CS231

Phân loại sản phẩm thời trang

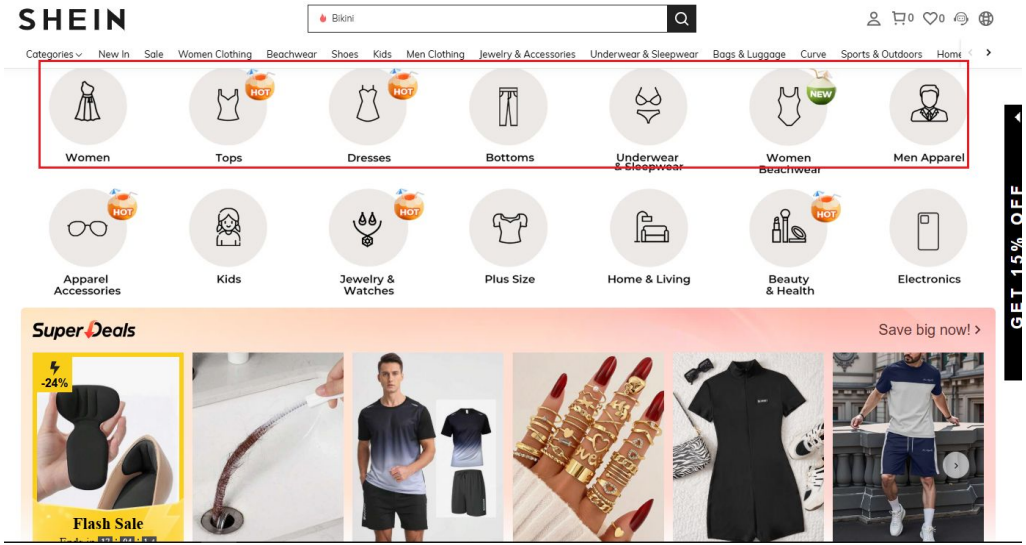
Giảng viên hướng dẫn: TS Mai Tiến Dũng
Sinh viên thực hiện: Võ Quốc Thịnh

Nội dung

01

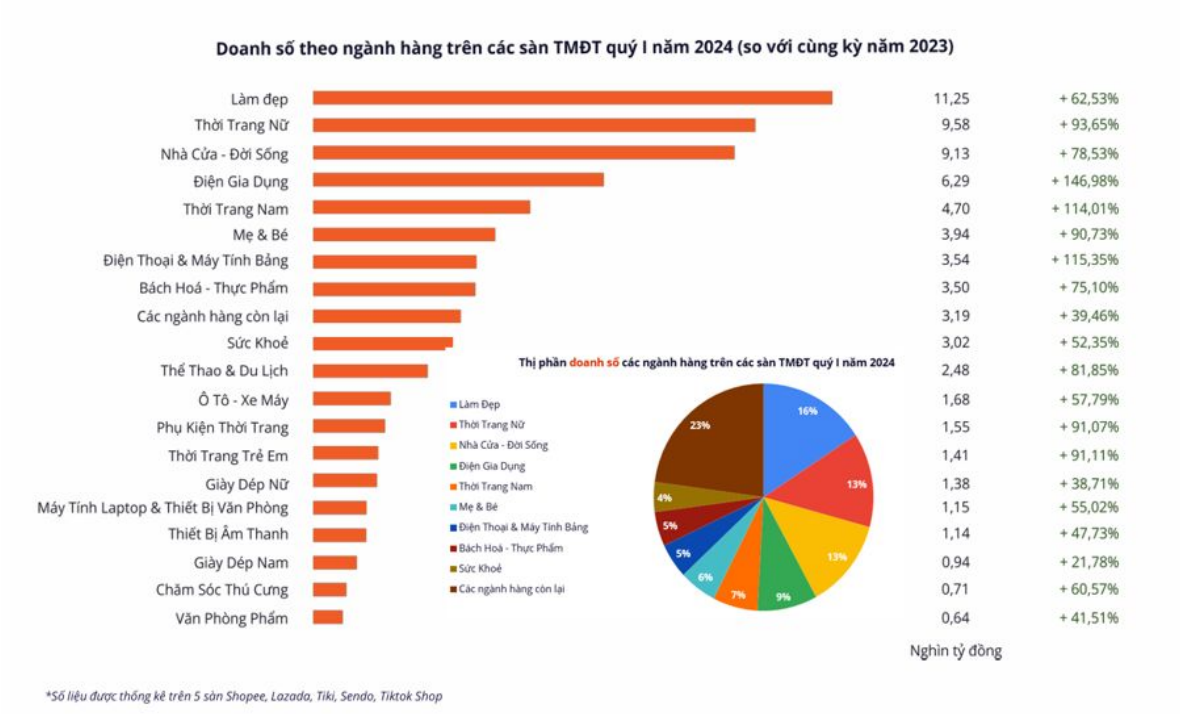
Giới thiệu bài toán

Giới thiệu bài toán



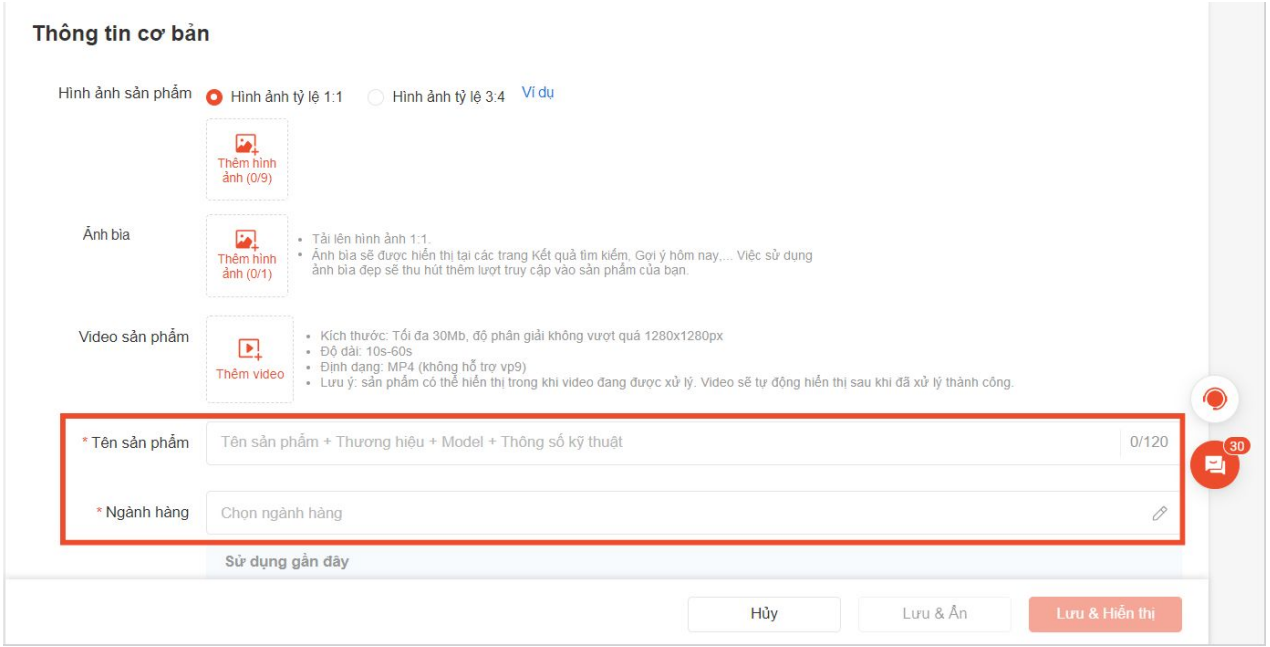
Các nền tảng mua sắm trực tuyến như Shopee, Tiki, Lazada, hay các website thương hiệu thời trang lớn liên tục cập nhật hàng ngàn sản phẩm mỗi ngày.

Thực trạng



Ngành hàng thời trang chiếm tỉ trọng cao trên các sàn TMĐT

Thực trạng



Người bán phải chọn thủ công danh mục sản phẩm thời trang trên sàn

Mục tiêu

Phân loại sản phẩm thời trang

- Xây dựng một mô hình phân loại tự động các sản phẩm thời trang dựa trên hình ảnh.
- Phân tích kết quả và đánh giá khả năng áp dụng mô hình vào thực tế, đặc biệt là trong các hệ thống thương mại điện tử cần phân loại số lượng lớn hình ảnh sản phẩm.

Ý nghĩa của đề tài

Cắt giảm chi phí khi phân loại sản phẩm thời trang thủ công.

Tăng trải nghiệm người dùng.

Phát biểu bài toán

Input: Một ảnh RGB của sản phẩm thời trang (kích thước tùy biến).

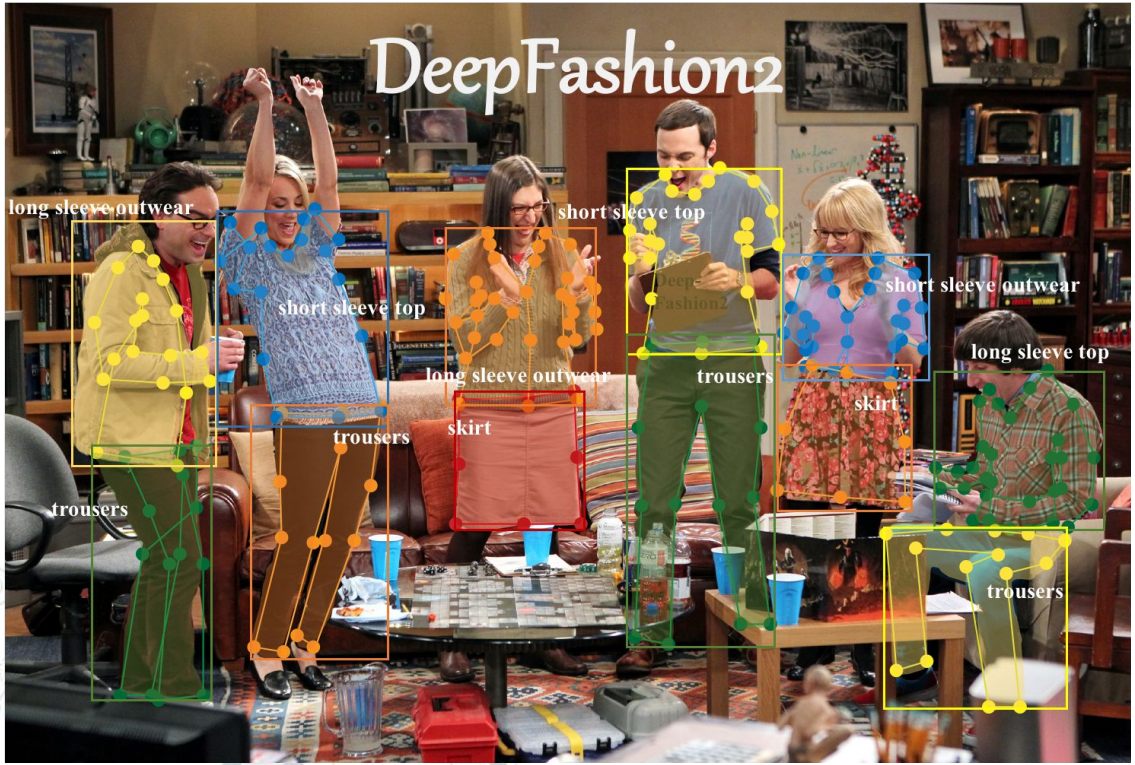
Output:

- Nhãn lớp dự đoán (ví dụ: "áo sơ mi", "quần dài", "váy ngắn", v.v.) thuộc một trong 13 lớp của tập dữ liệu.

02

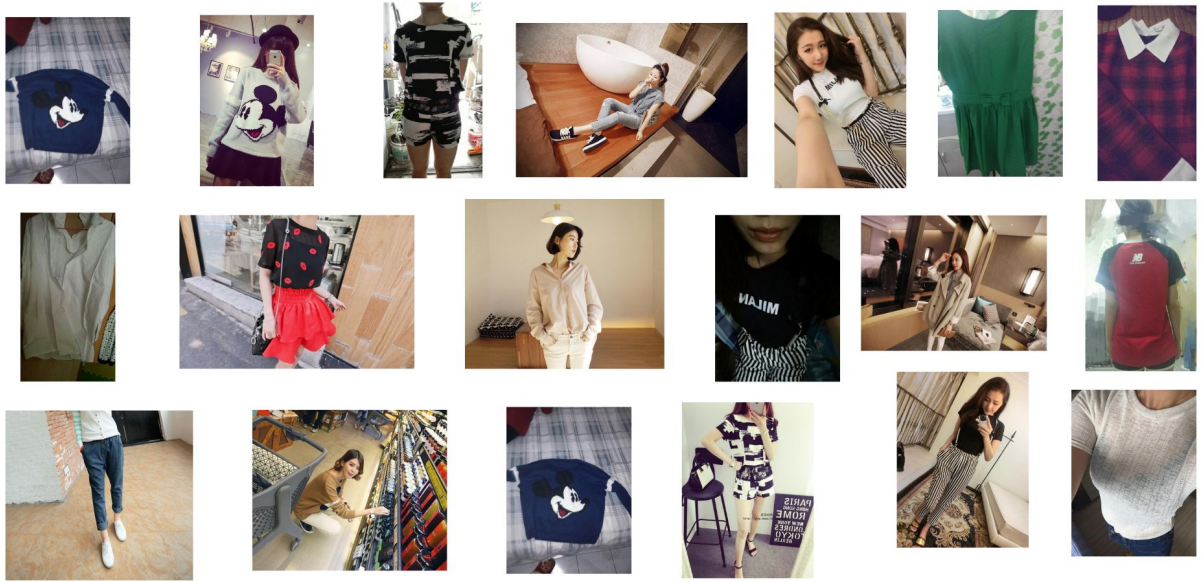
Tổng quan dữ liệu

DeepFashion2



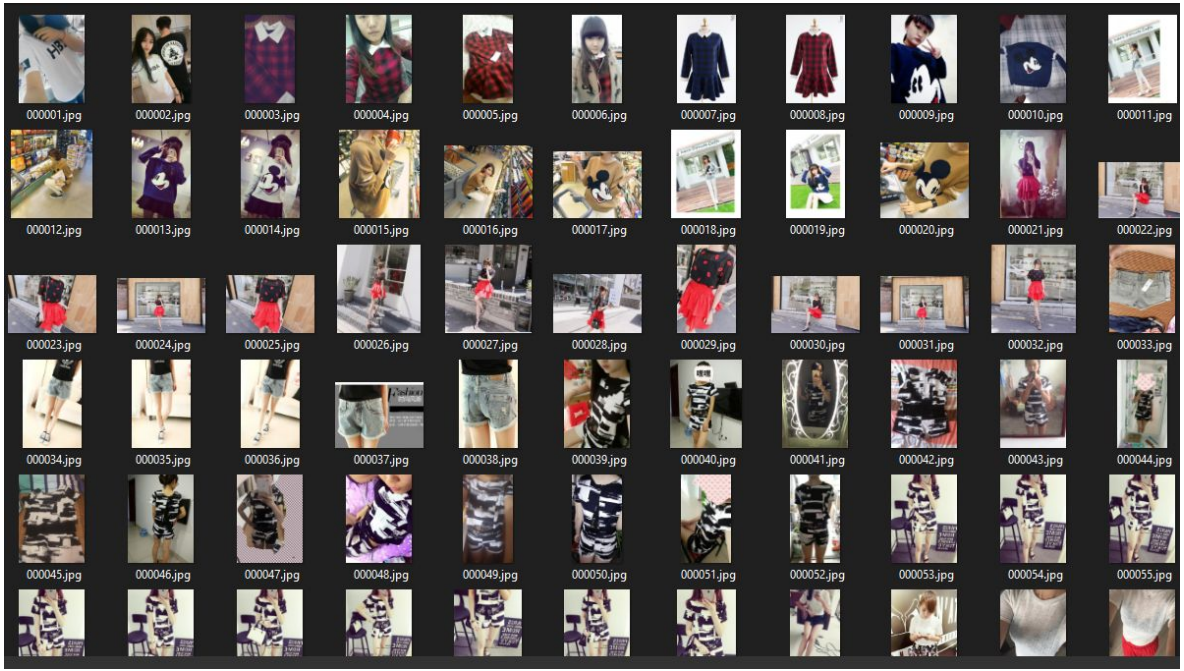
- Gần 800.000 ảnh.
- 13 nhãn.
- Được cải tiến từ DeepFashion.
- Thông tin chi tiết: keypoints, class, bounding box, mask

Tổng quan dữ liệu



Một số hình ảnh trong bộ dữ liệu

Tổng quan dữ liệu



Các cặp ảnh giống nhau ở điều kiện khác nhau

(ví dụ: góc chụp, tư thế người mẫu, ánh sáng...)

Tổng quan dữ liệu

short sleeve top

long sleeve top

short sleeve outwear

long sleeve outwear

vest

sling

Tổng quan dữ liệu

shorts

trousers

skirt

short sleeve dress

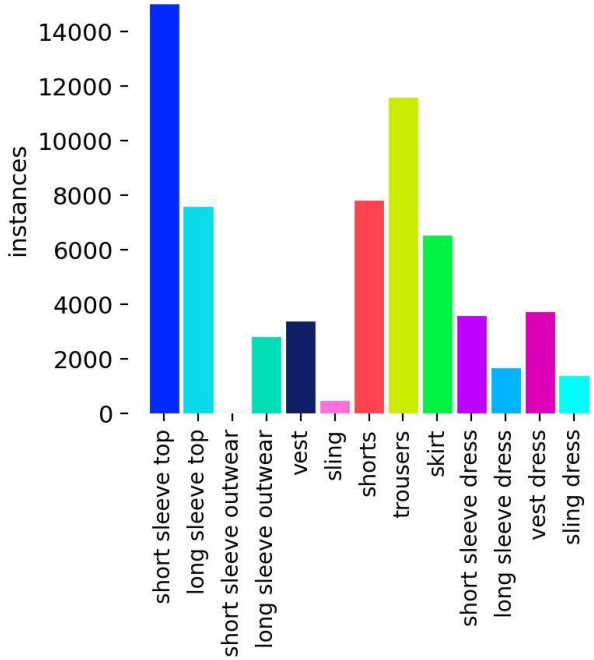
long sleeve dress

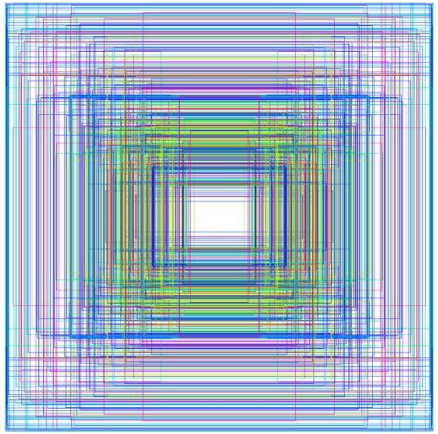
vest dress

sling dress

Dữ liệu tuỳ chỉnh

Train: 40.000 ảnh. Val: 10.000 ảnh

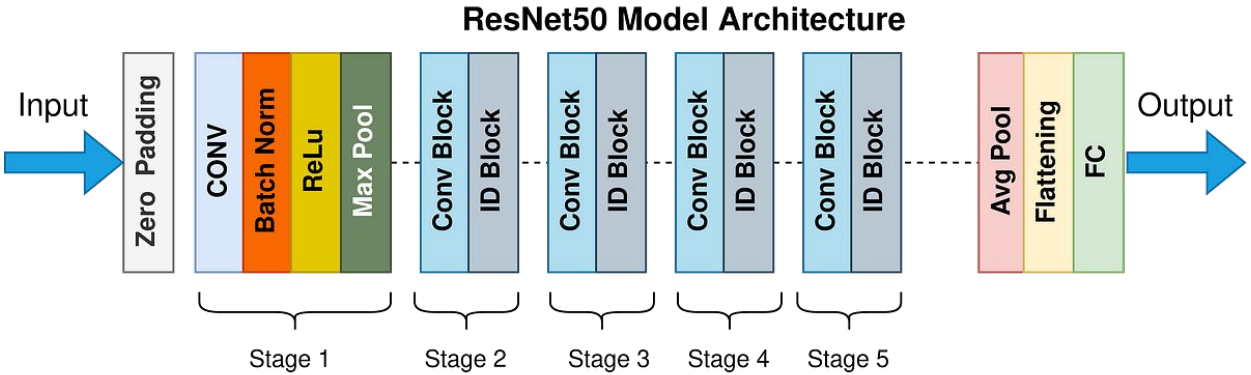




03

Thực nghiệm

Giới thiệu ResNet50



ResNet (Residual Network) là một trong những mạng CNN nổi tiếng do nhóm Microsoft Research giới thiệu vào năm 2015, giành chiến thắng tại cuộc thi ImageNet với độ chính xác vượt trội.

Huấn luyện ResNet50 / Thông số cơ bản


```
NUM_CLASSES = 13
IMG_SIZE = 224
BATCH_SIZE = 128
NUM_EPOCHS = 10
VAL_SPLIT = 0.2
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Huấn luyện ResNet50 / Thông số cơ bản


```
1   NUM_CLASSES = 13
2   IMG_SIZE = 224
3   BATCH_SIZE = 128
4   NUM_EPOCHS = 10
5   VAL_SPLIT = 0.2
6   DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Các thông số cơ bản như:

- Số class: 13.
- Kích thước ảnh: 224
- Batch Size: 128
- Số epoch:10
- Tỉ lệ chia train/val: 80/20

Huấn luyện ResNet50 / Thông số cơ bản

```python
train_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

val_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Huấn luyện ResNet50 / Thông số cơ bản

```
train_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

val_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Size ảnh về 224x224

Huấn luyện ResNet50 / Thông số cơ bản


```python
train_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

val_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Tăng cường dữ liệu tự động: xoay và lật

Huấn luyện ResNet50 / Thông số cơ bản

```python
train_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

val_transforms = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Chuẩn hóa từng kênh màu (RGB) của ảnh đầu vào theo mean và std (độ lệch chuẩn) của tập dữ liệu ImageNet

$$\text{pixel\_normalized} = \frac{\text{pixel\_original} - \text{mean}}{\text{std}}$$

Huấn luyện ResNet50 / Load dataset

```python
class CustomImageDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform
        self.classes = sorted(os.listdir(root_dir))
        self.class_to_idx = {cls_name: i for i, cls_name in enumerate(self.classes)}
        self.images = []

        for cls_name in self.classes:
            class_path = os.path.join(root_dir, cls_name)
            for img_name in os.listdir(class_path):
                if img_name.endswith(('.png', '.jpg', '.jpeg')):
                    self.images.append((os.path.join(class_path, img_name), cls_name))

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        img_path, class_name = self.images[idx]
        image = Image.open(img_path).convert('RGB')
        label = self.class_to_idx[class_name]

        if self.transform:
            image = self.transform(image)

        return image, label
```

Huấn luyện ResNet50 / Load dataset

```python
def create_dataloaders(data_dir):
    dataset = CustomImageDataset(data_dir, transform=train_transforms)

    total_size = len(dataset)
    val_size = int(VAL_SPLIT * total_size)
    train_size = total_size - val_size
    train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

    val_dataset.dataset.transform = val_transforms

    train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True, num_workers=4)
    val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False, num_workers=4)

    return train_loader, val_loader
```

Huấn luyện ResNet50 / Tiến hành huấn luyện

```
data_dir = "/content/dataset/eff_dataset/train"

# DataLoader
train_loader, val_loader = create_dataloaders(data_dir)

model = models.resnet50(pretrained=True)
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, NUM_CLASSES)
model = model.to(DEVICE)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

model = train_model(model, train_loader, val_loader, criterion, optimizer, NUM_EPOCHS)
```

Huấn luyện ResNet50 / Tiến hành huấn luyện


```
data_dir = "/content/dataset/eff_dataset/train"

# DataLoader
train_loader, val_loader = create_dataloaders(data_dir)

model = models.resnet50(pretrained=True)
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, NUM_CLASSES)
model = model.to(DEVICE)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

model = train_model(model, train_loader, val_loader, criterion, optimizer, NUM_EPOCHS)
```

Sử dụng pretrain

Huấn luyện ResNet50 / Tiến hành huấn luyện

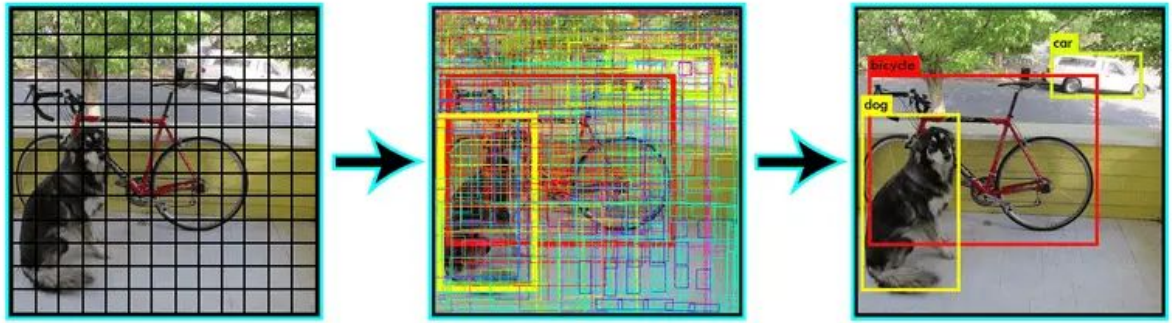
```
data_dir = "/content/dataset/eff_dataset/train"

# DataLoader
train_loader, val_loader = create_dataloaders(data_dir)

model = models.resnet50(pretrained=True)
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, NUM_CLASSES)
model = model.to(DEVICE)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

model = train_model(model, train_loader, val_loader, criterion, optimizer, NUM_EPOCHS)
```

Sử dụng Cross Entropy Loss, Adam Optimizer và Learning Rate 0.001

Huấn luyện ResNet50 / Kết quả huấn luyện


```
100%|██████████| 408/408 [09:44<00:00,  1.43s/it]
Epoch [1/10], Train Loss: 1.0056, Train Acc: 65.82%, Val Loss: 0.9468, Val Acc: 68.13%
Saved best model with accuracy: 68.13%
100%|██████████| 408/408 [09:43<00:00,  1.43s/it]
Epoch [2/10], Train Loss: 0.7531, Train Acc: 74.13%, Val Loss: 0.8069, Val Acc: 71.55%
Saved best model with accuracy: 71.55%
100%|██████████| 408/408 [09:41<00:00,  1.43s/it]
Epoch [3/10], Train Loss: 0.6370, Train Acc: 77.78%, Val Loss: 0.7529, Val Acc: 74.04%
Saved best model with accuracy: 74.04%
100%|██████████| 408/408 [09:40<00:00,  1.42s/it]
Epoch [4/10], Train Loss: 0.5568, Train Acc: 80.59%, Val Loss: 0.7333, Val Acc: 75.49%
Saved best model with accuracy: 75.49%
100%|██████████| 408/408 [09:41<00:00,  1.42s/it]
Epoch [5/10], Train Loss: 0.4755, Train Acc: 83.42%, Val Loss: 0.7824, Val Acc: 74.17%
100%|██████████| 408/408 [09:39<00:00,  1.42s/it]
Epoch [6/10], Train Loss: 0.4058, Train Acc: 85.67%, Val Loss: 0.8223, Val Acc: 73.91%
100%|██████████| 408/408 [09:38<00:00,  1.42s/it]
Epoch [7/10], Train Loss: 0.3401, Train Acc: 87.81%, Val Loss: 0.7222, Val Acc: 77.29%
Saved best model with accuracy: 77.29%
100%|██████████| 408/408 [09:38<00:00,  1.42s/it]
Epoch [8/10], Train Loss: 0.2749, Train Acc: 90.35%, Val Loss: 0.8017, Val Acc: 76.31%
100%|██████████| 408/408 [09:38<00:00,  1.42s/it]
Epoch [9/10], Train Loss: 0.2294, Train Acc: 91.79%, Val Loss: 0.8743, Val Acc: 75.99%
100%|██████████| 408/408 [09:38<00:00,  1.42s/it]
Epoch [10/10], Train Loss: 0.1903, Train Acc: 93.20%, Val Loss: 1.0503, Val Acc: 74.46%
Training completed and final model saved.
```

Giới thiệu YOLO



YOLO (You Only Look Once) là một thuật toán phát hiện đối tượng (object detection) tiên tiến, được giới thiệu lần đầu tiên vào năm 2015 bởi Joseph Redmon và các cộng sự

Tham số huấn luyện YOLO

Tên	Giá trị
Mô hình	Yolov8m, Yolov8l, Yolov5m
Epoch	30
Batch Size	64
Optimizer	AdamW
Learning Rate	0.000588
Momentum	0.9
Weight decay	0.0005
Augmentation	Blur, MedianBlur ToGray, CLAHE

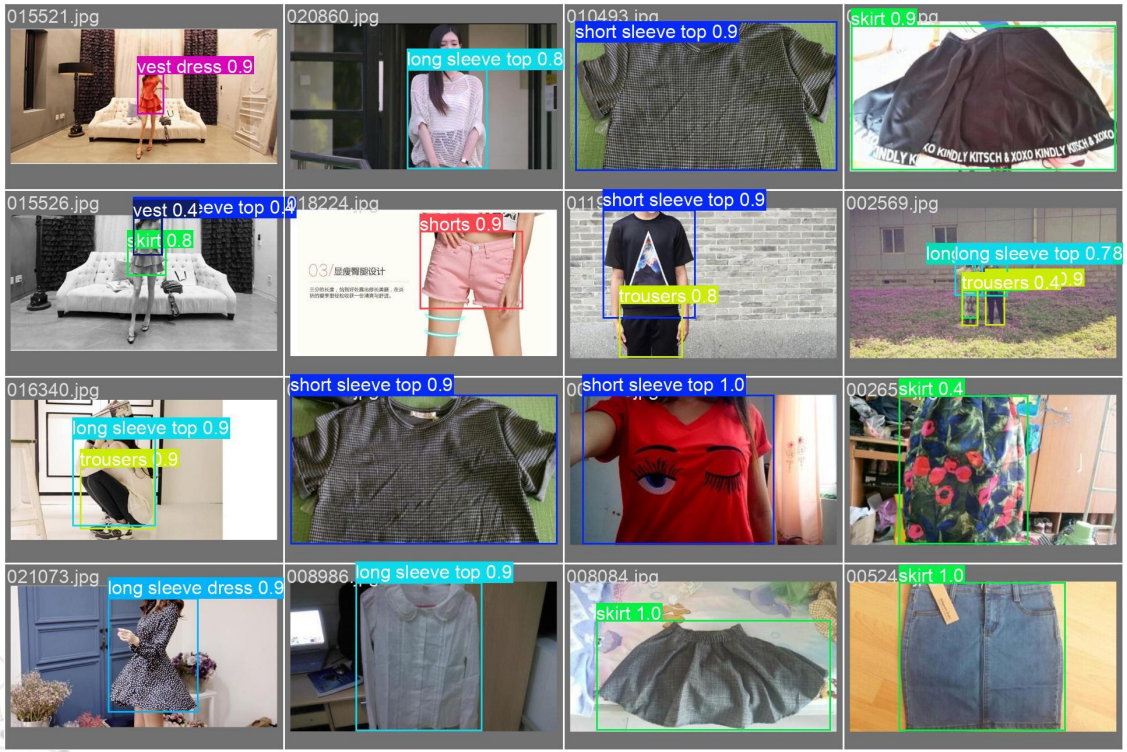
Kết quả huấn luyện YOLO

Tên phương pháp	mAP50	mAP50-95	Precision	Recall
YOLOv8m	0.759	0.657	0.745	0.695
YOLOv8l	**0.811**	**0.714**	**0.803**	**0.737**
YOLOv5m	0.746	0.632	0.755	0.688

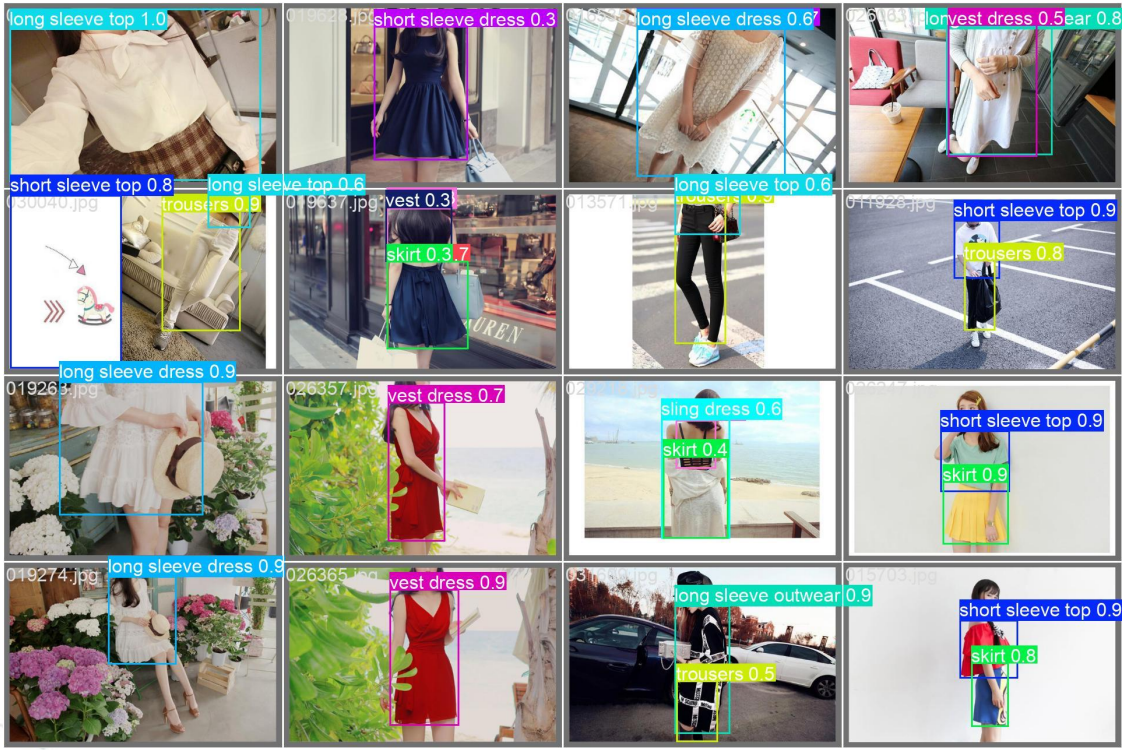
Kết quả huấn luyện YOLO

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	10000	16252	0.803	0.737	0.811	0.714
short sleeve top	3790	3842	0.909	0.889	0.95	0.865
long sleeve top	1825	1837	0.809	0.789	0.874	0.779
short sleeve outwear	37	37	0.642	0.484	0.491	0.441
long sleeve outwear	636	641	0.766	0.819	0.856	0.774
vest	640	646	0.774	0.827	0.869	0.74
sling	102	103	0.77	0.466	0.582	0.467
shorts	1310	1325	0.902	0.819	0.927	0.772
trousers	2957	2987	0.906	0.914	0.964	0.808
skirt	1976	1985	0.846	0.832	0.902	0.798
short sleeve dress	945	959	0.815	0.736	0.828	0.775
long sleeve dress	477	482	0.65	0.622	0.667	0.611
vest dress	1054	1067	0.817	0.731	0.847	0.77
sling dress	340	341	0.831	0.654	0.787	0.684

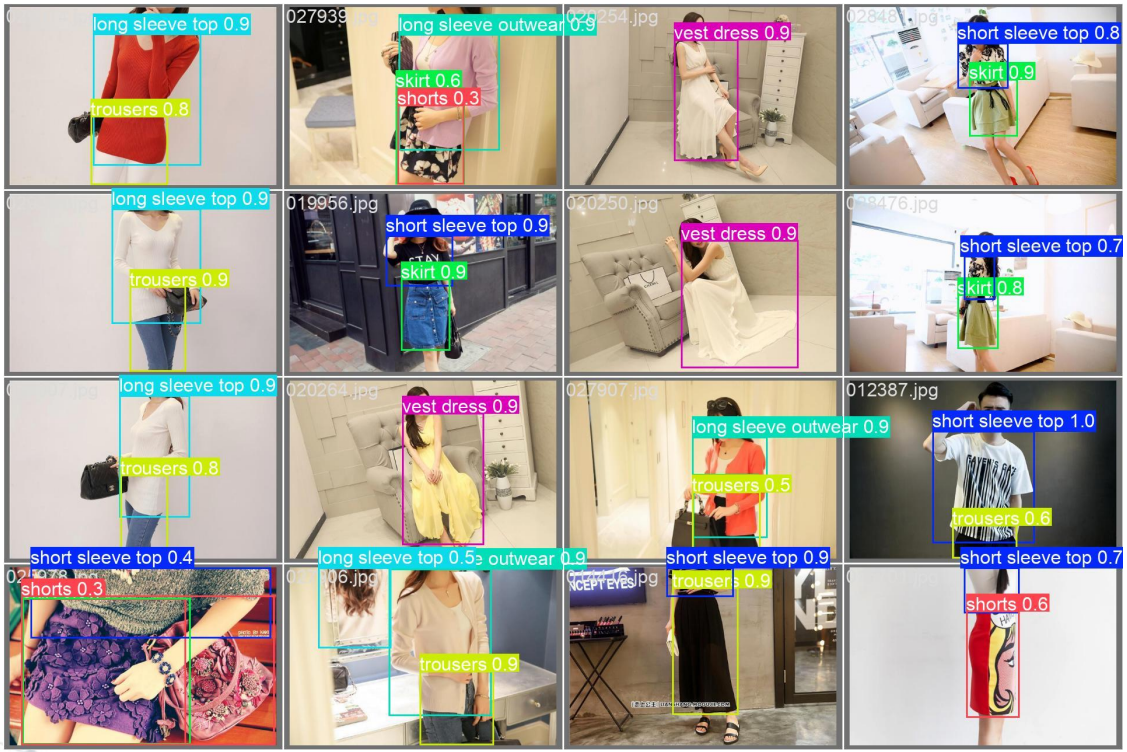
Một số hình ảnh dự đoán của YOLO



Một số hình ảnh dự đoán của YOLO



Một số hình ảnh dự đoán của YOLO



04

Kết Luận

Kết luận

ResNet50 (CNN):

- Accuracy: 77.29% trên dữ liệu phân loại cắt sẵn.
- Tốt cho bài toán classification đơn lẻ.

YOLOv8:

- Kết hợp phát hiện + phân loại trong một mô hình.
- Tốc độ nhanh, hiệu quả cao khi áp dụng vào ứng dụng thực tế.
- Nhận diện tốt sản phẩm ngay cả khi ảnh có nhiều vật thể hoặc không cắt sẵn.

Hướng phát triển

- Fine-tuning sâu hơn, tăng số epoch.
- Kết hợp metadata hoặc mô tả sản phẩm.
- Triển khai thành ứng dụng thực tế (gợi ý thời trang, phân loại ảnh sản phẩm).
- Sử dụng toàn bộ tập dữ liệu DeepFashion2 khi có đủ tài nguyên.

05

Q&A và Demo

THANKS FOR LISTENING!

Does anyone have any questions?

22520006@gm.uit.edu.vn