Python for Data Science and AI

OBJECT DETECTION WITH DETR-RESNET-50

By Lee Kuan I (20229022)
Gao Zihao (21223514)
Siripong Achariyasilp (21224501)

Require ment

Presentation Requirements

Deliverables:

1. Jupyter Notebook + Github Repository;
2. Slides: at least 10 slides;
3. Oral presentation: 15 mins

Jupyter Notebook Guidelines:

1. Text: Markdown syntax required, clear titles, labels
2. Design: Consistent color schemes, graphs (at least two)
3. Code: neat and well-documented

Outline of presentations:

1. Introduction: hugging face itself and importance of it + model
2. Problem and Motivation: for us, it can be the car accident detection
3. Data and Preprocessing: dataset, and how to prepare(preprocess?)
4. Model selection and training: why, how to train the model and evaluate performance?
5. Results and Discussion: What did you learn, implications of findings
6. Conclusion and future works: Main takeways, potential future directions

---

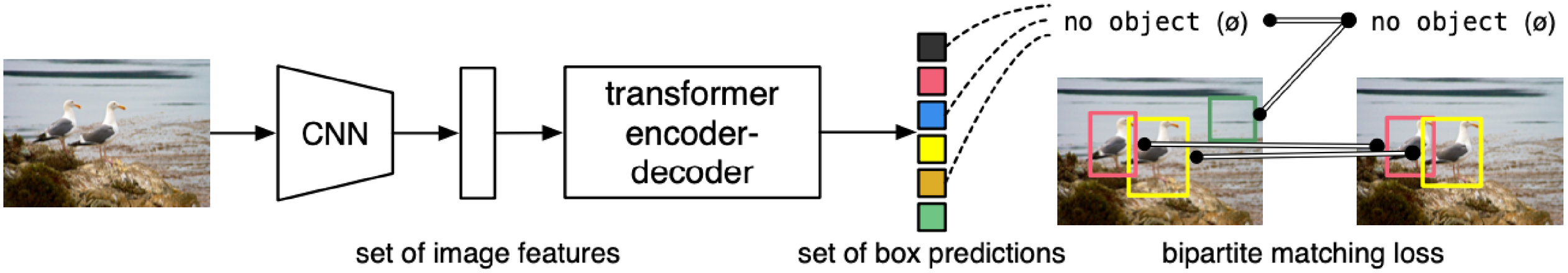
WHAT IS DETR

End-to-End Object Detection with Transformers

DEtection TRansformer

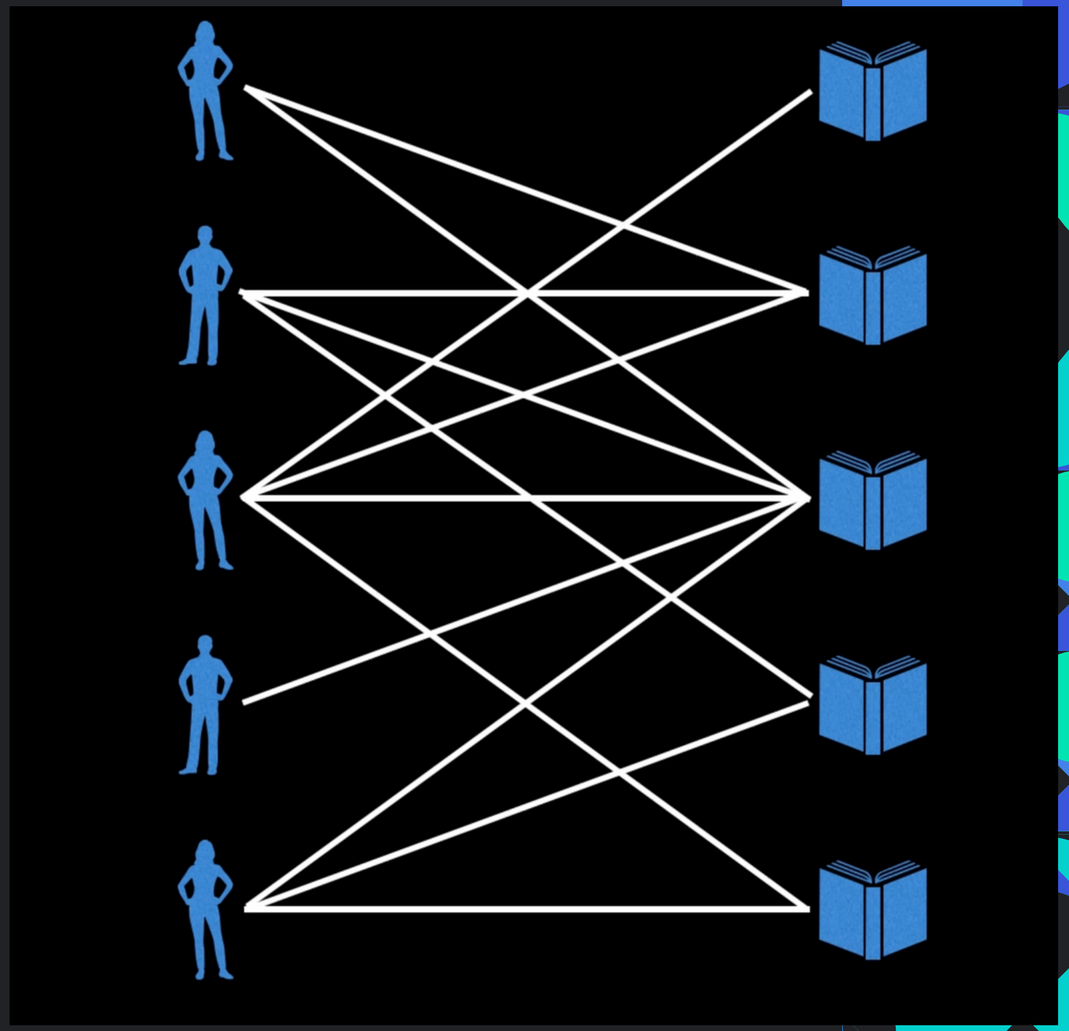
A specific type of model designed for object detection in computer vision. Object detection is a task where a computer system identifies and locates objects within an image

HOW IT WORKS

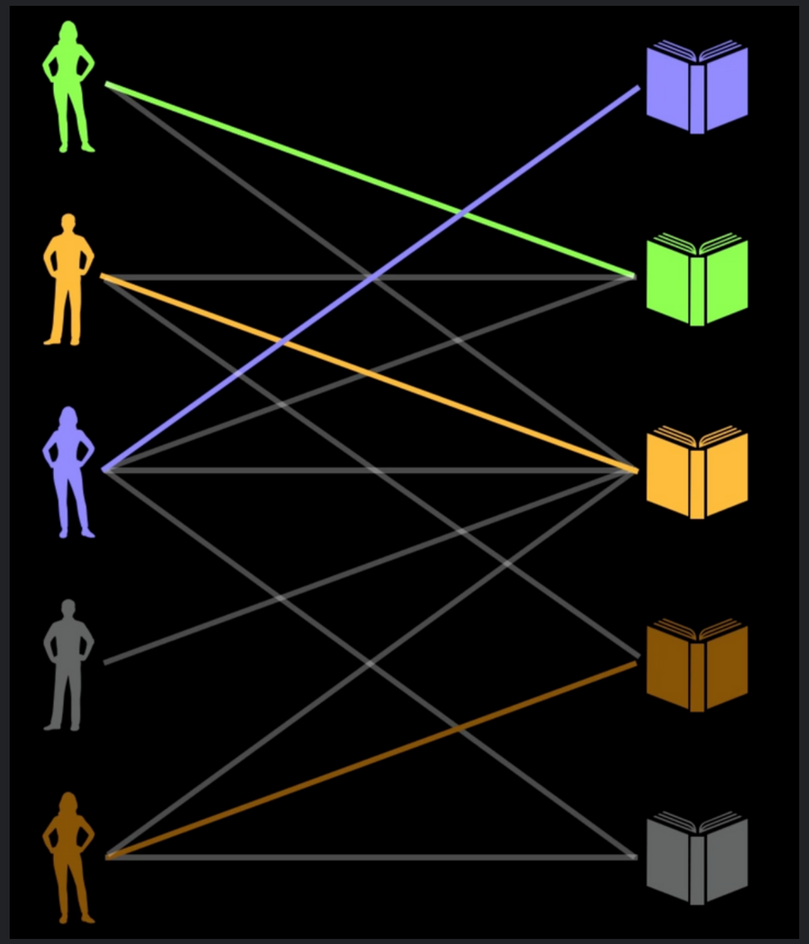

CNN
transformer encoder-decoder
no object (ø)
no object (ø)
set of image features
set of box predictions
bipartite matching loss

BIPARTITE MATCHING

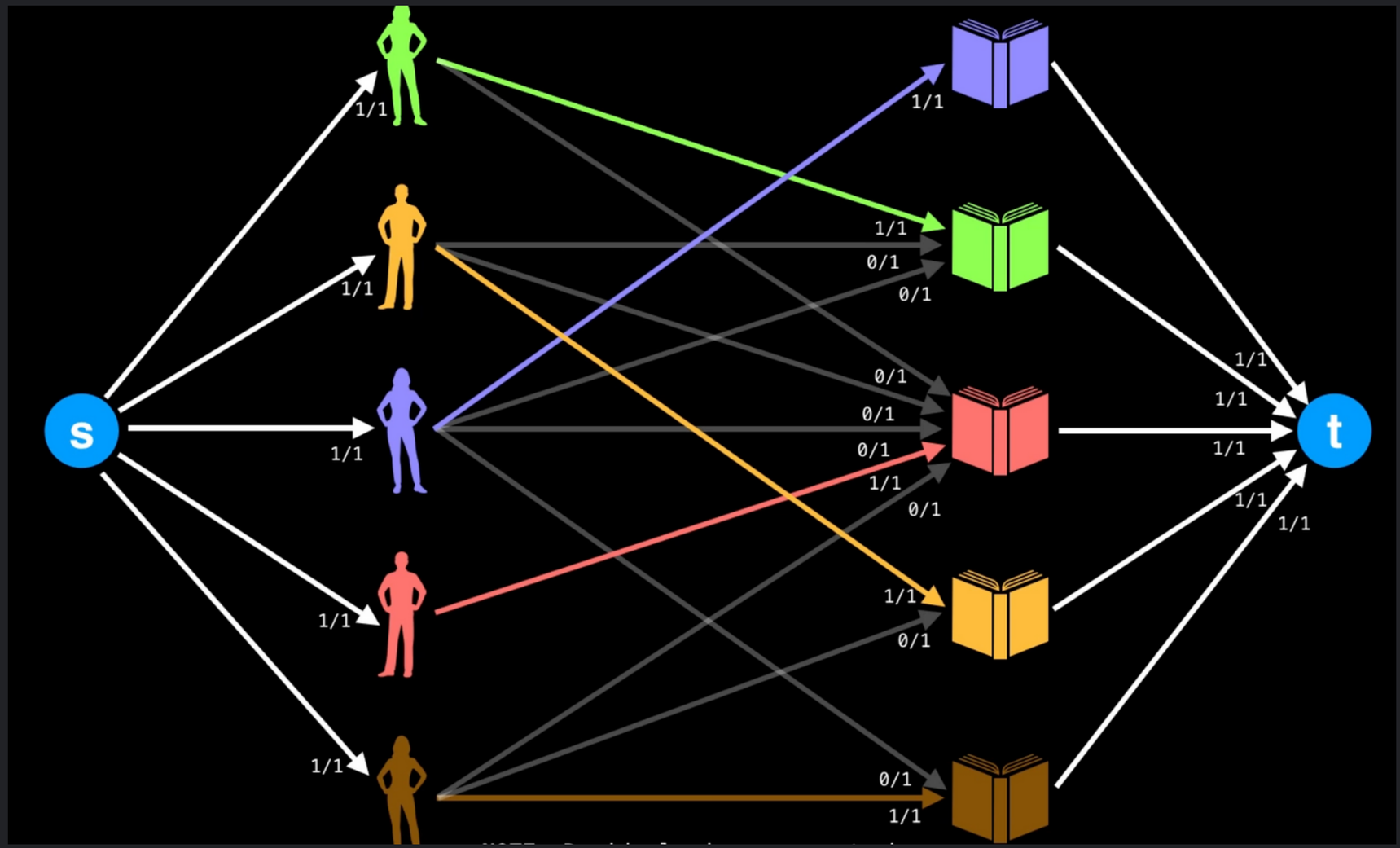
It's a subset of the edges for which every vertex belongs to exactly one of the edges. Our goal in this activity is to discover some criterion for when a bipartite graph has a matching.



Back to Agenda



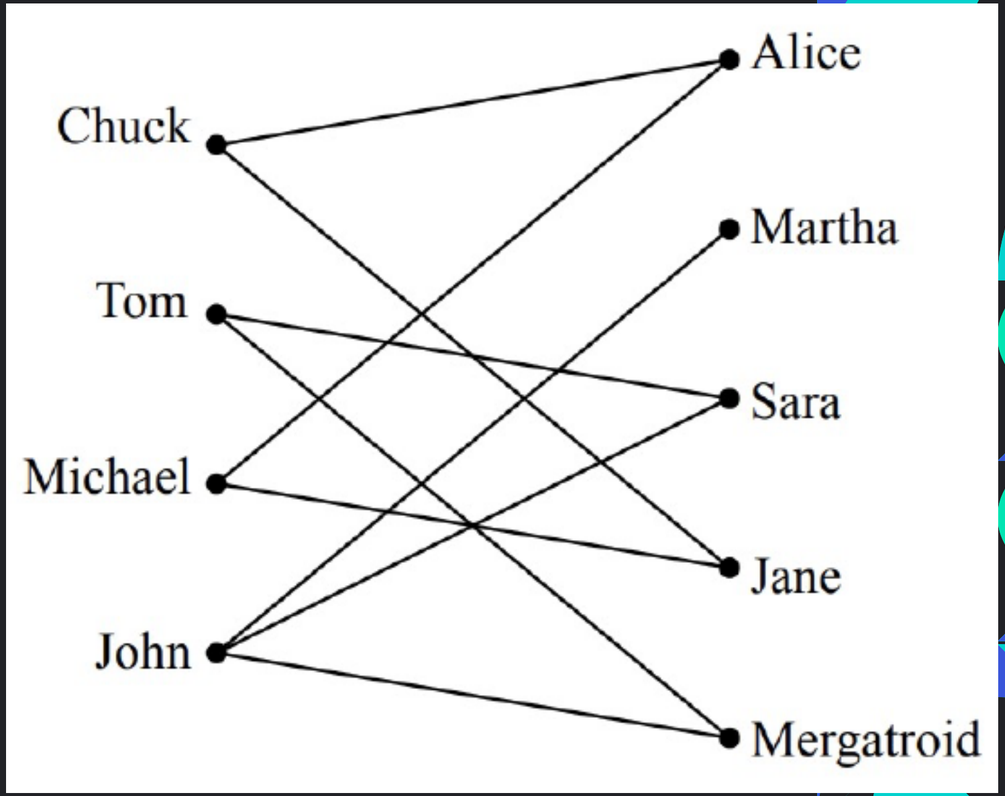
Maximum Bipartite Matching



BOTTLENECK

If there is a bottleneck, then no MACTH is possible! (Of Course)

When there are no bottleneck, then there is a MATCH! (Hall's Theorem)



Back to Agenda

VIRTUAL REALITY SUPREMACY

Honestly, I think VR tech's potential is limitless.

SIMPLICITY

DETR is simple compared to other methods. It skips many of these steps and directly predicts a set of things it thinks are in the picture.

ACCURACY

DETR is as good as other methods in recognizing objects in pictures. It doesn't get confused and predicts each thing only once.

COMPARISON WITH OTHER METHODS

DETR is compared with a popular method called Faster R-CNN, and they found that DETR is just as good, especially with big objects.

PANOPTIC SEGMENTATION

DETR isn't just about finding objects; it can also be used for something called panoptic segmentation, a more advanced way of understanding pictures.

PROBLEMS

Detect the number of car in accident

WHY IT IS IMPORTANT?

DETECTION CHALLENGE

Overlapped object detection is still a challenge. In the complex context, the accuracy of object detection will also become worse

TIMELY RESPONSE

Quick detection of car accidents allows for faster response times from emergency services.

DATA ANALYSIS

By collecting and analyzing data on car accidents, patterns and trends can be identified.

AI-POWERED SURVEILLANCE

AI and object detection technologies can be employed for continuous surveillance of roadways

DATASET

VEHICLE CRASH DATASET COMPUTER VISION PROJECT--FROM ROBOFLOW

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="T7QMxFcBVQ0rtXJspyh0")
project = rf.workspace("object-detection-3iugc").project("vehicle-crash-dataset")
dataset = project.version(1).download("coco")
```

VEHICLE CRASH DATASET COMPUTER VISION PROJECT--FROM ROBOFLOW

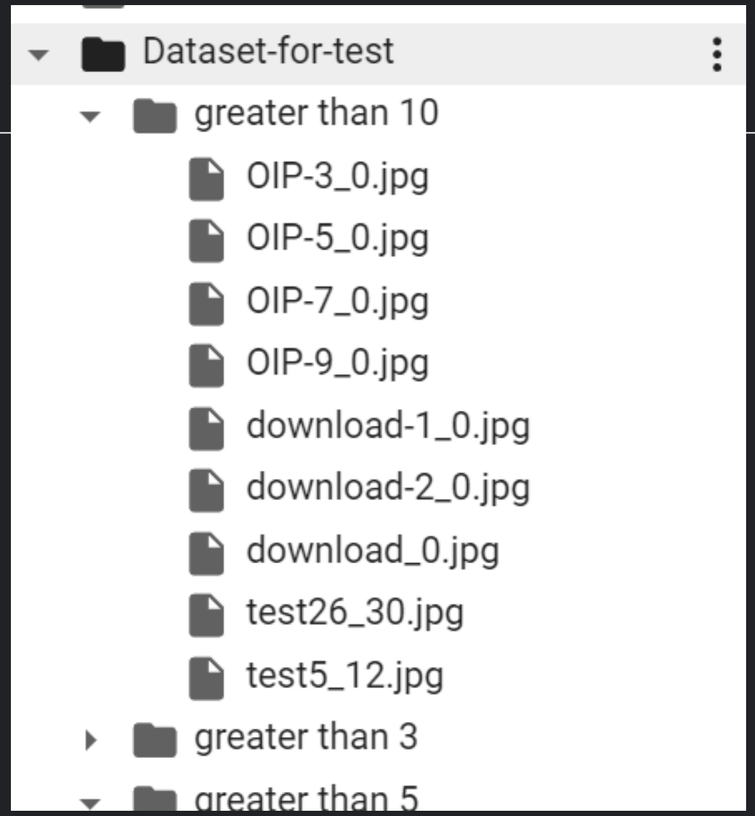
```
import pathlib
import requests
from transformers import DetrImageProcessor, DetrForObjectDetection
import torch
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.patches as patches

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

data_dir = pathlib.Path("Vehicle-Crash-Dataset-1/train/")
image_count = len(list(data_dir.glob('*_jpg.rf*.jpg')))
print(f"Number of jpg files: {image_count}")
```

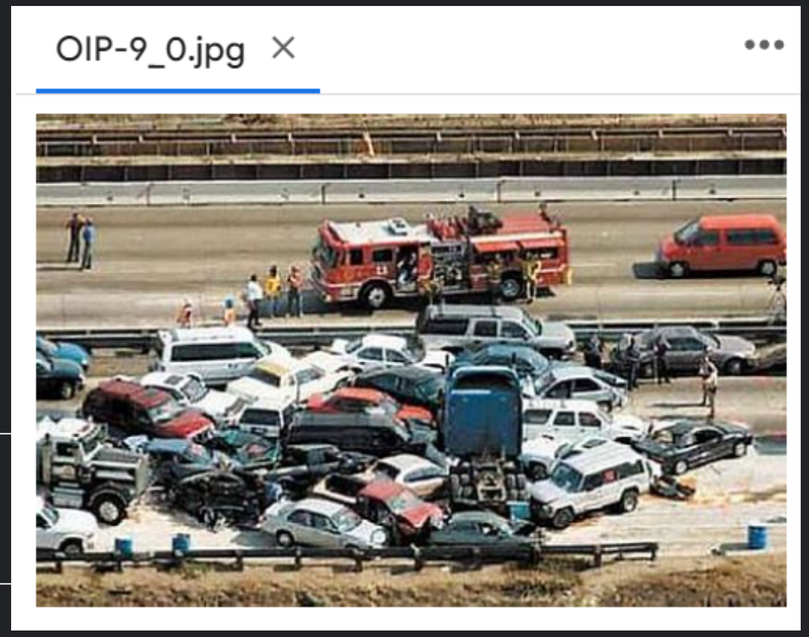
```
Number of jpg files: 471
```

```
from PIL import Image
car_graphs = list(data_dir.glob('*_jpg.rf*.jpg'))
Image.open(str(car_graphs[23]))
```

DATASET

IMAGES FROM INTERNET





DATA ANALYSIS

PROCESS WITH PRETRAINED MODEL

```
# Use pre-trained DETR image processors and object detection models
# Explanation of no_timm: timm is a library for implementing various visual models.
# a particular version of the timm library conflicts or causes problems with other libraries
# By using revision="no_timm", the loaded version of the model does not depend on the timm library, thus avoiding potential problems.
processor = DetrImageProcessor.from_pretrained("facebook/detr-resnet-50", revision="no_timm")
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50", revision="no_timm")

# Process the images and pass the result to model
inputs = processor(images=images, return_tensors="pt")
outputs = model(**inputs)
```

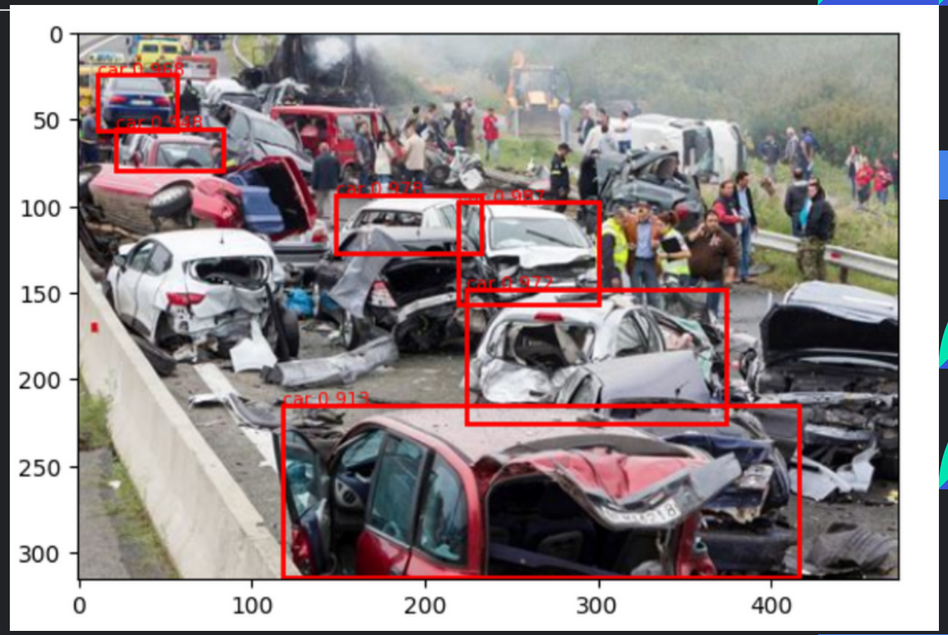
```
preprocessor_config.json: 100% 401/401 [00:00<00:00, 18.5kB/s]
config.json: 100% 6.60k/6.60k [00:00<00:00, 330kB/s]
pytorch_model.bin: 100% 167M/167M [00:03<00:00, 52.6MB/s]
```

```
for score, label, box in zip(results["scores"], results["labels"], results["boxes"]):
        box = [round(i, 2) for i in box.tolist()]
        print(
                f"Detected {model.config.id2label[label.item()]} with confidence "
                f"{round(score.item(), 3)} at location {box}"
        )
```

DATA ANALYSIS

VISUALIZATION AND GET THE RESULT

```
# Extract bounding boxes from the predictions
for score, label, box in zip(results["scores"], results["labels"], results["boxes"]):
        box = [round(i, 2) for i in box.tolist()]

        # Check if the label is 'car'
        if model.config.id2label[label.item()] == 'car':
                # Create a Rectangle patch
                rect = patches.Rectangle(
                        (box[0], box[1]), box[2] - box[0], box[3] - box[1],
                        linewidth=2, edgecolor='r', facecolor='none'
                )

                # Add the patch to the Axes
                ax.add_patch(rect)

                # Print label and confidence score
                text = f"car {round(score.item(), 3)}"
                ax.text(box[0], box[1], text, color='r', fontsize=8)

plt.show()
```



REFERENCE

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. Computer Vision – ECCV 2020 (pp. 213–229). https://doi.org/10.1007/978-3-030-58452-8_13

THANK YOU FOR LISTENING!

Q & A