dồi dào, dễ chuẩn bị. Một trong những thuật toán object detection nổi tiếng nhất là **YOLO**.

YOLO là mô hình mạng neuron tích chập (CNN) được sử dụng phổ biển để nhận dạng các đối tượng trong ảnh hoặc video. Điểm đặc biệt của mô hình này là có khả năng phát hiện tất cả các đối tượng trong một hình ảnh chỉ qua một lần lan truyền của CNN.

Các phương pháp truyền thống tách biệt bước đề xuất vùng và bước phân loại, YOLO xử lý đầu vào, vừa phân loại được các đối tượng, vừa dự đoán được vị trí của chúng trong một lần duy nhất.

YOLO có nghĩa là "You only look once", nghĩa là chỉ cần "nhìn" một lần là thuật toán đã có thể phát hiện được vật thể, cho thấy độ nhanh của thuật toán gần như là real-time.

Ứng dụng của YOLO cũng như nhiều thuật toán object detection khác, rất đa dạng: quản lý giao thông, đếm số sản phẩm trên băng chuyền nhà máy, đếm số vật nuôi trong chăn nuôi, phát hiện vật thể nguy hiểm (súng, dao,...), chấm công tự động,...

0.2 TẠO MÔI TRƯỜNG ẢO VÀ KERNEL CHẠY NOTEBOOK (LINUX)

Dự án Python cần **môi trường ảo (virtual environment)** để tự cách ly, tránh xung đột phiên bản thư viện giữa các dự án. `venv` là môi trường ảo mà chúng ta sẽ sử dụng trong dự án này. Sau

khi cài đặt `venv`, chúng ta di chuyển đường dẫn đến folder chứa dự án trong terminal và sử dụng lệnh sau để cài đặt môi trường ảo cho dự án:

```
python -m venv .venv
```

Trong đó, `.venv` là tên của folder chứa môi trường ảo của dự án, đồng thời nó cũng sẽ "đóng băng" phiên bản Python, pip và các thư viện sẽ được dùng trong dự án.

Kích hoạt môi trường ảo:

```
source .venv/bin/activate
```

Lúc này, phiên bản Python và `pip` được dùng là của môi trường ảo, các thư viện cài bằng `pip install` cũng chỉ ảnh hưởng trong `.venv`. Cách nhận biết đang ở môi trường ảo là promt terminal thường đổi thành `(.venv) user_name@machine:~` (nếu đang sử dụng Linux). Khi đã kích hoạt môi trường ảo, đảm bảo phiên bản Python và `pip` đã "đóng băng" trong đó, sử dụng lệnh:

```
which python && which pip
```

Nếu output có dạng `.../<project_name>/.venv/...` thì môi trường ảo đã được kích hoạt thành công.

Tiếp theo, tạo một kernel để chạy Jupyter Notebook. Cài đặt `ipykernel` để tạo kernel:

```
python -m pip install ipykernel
```

Sau khi cài đặt thành công, tiến hành tạo kernel để chạy file `.ipynb`:

```
python -m ipykernel install --prefix .venv --name yolovenv --display-name
"this_project"
```

`--prefix .venv`: kernel mặc định không tự lưu vào `.venv`, thuộc tính này sẽ lưu kernel đã tạo vào `.venv`
`--name yolovenv`: tên folder chứa kernel, ở đây tên folder là `yolovenv`. Kernel sẽ được lưu tại `.venv/share/jupyter/kernels/yolovenv/`
`--display-name "this_project`: kernel sẽ hiển thị dưới tên `this_project` trong VS Code.

Khi đã tạo kernel, click vào biểu tượng kernel ở góc trên bên phải, chọn

Select Another Kernel → Jupyter Kernel... → this_project

0.3 KHAI BÁO THƯ VIỆN VÀ CHUẨN BỊ DỮ LIỆU

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from ultralytics import YOLO

from pathlib import Path
```

0.4 KHAI BÁO, HUẤN LUYỆN VÀ LƯU MÔ HÌNH

Hàm train model.

[2]:
```python
def train(model, project_path, project_name):
    model.train(
        data="dataset/",
        epochs=100,
        imgsz=640,
        device=0,  # Sử dụng GPU
        project=project_path,
        name=project_name
    )

    return model
```

Sử dụng model YOLO11s.

[3]:
```python
BASE_MODEL = "yolo11s.pt"
MODEL_PATH = Path("runs/yolo11s_custom/my_model/weights/best.pt")
# Lưu model tại "runs/yolo11s_custom/my_model" sau khi train
PROJECT_PATH = "runs/yolo11s_custom"
PROJECT_NAME = "my_model"

# Huấn luyện mô hình với dataset
if MODEL_PATH.exists():
    print(f"Model has been trained already. Model is being loaded again:␣
 ↪{MODEL_PATH}")
    model = YOLO(str(MODEL_PATH))
else:
    print("Model hasn't been trained. Start training...")

    # Train model dựa trên model gốc là YOLO11s
    model = YOLO(str(BASE_MODEL))
    train(model, PROJECT_PATH, PROJECT_NAME)

    assert MODEL_PATH.exists()  # Load lại best.pt sau khi train
    model = YOLO(str(MODEL_PATH))

# Kiểm tra và khoá model với model gốc là YOLO11s
assert model.model.yaml['name'] == 'yolo11s'

model.info()
```

```
Model hasn't been trained. Start training…
Ultralytics 8.3.243  Python-3.14.2 torch-2.9.1+cu128 CUDA:0 (NVIDIA GeForce GTX
1650, 3716MiB)
engine/trainer: agnostic_nms=False, amp=True, augment=False,
auto_augment=randaugment, batch=16, bgr=0.0, box=7.5, cache=False, cfg=None,
classes=None, close_mosaic=10, cls=0.5, compile=False, conf=None,
copy_paste=0.0, copy_paste_mode=flip, cos_lr=False, cutmix=0.0, data=dataset/,
degrees=0.0, deterministic=True, device=0, dfl=1.5, dnn=False, dropout=0.0,
```

```
dynamic=False, embed=None, epochs=100, erasing=0.4, exist_ok=False, fliplr=0.5,
flipud=0.0, format=torchscript, fraction=1.0, freeze=None, half=False,
hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, imgsz=640, int8=False, iou=0.7, keras=False,
kobj=1.0, line_width=None, lr0=0.01, lrf=0.01, mask_ratio=4, max_det=300,
mixup=0.0, mode=train, model=yolo11s.pt, momentum=0.937, mosaic=1.0,
multi_scale=False, name=my_model, nbs=64, nms=False, opset=None, optimize=False,
optimizer=auto, overlap_mask=True, patience=100, perspective=0.0, plots=True,
pose=12.0, pretrained=True, profile=False, project=runs/yolo11s_custom,
rect=False, resume=False, retina_masks=False, save=True, save_conf=False,
save_crop=False,
save_dir=/home/buddyy/coding_corner/python_for_data_science/project/classify-
animal-and-human-images/runs/yolo11s_custom/my_model, save_frames=False,
save_json=False, save_period=-1, save_txt=False, scale=0.5, seed=0, shear=0.0,
show=False, show_boxes=True, show_conf=True, show_labels=True, simplify=True,
single_cls=False, source=None, split=val, stream_buffer=False, task=detect,
time=None, tracker=botsort.yaml, translate=0.1, val=True, verbose=True,
vid_stride=1, visualize=False, warmup_bias_lr=0.1, warmup_epochs=3.0,
warmup_momentum=0.8, weight_decay=0.0005, workers=8, workspace=None
```

```
---------------------------------------------------------------------------
IsADirectoryError                         Traceback (most recent call last)
File ~/coding_corner/python_for_data_science/project/
 ↪classify-animal-and-human-images/.venv/lib64/python3.14/site-packages/
 ↪ultralytics/engine/trainer.py:649, in BaseTrainer.get_dataset(self)
    643 elif str(self.args.data).rsplit(".", 1)[-1] in {"yaml", "yml"} or self.
 ↪args.task in {
    644     "detect",
    645     "segment",
    646     "pose",
    647     "obb",
    648 }:
--> 649     data = check_det_dataset(self.args.data)
    650     if "yaml_file" in data:

File ~/coding_corner/python_for_data_science/project/
 ↪classify-animal-and-human-images/.venv/lib64/python3.14/site-packages/
 ↪ultralytics/data/utils.py:404, in check_det_dataset(dataset, autodownload)
    403 extract_dir = ""
--> 404 if zipfile.is_zipfile(file) or is_tarfile(file):
    405     new_dir = safe_download(file, dir=DATASETS_DIR, unzip=True,␣
 ↪delete=False)

File /usr/lib64/python3.14/tarfile.py:3028, in is_tarfile(name)
   3027 else:
-> 3028     t = open(name)
   3029 t.close()
```

```
File /usr/lib64/python3.14/tarfile.py:1897, in TarFile.open(cls, name, mode,␣
 ↪fileobj, bufsize, **kwargs)
   1896 try:
-> 1897     return func(name,     , fileobj, **kwargs)
   1898 except (ReadError, CompressionError) as e:

File /usr/lib64/python3.14/tarfile.py:1972, in TarFile.gzopen(cls, name, mode,␣
 ↪fileobj, compresslevel, **kwargs)
   1971 try:
-> 1972     fileobj = GzipFile(name, mode +     , compresslevel, fileobj)
   1973 except OSError as e:

File /usr/lib64/python3.14/gzip.py:208, in GzipFile.__init__(self, filename,␣
 ↪mode, compresslevel, fileobj, mtime)
    207 if fileobj is None:
--> 208     fileobj = self.myfileobj = builtins.open(filename, mode or     )
    209 if filename is None:

IsADirectoryError: [Errno 21] Is a directory: 'dataset/'

The above exception was the direct cause of the following exception:

RuntimeError                              Traceback (most recent call last)
Cell In[3], line 16
     14 # Train model dựa trên model gốc là YOLO11s
     15 model = YOLO(str(BASE_MODEL))
---> 16 train(model, PROJECT_PATH, PROJECT_NAME)
     18 assert MODEL_PATH.exists()  # Load lại best.pt sau khi train
     19 model = YOLO(str(MODEL_PATH))

Cell In[2], line 2, in train(model, project_path, project_name)
      1 def train(model, project_path, project_name):
----> 2     model.train(
      3         data=           ,
      4         epochs=100,
      5         imgsz=640,
      6         device=0,  # Sử dụng GPU
      7         project=project_path,
      8         name=project_name
      9     )
     11     return model

File ~/coding_corner/python_for_data_science/project/
 ↪classify-animal-and-human-images/.venv/lib64/python3.14/site-packages/
 ↪ultralytics/engine/model.py:768, in Model.train(self, trainer, **kwargs)
    765 if args.get("resume"):
    766     args["resume"] = self.ckpt_path
```

```
--> 768 self.trainer =␣
 ↪(trainer or self._smart_load(          ))(overrides=args, _callbacks=self.call backs)
    769 if not args.get("resume"):  # manually set model only if not resuming
    770     self.trainer.model = self.trainer.get_model(weights=self.model if␣
 ↪self.ckpt else None, cfg=self.model.yaml)

File ~/coding_corner/python_for_data_science/project/
 ↪classify-animal-and-human-images/.venv/lib64/python3.14/site-packages/
 ↪ultralytics/models/yolo/detect/train.py:63, in DetectionTrainer.__init__(self ␣
 ↪cfg, overrides, _callbacks)
     55 def __init__(self, cfg=DEFAULT_CFG, overrides: dict[str, Any] | None =␣
 ↪None, _callbacks=None):
     56     """Initialize a DetectionTrainer object for training YOLO object␣
 ↪detection models.
     57
     58     Args:
   (…)     61         _callbacks (list, optional): List of callback functions t␣
 ↪be executed during training.
     62     """
---> 63     super().__init__(cfg, overrides, _callbacks)

File ~/coding_corner/python_for_data_science/project/
 ↪classify-animal-and-human-images/.venv/lib64/python3.14/site-packages/
 ↪ultralytics/engine/trainer.py:163, in BaseTrainer.__init__(self, cfg,␣
 ↪overrides, _callbacks)
    161 self.model = check_model_file_from_stem(self.args.model)  # add suffix,␣
 ↪i.e. yolo11n -> yolo11n.pt
    162 with torch_distributed_zero_first(LOCAL_RANK):  # avoid auto-downloadin␣
 ↪dataset multiple times
--> 163     self.data = self.get_dataset()
    165 self.ema = None
    167 # Optimization utils init

File ~/coding_corner/python_for_data_science/project/
 ↪classify-animal-and-human-images/.venv/lib64/python3.14/site-packages/
 ↪ultralytics/engine/trainer.py:653, in BaseTrainer.get_dataset(self)
    651             self.args.data = data["yaml_file"]  # for validating 'yolo␣
 ↪train data=url.zip' usage
    652 except Exception as e:
--> 653     raise RuntimeError(emojis(f"Dataset '{clean_url(self.args.data)}'␣
 ↪error  {e}")) from e
    654 if self.args.single_cls:
    655     LOGGER.info("Overriding class names with single class.")

RuntimeError: Dataset 'dataset' error  [Errno 21] Is a directory: 'dataset/'
```

[ ]:

0.5 TEST