Deep Learning Project

Video Object Tracking

Demo

Content

- Basic Concepts in Multi-Object Tracking & Segmentation (MOTS)
- Assignment Overview
- Data Preparation / Augmentation
- Object Detection
- Tracker / Similarity Model Training
- Evaluation & Submission

MOTS Application

- **Autonomous driving**: Tracking pedestrians, vehicles, and obstacles
- **Sports analytics**: Following players and ball movement
- **Surveillance systems**: Monitoring people and objects in security footage
- **Robotics**: Tracking objects for manipulation or navigation
- **Retail analytics**: Tracking customer movement in stores
- **Wildlife monitoring**: Following animal movements in their habitats
- **Traffic management**: Monitoring vehicle flow and pedestrian movement

Online vs offline tracking

Online tracking

- Processes two frames at a time
- For real-time applications
- Hard to recover from errors or occlusions

Offline tracking

- Processes a batch of frames
- Good to recover from occlusions (short ones as we will see)
- Not suitable for real-time applications
- Suitable for video analysis

Off-line Tracking

- Detect objects across frames in a video
- Create a gallery for Re-ID
- Identify a single object and assign it with the same id across frames
- Output frames with bounding boxes, alongside the id

Assignment Overview

Goal:

Parse and prepare data.

Tune the pre-trained model using the training data. (4-6G VRAM, colab)

Implement the tracker and use it with the model.

Generate videos with boxes and ids for testing videos.

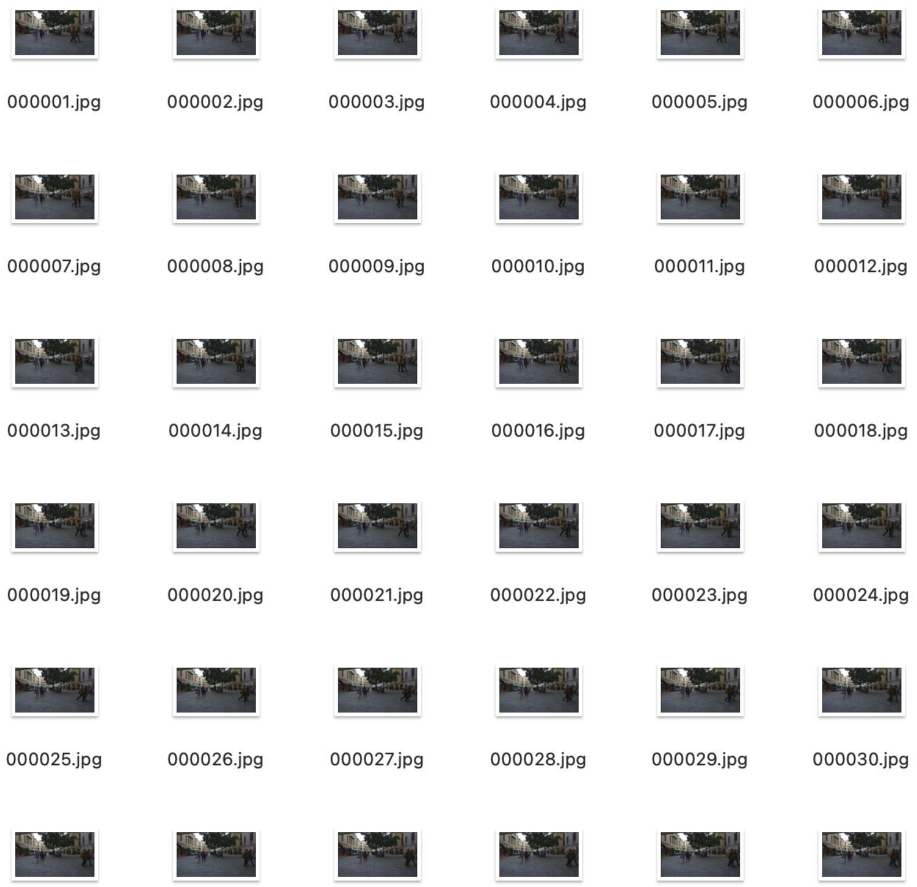
Data Preparation

Training dataset (MOTS):

1,3,586,447,85,263,1,1,1

Which means: (https://github.com/khalidw/MOT16_Annotator)

time frame 1
object id 3
bb_left 586
bb_top 447
bb_width 85
bb_height 263



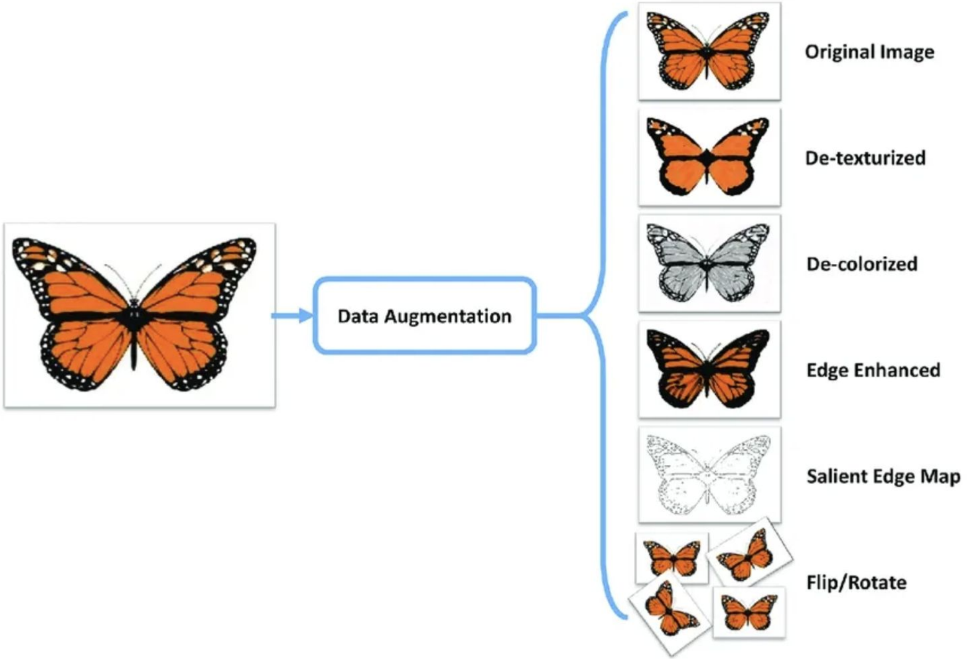
Data Preparation

Parse the ground truth:

```
import numpy as np


def parse_gt_file(file_path):
    data = []
    with open(file_path, 'r') as f:
        pass # do the preprocessing here
    return data
```

Data Augmentation

- Data augmentation is the process of artificially generating new data from existing data, primarily to train new machine learning (ML) models.
- Data augmentation artificially increases the dataset by making small changes to the original data.



Data Augmentation

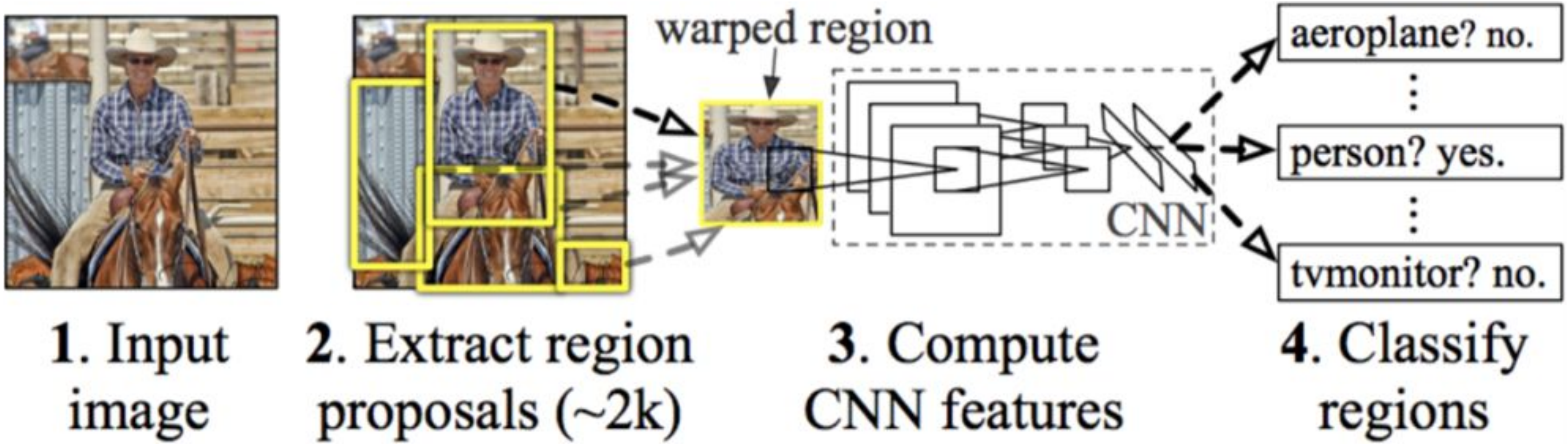
```
from torchvision import transforms

color_aug = transforms.Compose([
    transforms.ColorJitter(brightness=0.5, contrast=0.5, saturation=0.5),
    transforms.GaussianBlur(kernel_size=(5, 9), sigma=(0.1, 5)),
])

augmented_image = color_aug(original_image)
```

Object Detection Models

Since it requires enormous training data and computational resources to fully train a model from scratch to perform well on MOTS task, in this assignment, you are highly encouraged to load pre-trained models from `torchvision` and fine-tune it on the provided training dataset.

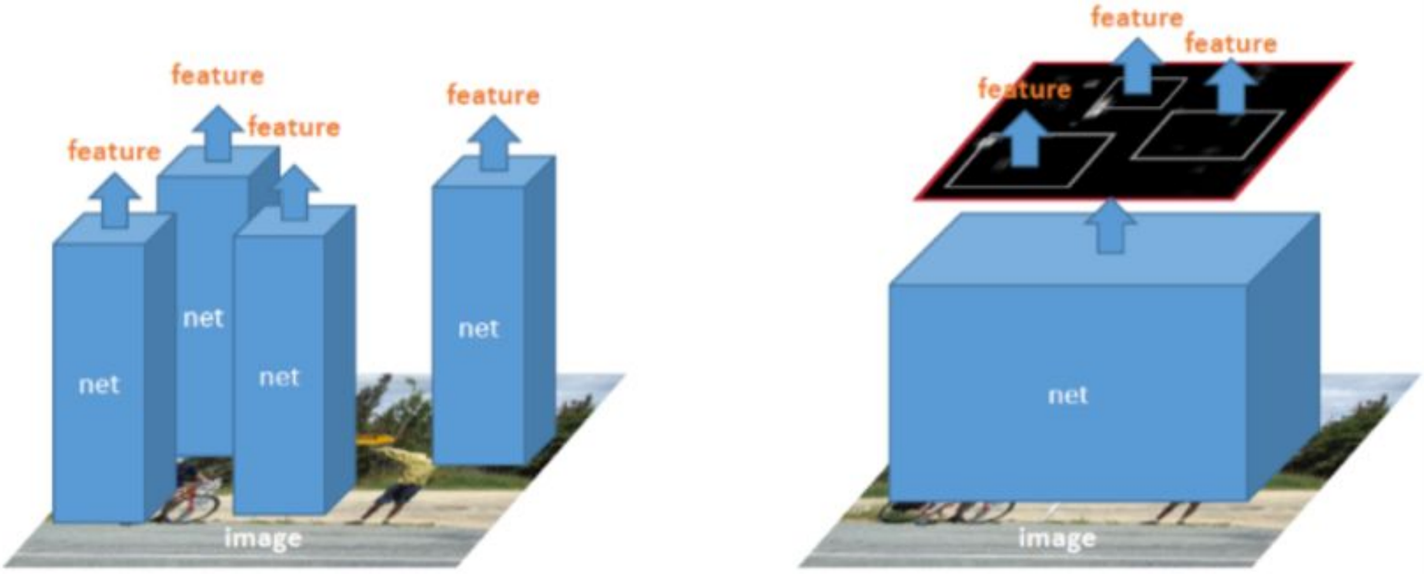
- **Fast R-CNN / Faster R-CNN**: Used for object detection (bounding box regression and classification)
- **Mask R-CNN**: Used for instance segmentation (pixel-wise mask prediction)

R-CNN



Girschick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

SPP-net

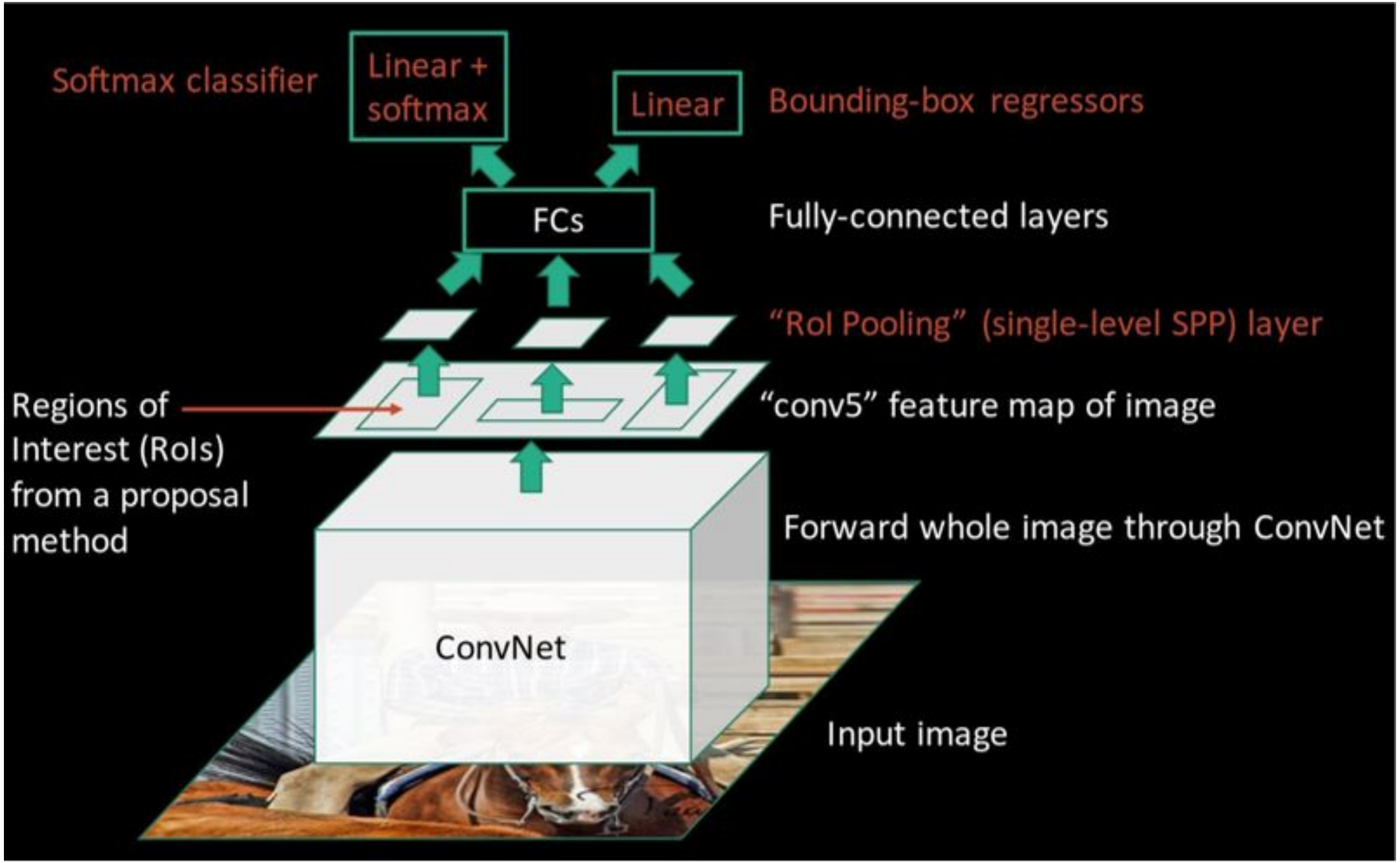


R-CNN
2000 nets on image regions

SPP-net
1 net on full image

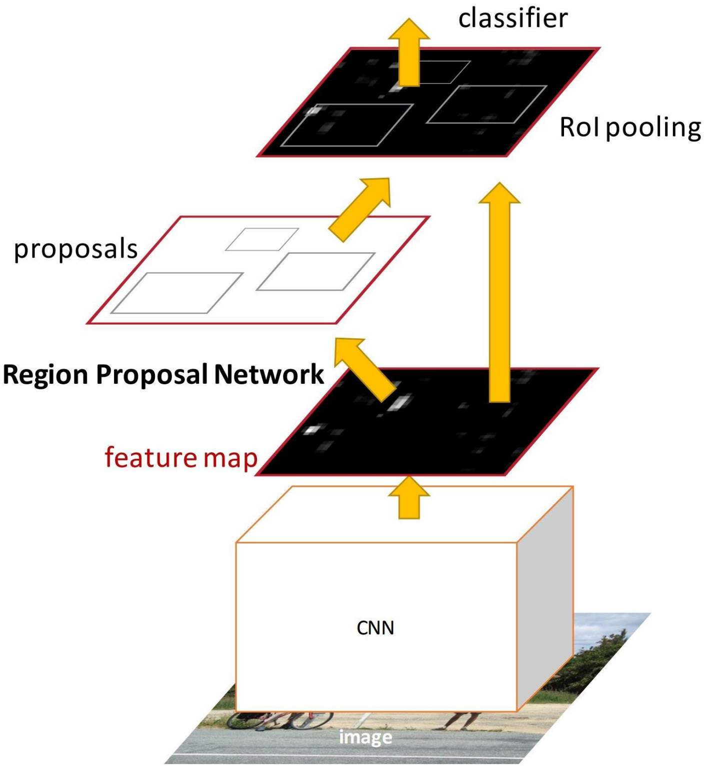
He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. ECCV 2014.

Fast R-CNN



Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Faster R-CNN



- Have the proposal generation integrated with the rest of the pipeline
- Region Proposal Network (RPN) trained to produce region proposals directly.
-

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Comparison

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9

Mask R-CNN (optional, pixel-level seg)

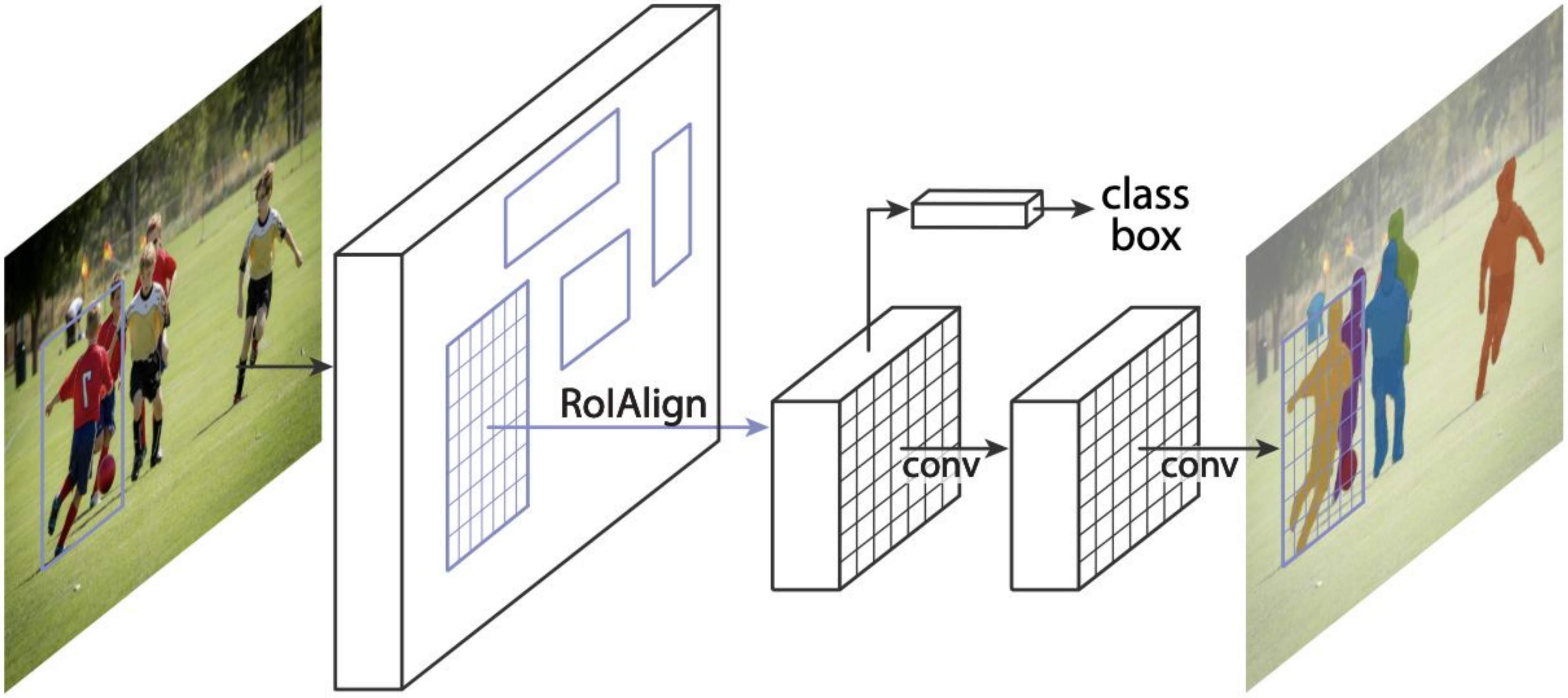


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Faster R-CNN

Since we will have limited amount of data to fully train a model from scratch, it's a good approach to do few-shot fine-tuning which is a form of transfer learning.

- Transfer learning is a machine learning technique in which knowledge gained through one task or dataset is used to improve model performance on another related task and/or different dataset.

- Few-shot fine-tuning is to only tune a small portion of overall parameters in a large model.

Faster R-CNN

Since we will have limited amount of data to fully train a model from scratch, it's a good approach to do few-shot fine-tuning which is a form of transfer learning.

- Transfer learning is a machine learning technique in which knowledge gained through one task or dataset is used to improve model performance on another related task and/or different dataset.

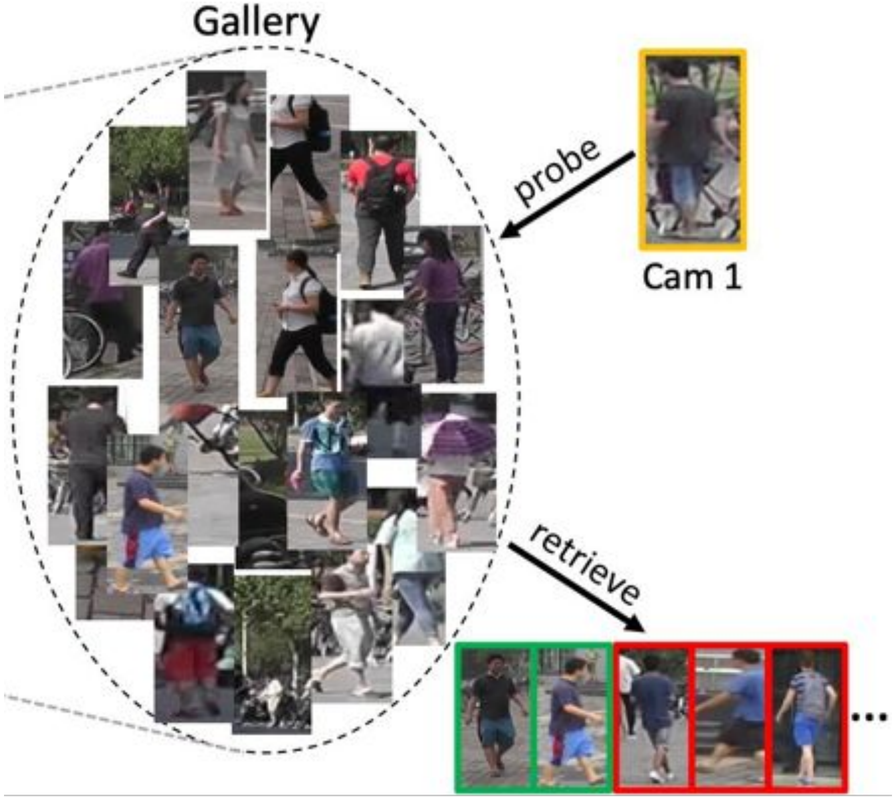
- Few-shot fine-tuning is to only tune a small portion of overall parameters in a large model.

- Together with the data augmentation, we can simply tune a pre-trained model on our own small training dataset.

Fine-tune a pre-trained Faster R-CNN

```
import torchvision
from torchvision.models.detection import fasterrcnn_resnet50_fpn
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor

# Freeze backbone layers
for param in model.backbone.parameters():
    param.requires_grad = False

# Only fine-tune the heads for classification and mask prediction
params_to_optimize = [p for p in model.parameters() if p.requires_grad]
```

Re-ID

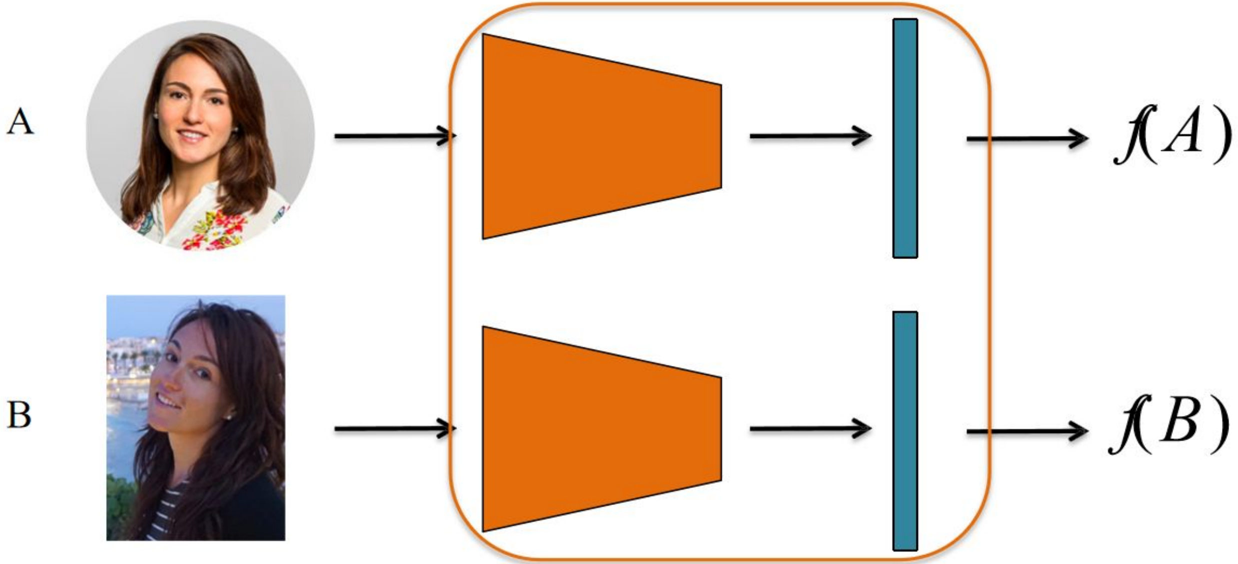


Basic idea: train a model to identify all detected objects.

Cons: We have to fine-tune the model once new objects join or detected object leave

Tracker (Re-ID)

- Siamese network =shared weights

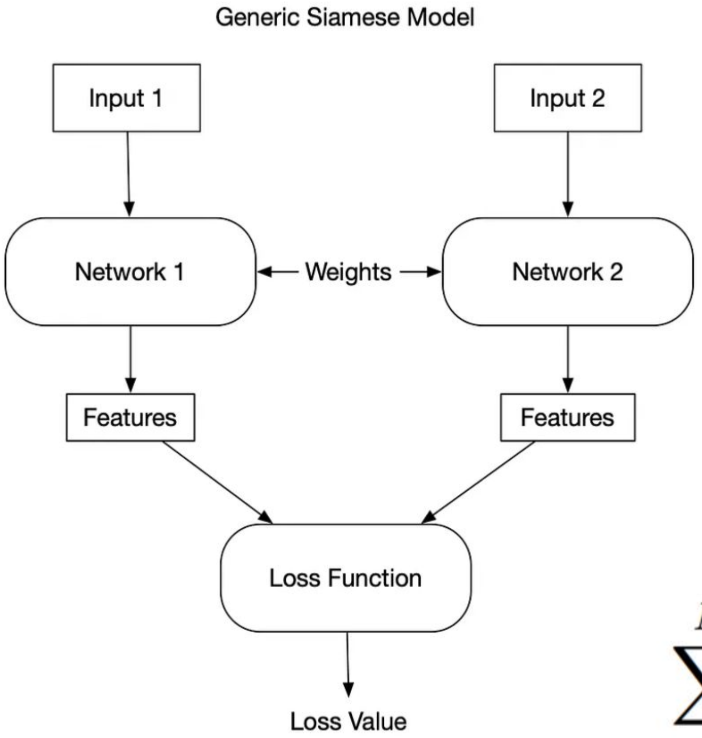


Taigman et al. „DeepFace: closing the gap to human level performance“. CVPR 2014

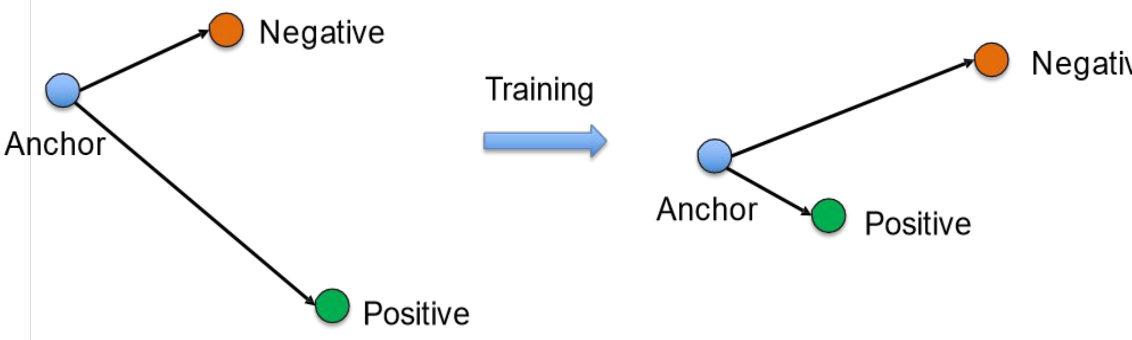
We do not need to identify any face, our model is trained to just tell the difference between them

Otherwise, we have to fine-tune the model frequently whenever there is new one here

Tracker (Similarity Model)



Triplet loss



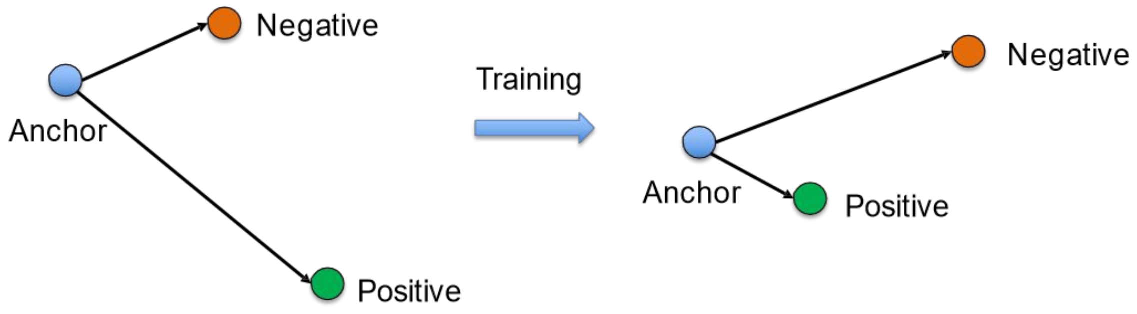
$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]$$

Tracker (Similarity Model)

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Siamese_Network(nn.Module):
    def __init__(self):
        super(Siamese_Network, self).__init__()

        # CNN layers for feature extraction
        self.conv1 = nn.Conv2d(1, 64, kernel_size=3)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3)
        self.conv3 = nn.Conv2d(128, 128, kernel_size=3)
        self.fc1 = nn.Linear(128 * 22 * 22, 256)
        self.fc2 = nn.Linear(256, 256)

def forward_one(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2(x), 2))
        x = F.relu(self.conv3(x))
        x = x.view(-1, 128 * 22 * 22)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

    def forward(self, input1, input2):
        output1 = self.forward_one(input1)
        output2 = self.forward_one(input2)
        return output1, output2
```

Tracker (Similarity Model)

Triplet loss



Market-1501 Dataset

Implementation Pipeline

Data processing(decode gt (rle)) &
Data augmentation

↓

Faster R-CNN few-shot fine-tuning

↓

Train the Siamese Network for Re-ID

↓

Perform Detection and Re-ID for tracking

Faster R-CNN
Object Detection and bounding box segmentation

Re-ID
Tracking objects across frames

Evaluation

Present your design, implementation details, challenges encountered and a result video in class.

Rubric:

- A: Perfect tracking and bounding box across frames without missing the target.
- B: Can track an object across a majority of frames but not all.
- C: At least can detect the targeted object and assign a bounding box to it across some frames.

Submission

- All of your source code (do not include any model weights)
- A tracking video of at least one single target pedestrian on one MOT16 test data (video or link)
- A pdf report that contains the following sections (recommend using Latex):
    - Abstract, Introduction
    - Methodology (System Design, Implementation Details)
    - Challenges encountered (how you solved them)
    - Self-evaluation and Proper Citations

Thank you!

This page is for guiding students who have even no idea about the implementation

Train the similarity network (input: raw images 16 * 16) -> output: 0, 1

Fine-tune the Faster RCNN (input: raw images) -> output: bbox of each object, class id ..

Inference of similarity network (24 * 24 images + gt) -> (1 , 0)

Get the bbox position from RCNN, retrieve images (30 * 16) (representing single object)

Resize the retrieved image for each object as the size of the image that you use to train the similarity network