Master in
Computer Vision
Barcelona

Project
Module 6
Coordination

Video Surveillance for Road Traffic Monitoring

J. Ruiz-Hidalgo / X. Giró

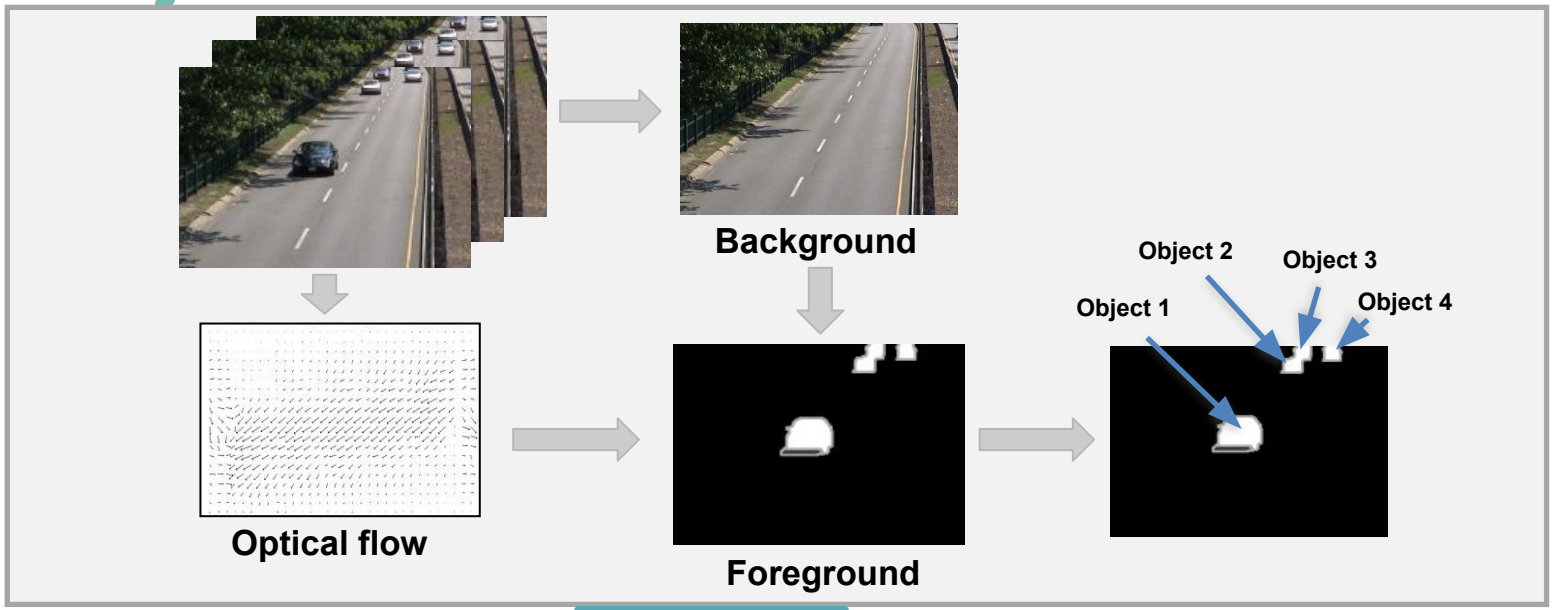
j.ruiz@upc.edu / xavier.giro@upc.edu

Week 3: Report

Teams & Repos (please state any change)

Repo	Students
Team 1	Pau Torras, Daniel Rodríguez, Marcos Melgar
Team 2	David Serrano, Igor Ugarte, Joan Antonio Rodríguez, Francesc Net
Team 3	Adam Szummer, Sergio Montoya, Laia Albors, Ibrar Malik
Team 4	Mert Yazan, Berkay Arpacı, Marcos Conde
Team 5	Brian Guang Jun Du, Eric Henriksson Martí, Ignacio Galve Ceamanos, Josep López
Team 6	Kevin Martín, Eudald Ballesca, Marcelo Sanchez, Sergi Vidal

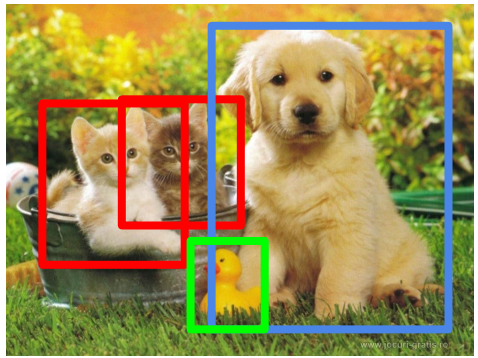
Project Schedule



Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
• Introduction • DB • Evaluation metrics	• Background estimation • Stauffer & Grimson	• Object Detection • Tracking	• Optical flow • Tracking	• Multiple cameras • Speed	• Presentation workshop

Tasks

- Task 1: Object detection
  - **Task 1.1: Off-the-shelf**
  - Task 1.2: Fine-tune to your data
- Task 2: Object tracking



Task 1.1: Object Detection: Off-the-Shelf (Team X)

Copy & paste (max 1 page)

Task 1.1: Object Detection: Off-the-Shelf (Team 1)

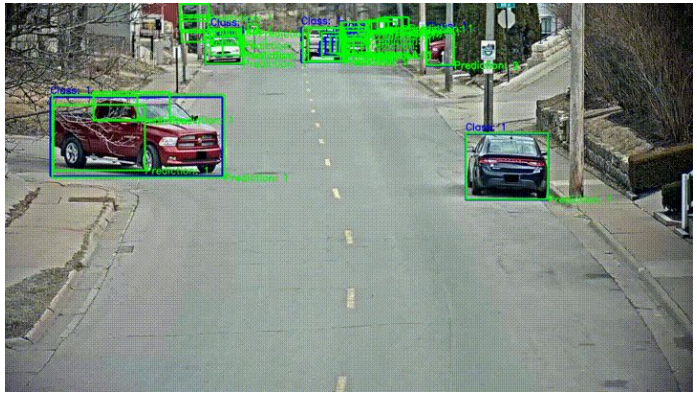
COCO classes don't fully overlap with the annotations in the dataset: The moving pair person + bike is seen as two classes → **Low results in bike**

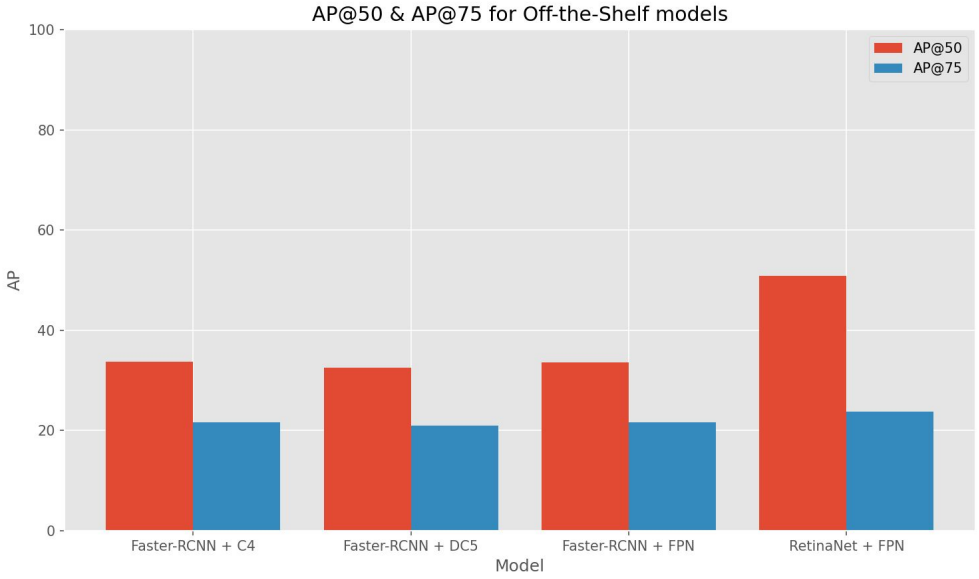
Results are coherent but lack curation (many false positives).

Prediction

Ground Truth

Retina prediction





ResNet-50 backbone	RetinaNet + FPN	Faster-RCNN + FPN	Faster-RCNN + DC5	Faster-RCNN + C4
AP50	50.85	33.54	32.50	33.71
AP75	23.72	21.62	20.96	21.58
AP-bike	8.87	0.25	0.06	0.40
AP-car	45.27	42.57	41.31	42.94

Task 1.1: Object Detection: Off-the-Shelf (Team 2) (1/2)

Results

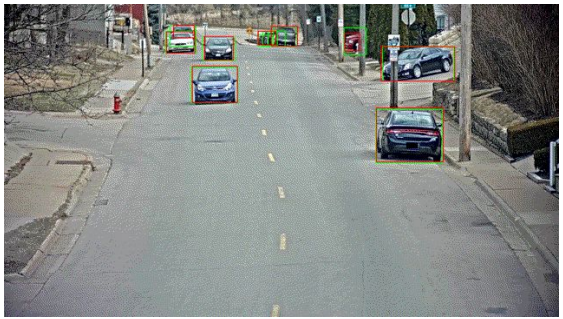
Model	Implementation	$AP_{0.5car}$
RetinaNet (R101)	Detectron2	0.7269
Faster R-CNN (X101-FPN)	Detectron2	0.4855
Mask R-CNN (X101-FPN)	Detectron2	0.4952
SSD 512	w1 ground truths	0.3631
Yolo v3	w1 ground truths	0.4428

***note: We only use car detections for all tasks (moving or parked cars)**

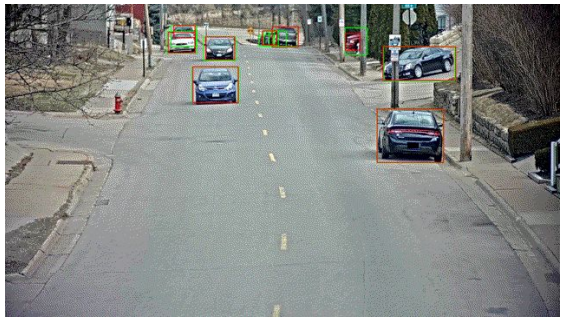
Task 1.1: Object Detection: Off-the-Shelf (Team 2) (2/2)

Qualitative results

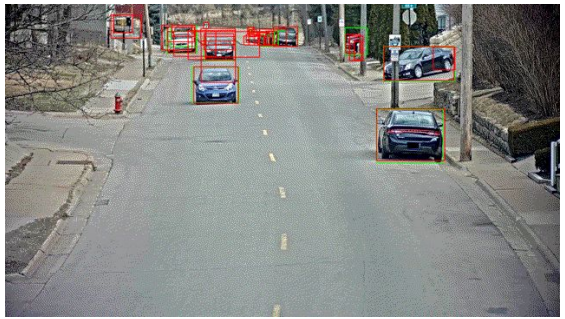
Mask R-CNN (X101-FPN)



Faster R-CNN (X101-FPN)



RetinaNet (R101)



Ground truth Detections

Quantitative results:

- RetinaNet gives us the best results due to the accurate detection in the parked cars. However, if we take a look on the quantitative results have some strange false detections.

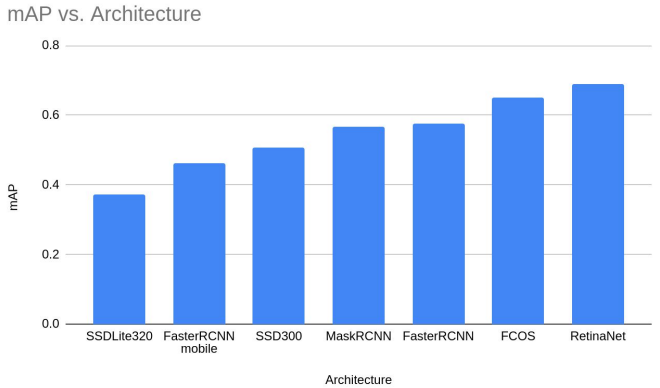
Task 1,1: Object Detection: Off-the-Shelf (Team 3)

Copy & paste (max 1 page)

Architecture	Backbone	mAP
SSDLite320	MobileNetV3	0,3722
FasterRCNN mobile	MobileNetV3	0,4622
SSD300	VGG16	0,5082
MaskRCNN	ResNet50	0,5667
FasterRCNN	ResNet50	0,5746
FCOS	ResNet50	0,6500
RetinaNet	Resnet50	0,6907

Everything was implemented using native torchvision implementation with pretrained network on COCO dataset, The detailed list of architectures can be found here,

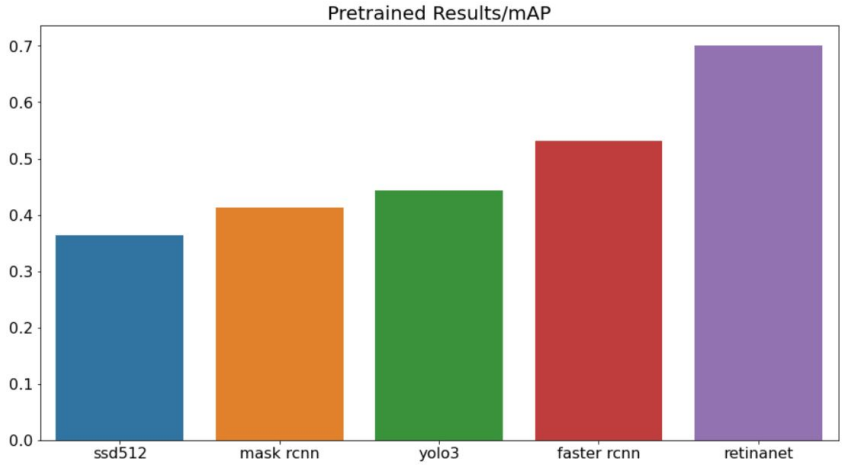
We can see that off-the-shelf ResNet50 backbones are better in mAP with Single Stage Detector RetinaNet performing best, In the analysis of task 1,2, we can see how this changes after fine-tuning,

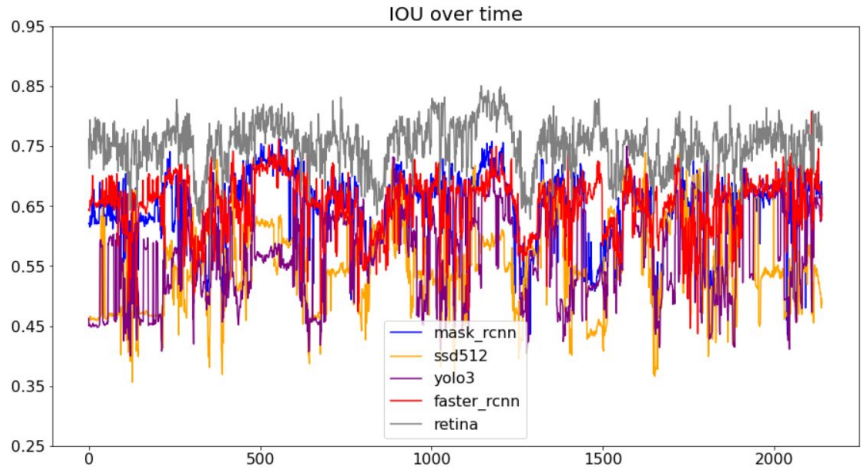


Task 1,1: Object Detection: Off-the-Shelf (Team 4)

- We used detectron2 to try out RetinaNET and FasterRCNN, They both outperform the other networks, and a pretrained RetinaNET is able to achieve %70,1 accuracy,
- IOU results are more close but RetinaNET and FasterRCNN outperforms in IOU too,
- A more fair approach would be to compare each network on a detectron2 setting, because their training methods can be different,

Network	mAP	mIOU
RetinaNET	**%70,1**	**%75,1**
Faster RCNN	%53,1	%65,2
YOLO3	%44,3	%56,8
Mask RCNN	%41,2	%64,7
SSD 512	%36,3	%55,1



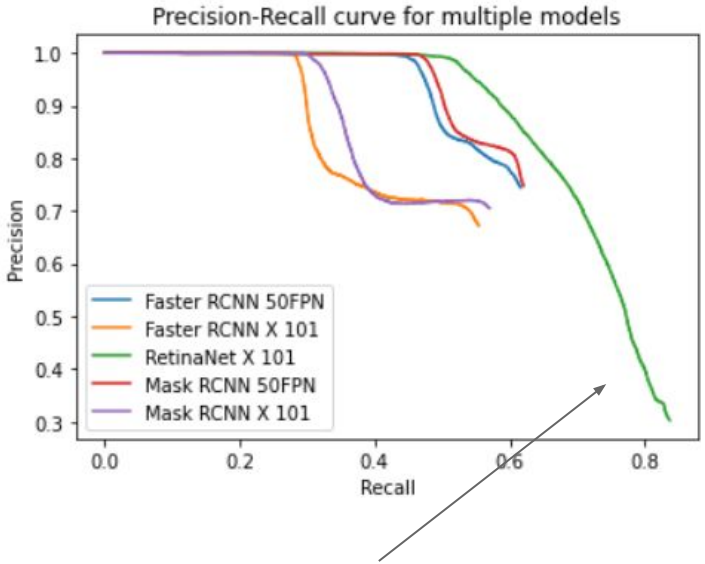


Task 1.1: Object Detection: Off-the-Shelf (Team 5)

Several models pre-trained on the *COCO* dataset from *Detectron2* have been evaluated in terms of object detection on the image sequence at hand. The class “car” included in the *COCO* dataset has been considered for this purpose.

Model	$AP_{0.5}$	Execution Time
Faster RCNN - 50 FPN	0.603	172.89
Faster RCNN - X101-FPN	0.485	259.27
RetinaNet - R101	0.727	211.25
Mask R-CNN - 50 FPN	0.612	193.18
Mask R-CNN - X101-FPN	0.495	244.03

RetinaNet - R101 displays the best performance in terms of AP, while Faster R-CNN - 50 FPN is the fastest in inference time.



RetinaNet has very high recall but at the expense of lower precision. This is more evident later on with the presents of FPs.

Task 1.1: Object Detection: Off-the-Shelf (Team 6) (½)

Model Architecture	$AP_{0.5}$	IoU	Inference Time (s)
RetinaNet (retinanet_R_101_FPN_3x)	0.724	0.794	200.31
RetinaNet (retinanet_R_50_FPN_3x)	0.567	0.612	172.43
Faster R-CNN (faster_rcnn_X_101_32x8d_FPN_3x)	0.543	0.591	220.43
Faster R-CNN (faster_rcnn_R_50_FPN_3x)	0.487	0.56	165.34
Mask R-CNN (mask_rcnn_R_101_FPN_3x)	0.501	0.543	215.23

Retina 101 outperforms in **accuracy** (AP and mIou) however, Faster **R-CNN 50 FPN** is **fastest**

Task 1.1: Object Detection: Off-the-Shelf (Xavi)

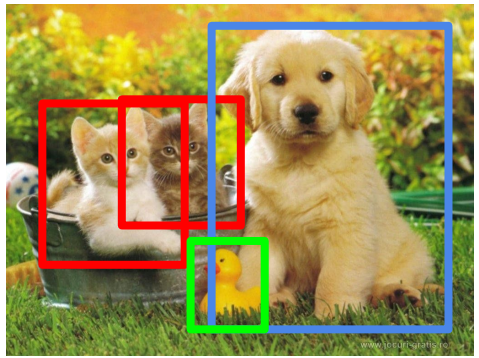
Repo	Feedback
Team 1	4 x object detectors with a Reset-50 backbone. RetinaNet + FPN is the best. + Study of the casecase of person + bike - The used implmentation is not referred. - Missing evaluation in terms of computational requirements
Team 3	7 x object detectors from torchvision. Best v with RetinaNet. + Provides de backbone for a more fair comparison. Missing evaluation in terms of computational requirements
Team 5	5 x object detectors. Best accuracy with RetinaNet. + Provides computation time. + Provides de backbone for a more fair comparison + Precision vs Recall curve.

Task 1.1: Object Detection: Off-the-Shelf (Javi)

Repo	Feedback
Team 2	3x detectors from detectron2. RetinaNET best results References not included. Info on classes evaluated. No evaluation of computational requirements. Qualitative and quantitative results shown. Bonus showing the difference between qualitative and quantitative results for retinanet.
Team 4	2x detectors from detectron2. RetinaNET best results References not included. No info on classes evaluated. No evaluation of computational requirements. Qualitative results shown. Table the same as bars? No quantitative results shown.
Team 6	5x detectors from detectron2. RetinaNET best results No details on the implementation. Which one? References not included. No info on classes evaluated. Evaluation of computational requirements. Qualitative results shown. No quantitative results shown. No discussion.

Tasks

- Task 1: Object detection
  - Task 1,1: Off-the-shelf
  - **Task 1,2: Fine-tune to your data**
- Task 2: Object tracking



Task 1,2: Object Detection: Fine-tune (Team X)

- Copy & paste (max 2, pages)

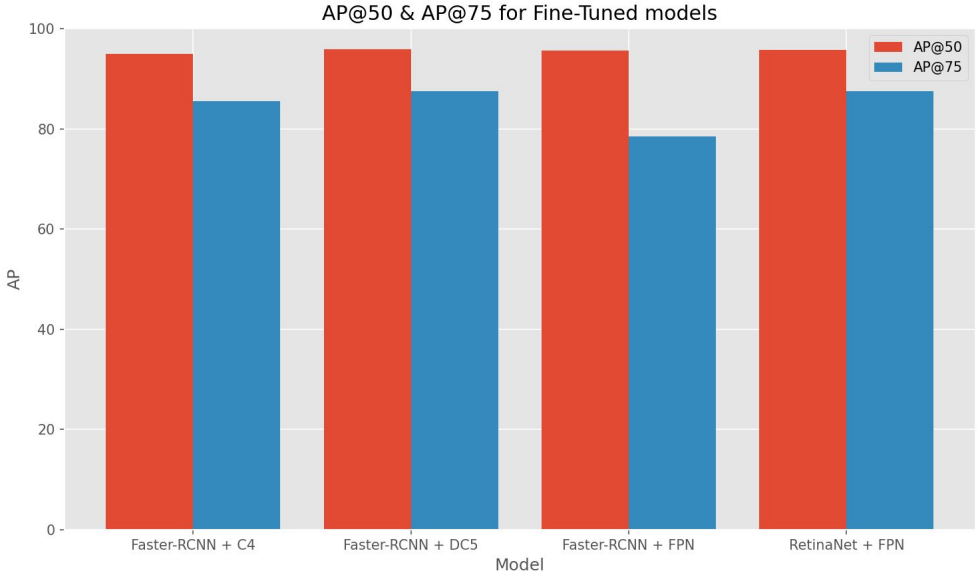
Task 1.2: Object Detection: Fine-tune (Team 1 - 1/2)

Models fine-tuned with holdout 25-75 train-test partitions.

Frames 0-535 contain a cyclist that runs down the road →Many examples of bike class; sensible improvement in it.

Rogue classes should not appear anymore.

We initially chose not to use heavy data augmentation (default Detectron2 settings) as the test partition is known to be closely aligned to the training partition. We tested adding them with comparable results, in both cases with highly overconfident predictions.



	RetinaNet + FPN	Faster-RCNN + FPN	Faster-RCNN + DC5	Faster-RCNN + C4
AP50	95.81	95.63	95.87	94.96
AP75	87.46	78.50	87.57	85.45
AP-bike	71.38	58.18	64.74	61.54
AP-car	86.55	87.36	84.68	86.21

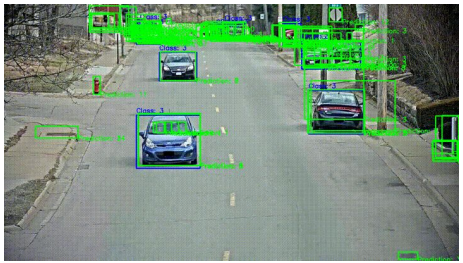
Task 1.2: Object Detection: Fine-tune (Team 1 - 2/2)

Above: Off the shelf
Below: Fine-tuned

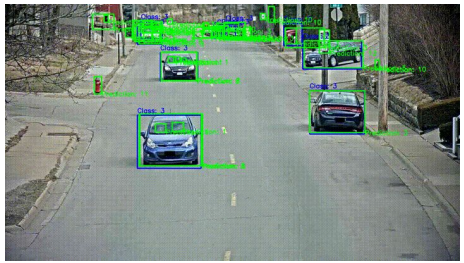
Prediction

Ground Truth

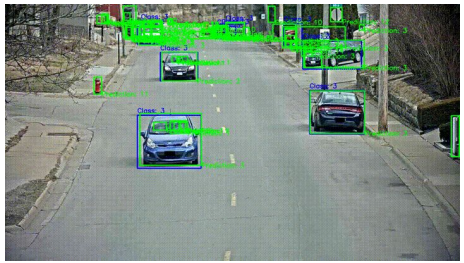
Qualitatively, off the shelf models are quite good at detecting the required objects (plus extra boxes that are ignored in evaluation but shown here for illustration). However, trained models are more stable. In particular, Faster RCNNs seem to have less false positives than Retina FPN, even if the latter is overall better.



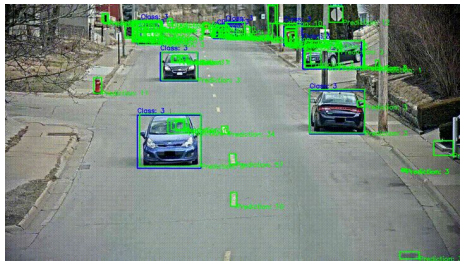
Retina FPN



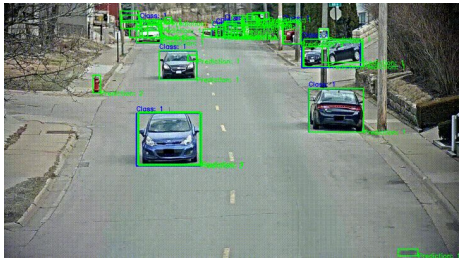
Faster RCNN + FPN

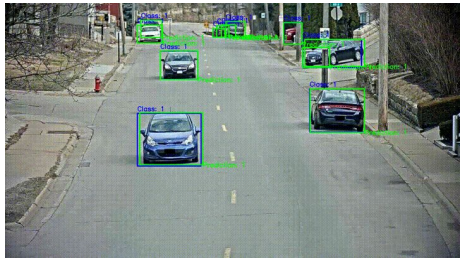


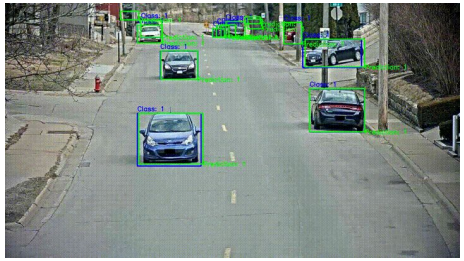
Faster RCNN + DC5

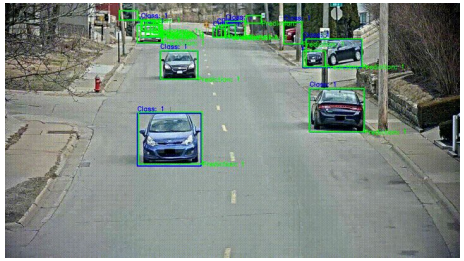


Faster RCNN + C4









Task 1.2: Object Detection: Fine-tune (Team 2) - (1/2)

Quantitative results:

- ❏ To improve the results from using the off-the-shelf models, we have fine-tuned them with the first 25% of the frames.
- ❏ To avoid the overfitting we have used data-augmentation.
- ❏ We have babysitted all the models as well as found their learning rated by grid-searching.

Hyperparameters:

- ❏ LR: 0.001
- ❏ Images per batch: 2
- ❏ Unfreezed layers: last 2

Data augmentation techniques:

- ❏ Horizontal Flip
- ❏ Random Crop

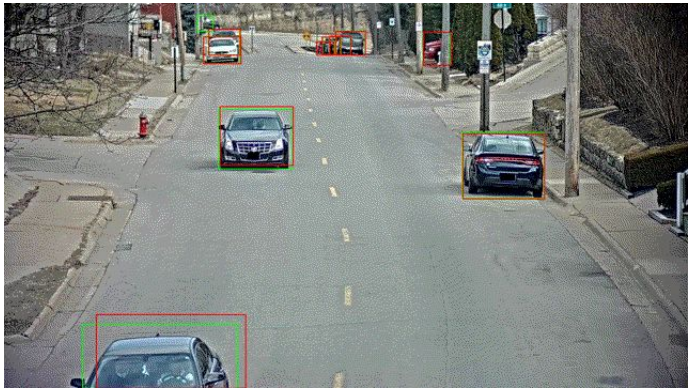
By fine-tuning the models we have obtained a mAP increase by 33.44% and 102.92% on RetinaNet and Faster R-CNN respectively. The value difference is mainly in the difference on the room for improvement of each algorithm since the results are quite similar.

Model	Off-the-shelf $AP_{0.5car}$	Fine-tune $AP_{0.5car}$	Data Aug fine-tune $AP_{0.5car}$	
RetinaNet (R101)	0.7269	0.9376	0.9700	+ 0.2431
Faster R-CNN (X101-FPN)	0.4855	0.9513	0.9852	+ 0.4997

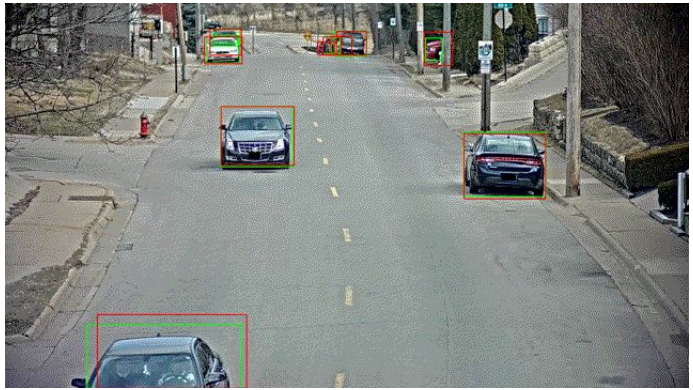
Task 1.2: Object Detection: Fine-tune (Team 2) - (2/2)

Qualitative results:

We compare the fine-tuned **Faster R-CNN** with both the ground truth and the off-the-shelf method.



Ground truth Fine-tuned

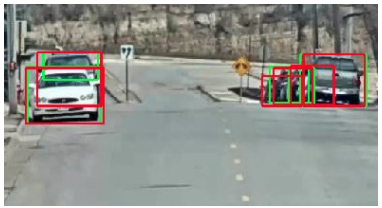


Off-the-shelf Fine-tuned

We can see the two main reasons why the algorithm improves so much:

1. The parked cars are learned after training and they are detected almost perfectly.
2. Pick-ups sometimes are detected as trucks in the off-the-shelf models.

1)



2)

Task 1,2: Object Detection: Fine-tune (Team 3)

For all network architectures after 1-3 epochs there was no change in mAP regardless if it was trained for longer, Networks stopped learning more, Similarly to augmentations, The problem seems very easy to learn and not reaching better mAP might be due to one reason not related to the ability to learn (explained in the next slide),

FasterRCNN

FasterRCNN mobile

MaskRCNN

RetinaNet

FCOS

SSD

SSDLite

Have been trained with SGD and following parameters:

- LR = 0,005
- momentum = 0,9
- weight_decay = 0,0005

mAP 0,90

Other LRs and approaches including augmentations have been tried but nothing has reached more than ~0,90,

Have been trained with SGD and following parameters:

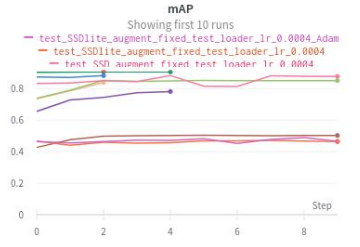
- LR = 0,00004
- momentum = 0,9
- weight_decay = 0,0005

Further changing LR and step size lead to slightly better results

mAP 0,85

mAP 0,88

mAP 0,50



Task 1,2: Object Detection: Fine-tune (Team 3)

From the previous slide, we can see that the chosen backbone greatly impacts the performance of models, All models with ResNet50 reach easily 90%, VGG16 80%, and MobileNet3 50% of mAP,

Qualitative analysis shows that:

- Networks are detecting well parked cars
- Passing cars are overlapping parked cars but groundtruth keeps hidden cars marked which are impossible to be detected
- Passing Truck is marked as car in ground truth while COCO dataset have separate class for trucks,
- Some cars stop being detected before they fully leave the scene

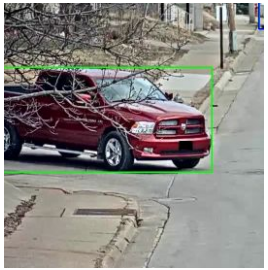
We believe that those mentioned cases are responsible for not achieving higher mAP values,

Parked cars are well detected

Randomness of detection or missing detections as passing cars occludes the parked cars

Truck is not detected, probably because coco has separate class for it,

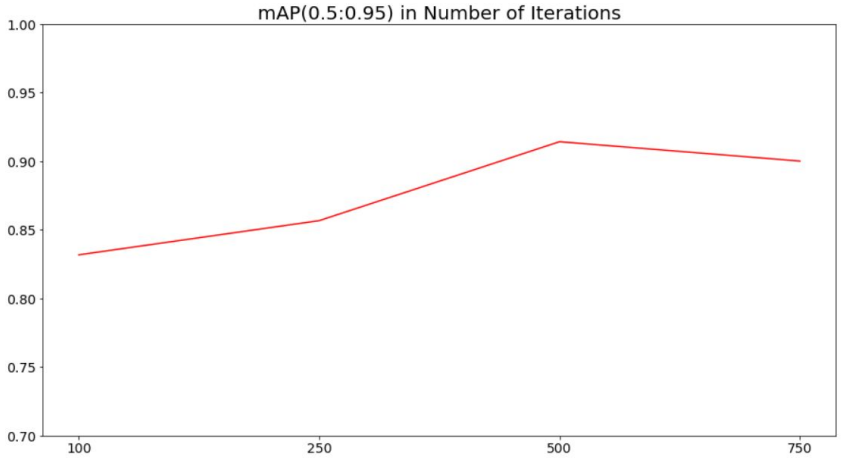
Cars that leave the scene are not detected for few frames,



Not detected because probably branches confuses network or is detected as a truck,

Task 1,2: Object Detection: Fine-tune (Team 4)

- We used detectron2's Trainer class to finetune the Faster RCNN network, RetinaNET is also considered but it requires more memory, thus harder to train,
- We see that 2 most critical parameters during finetuning is the learning rate and the number of iterations,
- We tried different learning rates but they don't seem to affect the performance too much, Number of iterations were important because too many iterations lead to overfitting, 500 seemed to be the ideal number,



Coco evaluation metrics:

AP: AP@(0,5:0,95)
AP50: AP@0,5
AP75: AP@0,75
APs: AP@(0,5:0,95) of objects between 0x0 and 32x32
APm: AP@(0,5:0,95) of objects between 32x32 and 96x96
APl: AP@(0,5:0,95) of objects bigger than 96x96

Network	AP	AP50	AP75	APs	APm	APl
Pretrained Faster RCNN	35,32	54,03	36,29	**27,22**	81,84	**23,31**
Pretrained RetinaNET	47,18	70,46	48,31	**37,71**	90,09	85,18
Finetuned Faster RCNN	**91,61**	**98,01**	**95,95**	**91,09**	**96,31**	**95,44**

Task 1,2: Object Detection: Fine-tune (Team 4)

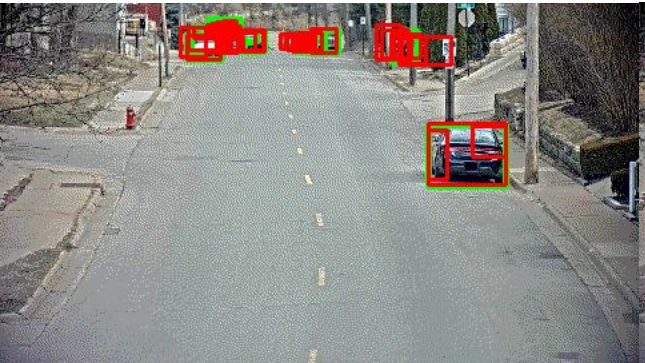
The most striking difference between a pretrained and finetuned Faster R-CNN is in small and large objects:

Change in AP with finetuning:

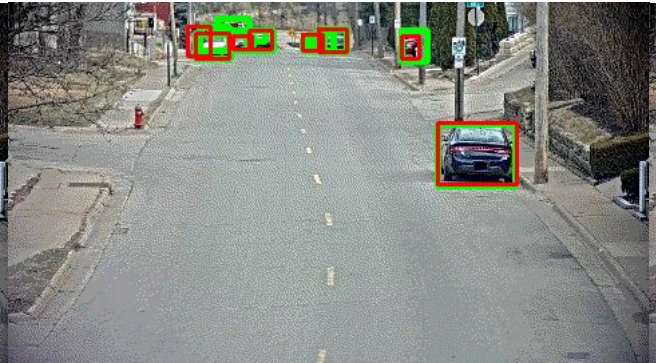
APs: %27,22 to %91,09, change = +%313
APm: %81,84 to %96,31 change = +%18,4
APl: %23,21 to %95,44 change = +%313

Pretrained RetinaNet makes too much predictions, it requires NMS, Pretrained Faster R-CNN isn't consistent, It can detect a car in a frame but not in the next one, Finetuning helps immensely, it can detect everything in this given sequence and it is consistent,

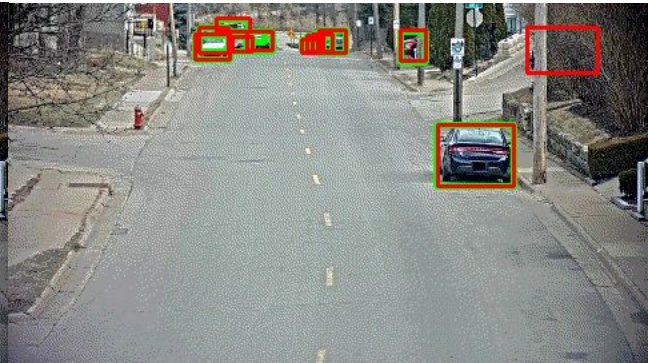
Pretrained RetinaNet



Pretrained Faster RCNN



Finetuned Faster RCNN



Task 1.2: Object Detection: Fine-tune (Team 5)

Considering the performance comparison of the pre-trained models, **RetinaNet - R101** has been selected and fine-tuned to the image sequence at hand.

Several learning rates are tested, the most effective one being 0.01

Lr	$AP_{0.5}$
0.001	98.939
0.005	98.985
0.01	98.981
0.1	Invalid (explodes after 450 iterations)

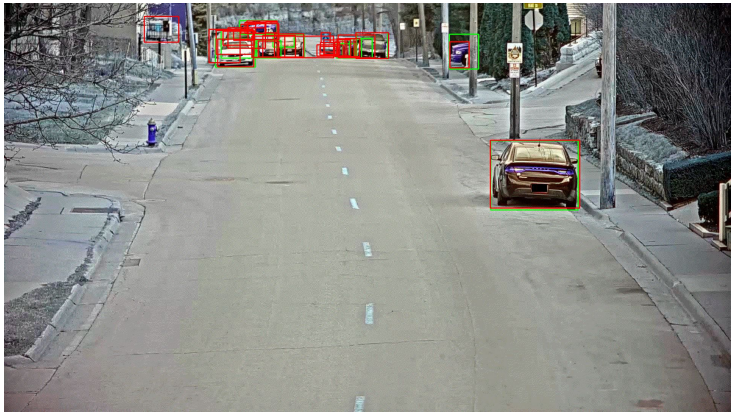
Freezing certain layers has also been considered, and it appears freezing at layer 2 or 3 yields the best AP.

Freeze at layer	$AP_{0.5}$
0	98.948
1	98.974
2	98.987
3	98.988

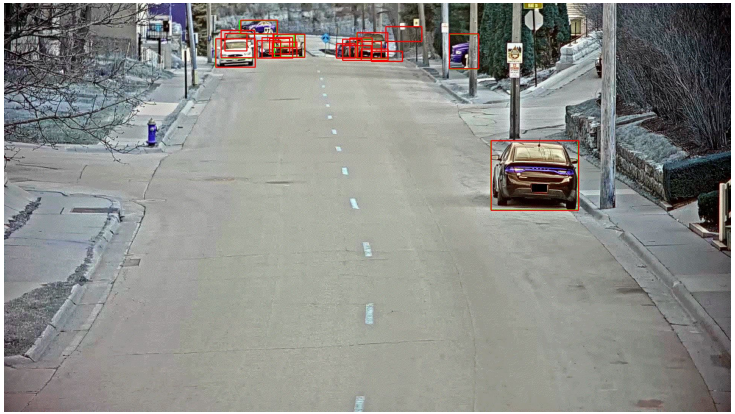
It is obvious how the trained model delivers a much better performance, presenting an increase of more than 26% in AP.

Task 1.2: Object Detection: Fine-tune (Team 5)

The improvement of the trained model can be appreciated comparing the animations below. The pre-trained model displays much more unstable detections compared to the trained one.

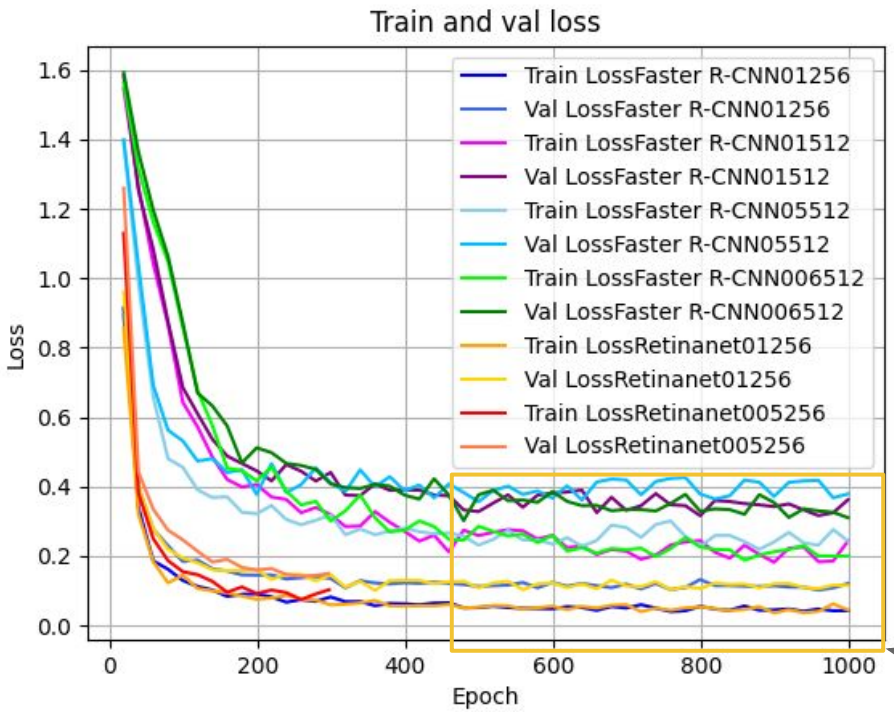


Pre-trained (Noisy)



Trained (Stable)

Task 1,2: Object Detection: Fine-tune (Team 6)



As the results are similar with all setups and fine-tuning is what defines the final performance, and after checking the general performance of different builds, for this task we chose work with R_50_FPN_3x build for Faster-RCNN and also R_50_FPN_3x build for RetinaNet because are a bit faster than other builds but results are good enough.
We tune learning rate, and batch size. Learning rate not seems to be a condicional, it change the speed but has similar final results, meanwhile changing batch size improve significantly the performance.
RetinaNet seems to have a better performance than Faster R-CNN, but this is just because the represented tries are with lower batch

With LR between 1-e^-2 and 1-e^-3 it converge in less than 600 epoch, after that there is no real improvement

Task 1,2: Object Detection: Fine-tune (Team 6)

Method	LR	Batch	AP	AP50
retinanet_R_50_FPN_3x	0.01	256	87.8	98.075
retinanet_R_50_FPN_3x	0.01	128	84.126	97.732
faster_CNN_R_50_FPN_3x	0.01	256	78.52	96.94
faster_CNN_R_50_FPN_3x	0.005	256	82.64	95.96

As the previous slides shows, retinanet with 256 batch has one of the best performance, differences between this option and others are trivial.

As we were following the previous task decision we didn't try to fine tune a different net until was too late to be implemented on following task before delivery time, improvements will be seen in final report.

Task 1.2: Object Detection: Fine-tune (Xavi)

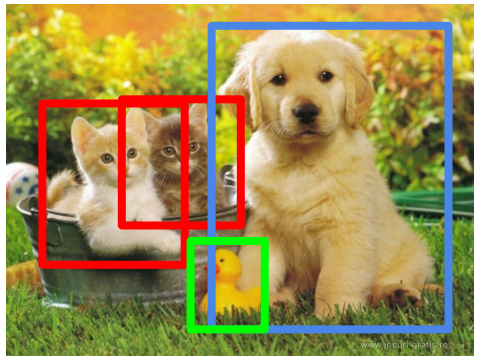
Repo	Feedback
Team 1	Fine-tune with the first 25% of the sequence. + Explored data augmentation implemented in Detectron2 (no significant gains). + Visually powerful qualitative results. - Does not directly compare the gains with respect to the none fine-tuned versions.
Team 3	? No improvement after fine-tuning. What was your training data ?
Team 5	Based on RetinaNet R1010. + Tested different LR and layers to unfreeze. - Does not directly compare the gains with respect to the none fine-tuned versions. ? What was the training data used ?

Task 1.2: Object Detection: Fine-tune (Javi)

Repo	Feedback
Team 2	Fine tuned RetinaNet and Faster R-CNN. Analysis of data augmentation. Optimizer hyperparameters reported. No mention of layers frozen. No mention of division of training / validation / test. Learning curves plotted (steps not shown) Qualitative and quantitative results shown. Clear discussions.
Team 4	Fine tuned Faster R-CNN. No report on data augmentation. Optimizer hyperparameters commented. No mention of layers frozen. No mention of division of training / validation / test. No learning curves plotted. Qualitative and quantitative results shown. Do not report that many metrics, you think they are useful? Clear discussions.
Team 6	Fine tuned RetinaNet and Faster R-CNN. No report on data augmentation. Optimizer hyperparameters commented. No mention of layers frozen. No mention of division of training / validation / test. Learning curves plotted. Qualitative but not quantitative results shown.

Tasks

- Task 1: Object detection
  - Task 1,1: Off-the-shelf
  - Task 1,2: Fine-tune to your data
  - **Task 1,3: K-Fold Cross-validation**
- Task 2: Object tracking



Task 1.3: K-Fold Cross-validation

Try different data partitions on your sequence:

Strategy A (same as week 2):

- First 25% frames for training
- Second 75% for test.

Strategy B:

- K-Fold cross-validation (use K=4).
- First 25% Train - last 75% Test (same as Strategy A).

Strategy C:

- K-Fold cross-validation (use K=4)
- Random 25% Train - rest for Test

Task 1,3: K-Fold Cross-validation (Team X)

- Copy & paste (max 1, pages)

Task 1.3: K-Fold Cross-validation (Team 1)

Strategy A	RetinaNet + FPN	Faster-RCNN + FPN	Faster-RCNN + DC5	Faster-RCNN + C4
AP50	95.81	95.63	95.87	94.96
AP75	87.46	78.50	87.57	85.45
AP-bike	71.38	58.18	64.74	61.54
AP-car	86.55	87.36	84.68	86.21

Problem: The bike class is underrepresented in latter segments of the video

It's better to keep one single partition that is representative enough of the statistics of the whole video or random sample frames. Successive frames are more similar to each other.

RetinaNet has many false positives in their predictions from classes found in COCO.

Strategy B	RetinaNet + FPN	Faster-RCNN + FPN	Faster-RCNN + DC5	Faster-RCNN + C4
AP50	95.52	71.81	71.98	72.10
AP75	87.81	65.75	66.97	67.90
AP-bike	71.39	31.94	34.11	33.45
AP-car	86.72	87.19	85.24	86.89

Strategy C	RetinaNet + FPN	Faster-RCNN + FPN	Faster-RCNN + DC5	Faster-RCNN + C4
AP50 (avg)	97.85	97.26	97.61	97.37
AP75 (avg)	93.59	94.32	94.30	93.99
AP-bike (avg)	80.15	78.04	80.36	79.66
AP-car (avg)	89.61	91.06	89.69	90.55

Task 1.3: K-Fold Cross-validation (Team 2)

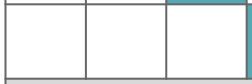
We have tested two ways of evaluating the system:

- ❏ First 25% for training and the remaining 75% for validation. (strategy A)
- ❏ Dividing the sequence in 4 folds and using each fold once for training and the remaining for validation. (strategy B)

Strategy A

Training Split in Blue	RetinaNet	Faster R-CNN
	0.9700	0.9852

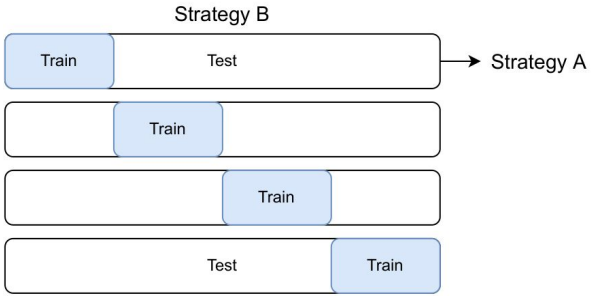
Strategy B

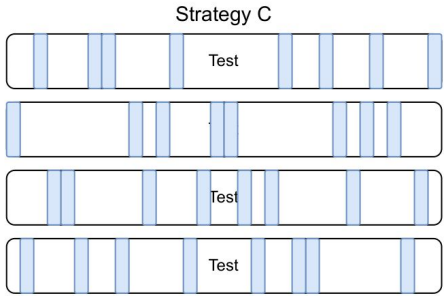
Training Split in Blue	RetinaNet	Faster R-CNN
	0.9700	0.9852
	0.9542	0.9692
	0.9727	0.9859
	0.9663	0.9831
Strategy B avg	0.9658	0.9809

- ❏ With cross-validation we obtain a more truthful evaluation of the models since every part of the sequence is used for training and validation. However, we obtain a quite similar results along all the different splits.
- ❏ The main drawback of cross-validation is that the time has been increased from **30 min to almost 2 hours** using a NVIDIA GeForce RTX 3060.

Task 1,3: K-Fold Cross-validation (Team 3)

We have used cross-validation with 4 folds to fine-tune the **RetinaNet** model (we also include the evaluation of the same folds for the pre-trained model to compare results). The fine tuning has been trained for 3 epochs with the best values for the hyperparameters that we found before (lr=0.005, momentum=0.9, weight_decay=0.0005). The three proposed strategies (two in reality) have been tried:





The strategy that gives better results is C, the one that uses 25% random frames for training. Probably because in this case we are considering frames with different illuminations and objects in different positions. But the difference with the other strategies is minimal.

	Strategy	Test $AP_{0.5}$ for fold #				Mean $AP_{0.5}$
		1	2	3	4	
Pre-trained	B	68.79% (Strategy A)	68.09%	69.33%	70.49%	69.18%
	C	69.11%	69.12%	69.28%	69.21%	69.18%
Fine-tuned	B	90.88% (Strategy A)	90.76%	90.89%	90.90%	90.86%
	C	**90.90%**	**90.90%**	**90.91%**	**90.91%**	**90.90%**

Task 1,3: K-Fold Cross-validation (Team 4)

We tried all the strategies, and we showcase their performance for each metric in the table. For each category, best performing strategy is C, which is to split train-test randomly in every fold. We think it is because with random splitting, network is able to see many frames from different parts and circumstances of the object, so it generalizes better.

Strategy	AP	AP50	AP75	APs	APm	APl
A	85,71	96,94	93,86	84,96	91,77	84,25
B	90,11	96,99	93,99	89,22	95,68	95,18
C	**91,61**	**98,01**	**95,95**	**91,09**	**96,31**	**95,44**

Task 1.3: K-Fold Cross-validation (Team 5)

As proposed different methods to train the model have been tested in terms of data splits and k-fold cross-validation. We try to see if the choice of frames for training has a real impact or not, in an attempt to make estimations more robust. The result show that there is not much of a difference at all between these different approaches when it comes to the AP ratings they each result in.

	$AP_{0.5}$	$AP_{0.75}$
Classic training	98.98	97.00
K-Fold (first 25% training)	98.92	96.98
K-Fold (random 25% for training)	98.94	97.06

Task 1.3: K-Fold Cross-validation (Team 6)

In this case, the train and validation sets are defined using **K-Fold Cross-validation** with **K = 4**. That is, the 25% of the frames is divided in 4 subsets, and we evaluate each of them after training the model for the rest K-1 subfolds. Using strategy B, the 25% of frames used for training/validation correspond to the initial subsequence of the video, while in strategy C, the frames are selected randomly from the entire sequence. The corresponding experiments are presented in the following table:

Strategy	Split 1	Split 2	Split 3	Split 4	Mean $AP^{0.5}$
A	0.96	-	-	-	0.96
B	0.99	0.99	0.88	0.98	0.96
C	0.96	0.98	0.97	0.99	**0.98**

The table results are obtained from **Faster RCNN** with a learning rate of 1e-3.

We can observe that when using the **strategy C**, we obtain **better results**, which mean that using random selected samples of the sequence is a better strategy to help the model to better generalize to unseen scenes (even if they are similar during the entire sequence).

Task 1: Object Detection

Team ID	Model (choose one)	T1,1 Off-the-shelf (mAP_{50}) (Random 3-Fold cross-validation)	T1,2 Fine-tuned (mAP_{50}) (Random 3-Fold cross-validation)
Team 1	RetinaNet + FPN	51.87 %	97.85 %
Team 2	Faster R-CNN (X101-FPN)	48.55 %	98 %
Team 3	RetinaNet	69.2%	90.9%
Team 4	Faster RCNN	%53,1	%98
Team 5	RetinaNet R101	72.7%	98.9%
Team 6			98.07%

Task 1.3: K-Fold Cross-validation (Xavi)

Repo	Feedback
Team 1	+ Explains why random sampling for the 25% train (Strategy C) provides the best results. - Does not discuss the difference between one run (Strategy A) and K=4 runs (Strategy B). - Some visuals of the configurations may help understading better.
Team 3	+ Nice visuals about the three strategies. + Explains why random sampling for the 25% train (Strategy C) provides the best results. + Explored both off-the-shelf and fine-tuned configurations. - Does not discuss the difference between one run (Strategy A) and K=4 runs (Strategy B).
Team 5	- Some visuals of the configurations may help understading better. - Does not discuss the difference between one run (Strategy A) and K=4 runs (Strategy B). - The final table of results is not completed for this team. ? Obtained results are much more similar for all strategies than other teams. Why ?

Task 1.3: K-Fold Cross-validation (Javi)

Repo	Feedback
Team 2	Tried 2 strategies. Good visual explaining the strategies. Discussions about time constraints. Strategy C why not?
Team 4	Tried 3 strategies. Visuals explaining the strategies missing. Clear discussions about validity of strategy C. No need of using 6 metrics which report the same thing.
Team 6	Tried 3 strategies. No visuals explaining the strategies, but explained in the text. Clear discussions about validity of strategy C.

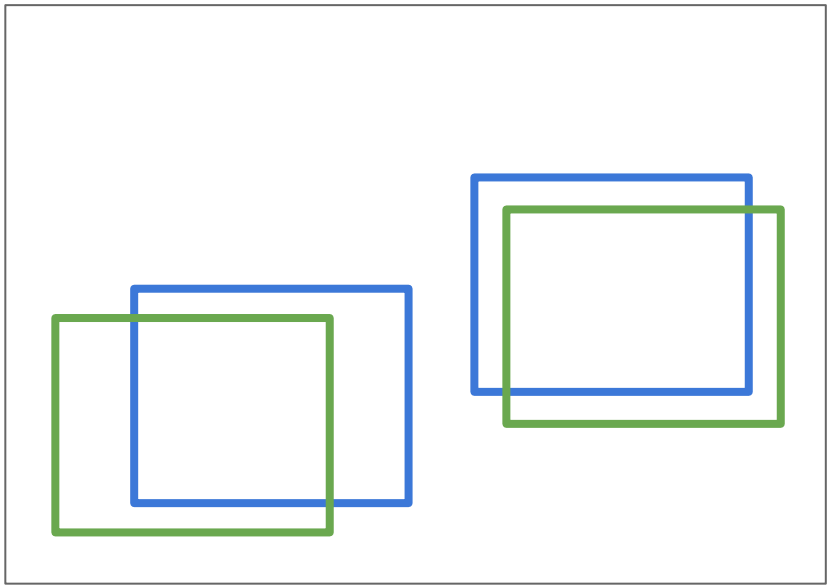
Tasks

- Task 1: Object detection
- Task 2: Object tracking
  - **Task 2,1: Tracking by Overlap**
  - Task 2,2: Tracking with a Kalman Filter
  - Task 2,3: IDF1 score

Task 2.1: Tracking by Maximum Overlap

Basic algorithm (your task is to modify / improve it based on your experiments):

1. Assign a unique ID to each new detected object in frame N.
2. Assign the same ID to the detected object with the highest overlap (IoU) in frame N+1.
3. Return to 1.



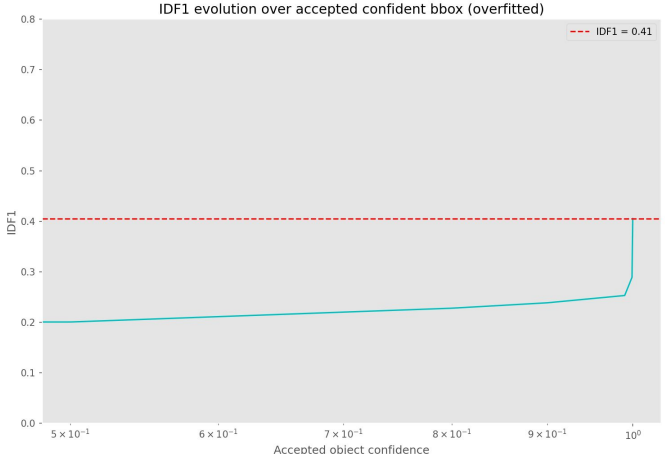
Detected Objects @ Frame N

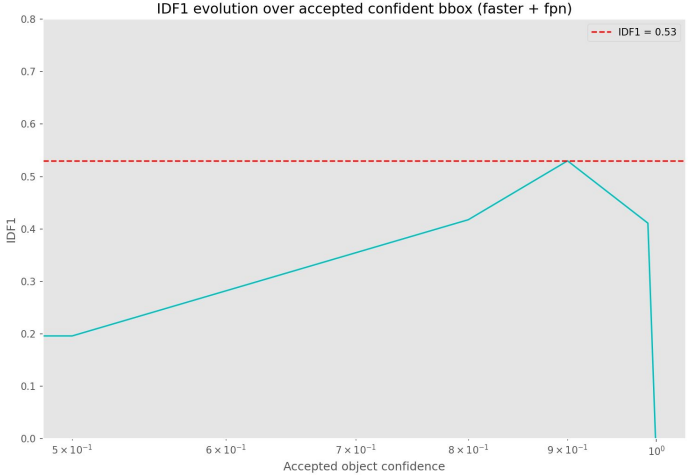
Detected Objects @ Frame N+1

Task 2,1: Tracking by Maximum Overlap (Team X)

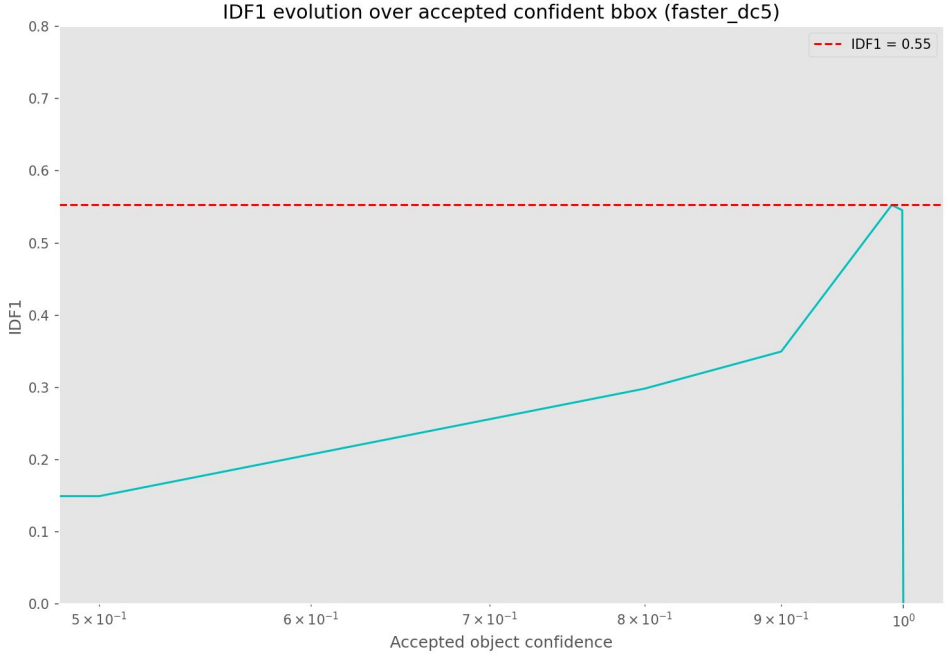
- Copy & paste (max 2, pages)

Task 2.1: Tracking by Maximum Overlap (Team 1 - 1/2)

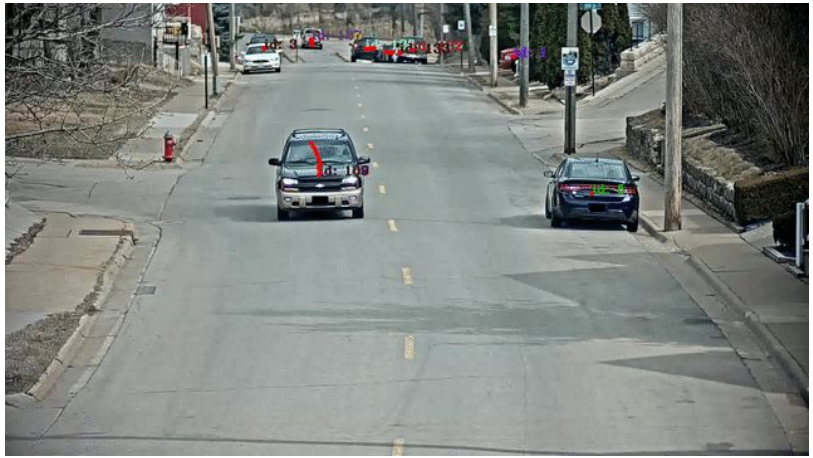




We found that some models we trained were overly confident in their predictions (>.99 score in most boxes). We re-trained them and adjusted the acceptance threshold for better tracks.
We also found that the background was the trickiest part, as we had many changing tracks.



Task 2.1: Tracking by Maximum Overlap (Team 1 - 2/2)



Trail cleared every 50 frames for legibility.

Parked vehicles that are far away from the cameras are updated with new track id's due their bounding box instability.

Decreasing IoU threshold helped to reduce the number of updated tracks.

Task 2.1: Tracking by Maximum Overlap (Team 2) (1/2)

The **Maximum Overlap** tracking method estimates the track of an object computing the maximum IoU between the current bounding box in frame N and the detected bounding boxes in frame N-1. The algorithm performs as follows:

1. Assign a unique ID to each new detected object in frame N.
2. Assign the same ID to the detected object with the highest overlap (IoU) in frame N+1.
3. Return to 1

Model	$AP_{0.5car}$	IDP	IDR	IDF1
Off-the-shelf Faster R-CNN	0.4855	0.7227	0.5962	0.6534
Off-the-shelf RetinaNet	0.7269	0.2441	0.6748	0.3585
Fine-tuned RetinaNet	0.9700	0.1703	0.6604	0.2708
Fine-tuned Faster R-CNN	0.9852	0.7661	0.7651	0.7656

[1] https://github.com/abewley/sort [2] https://github.com/cheind/py-motmetrics

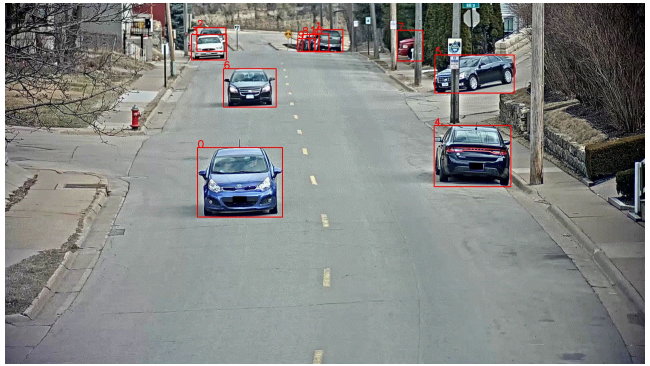
We can see that applying Maximum Overlap to detections made by **Faster R-CNN fine tuned** performs much better. This is because the detections are more robust.

In **Faster R-CNN** it can be seen that the fine-tuned version performs better as it has better predictions.

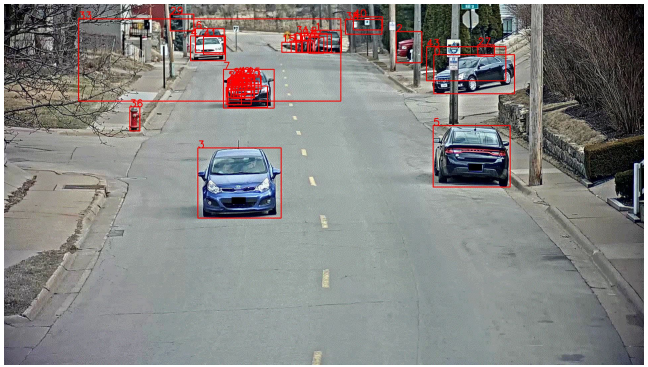
In the case of **RetinaNet** it seems to perform worse, maybe to noisy detections or some implementation errors.

Task 2.1: Tracking by Maximum Overlap (Team 2) (2/2)

Faster R-CNN Fine tuned



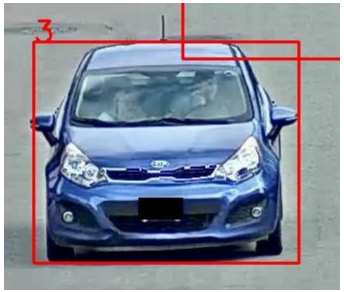
RetinaNet Fine Tuned



In this two representations we can see, how track detections using **Retinanet** are **much noisier** than the ones made by **Faster RCNN**.

Other problems that can be seen is that **tracks can be splitted** in two if some object position in the track has not been detected.

Example of track splitting using RetinaNet Fine Tuned



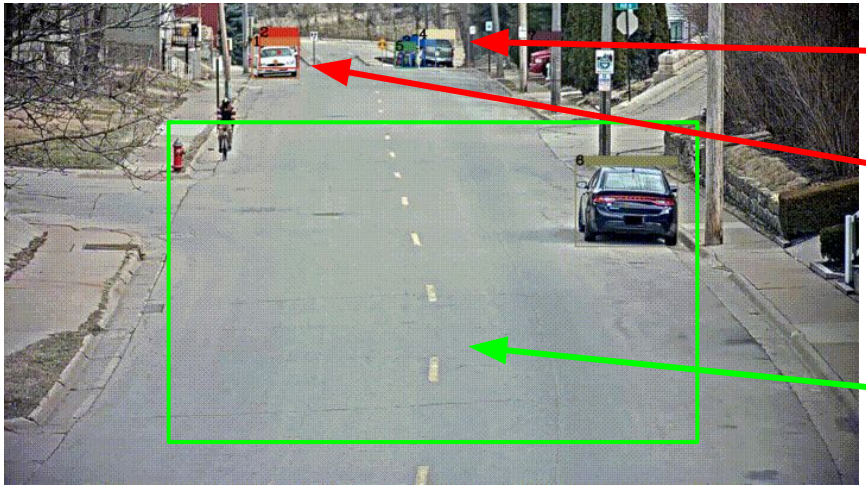
Frame 542

Frame 543

Task 2,1: Tracking by Maximum Overlap (Team 3 1/2)

We have implemented the tracking by maximum overlap in the following way:

- Create a new track from each detection in the first frame,
- For the following frames:
  - 1, For each detection over a threshold, compute the maximum overlap bounding box from a track,
  - 1a: If the IoU of detection and bounding box is over a threshold → Add the bounding box to the tracking,
  - 1b: Otherwise, start a new track with the detection,
  - 2, Remove tracks with no match in the previous *n* frames,



Problem:

- When cars pass in front of other cars, tracks can be confused and some of them might be lost,
- If a car is not detected in the whole sequence, that track is lost,

Good: Cars that are not occluded by others are correctly tracked for the whole sequence,

Task 2,1: Tracking by Maximum Overlap (Team 3 2/2)

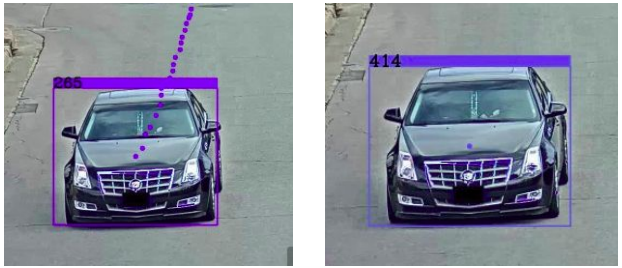
- By keeping a track alive for a few frames allows to keep tracks correct for the whole sequence, for example this can be seen in parked cars:

Example of lost track due to not detecting the car in one frame,

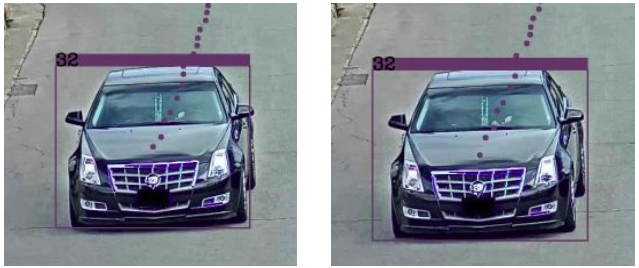
Example of recovered track by maintaining the track alive for more than 1 frame,

- The minimum IoU for boxes between frames has to be high enough to not miss cars that move considerably between frames,

Example of lost track due to using a minimum IoU that is too high with IoU ~= 0,9,



Example of the same car correctly tracked with minimum IoU ~= 0,4,



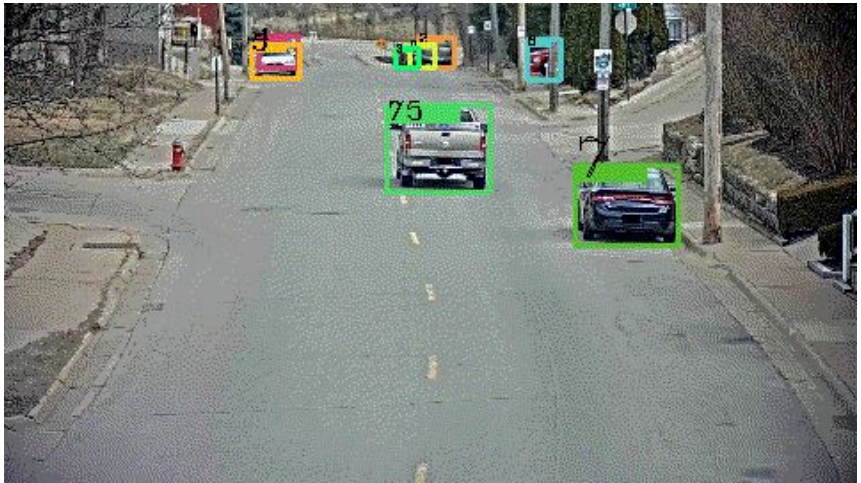
Task 2,1: Tracking by Maximum Overlap (Team 4)

We used the finetuned Faster R-CNN. Our algorithm step was as follows:

- Assign IDs for each detection
- Check IoU criteria (>threshold)
- Assign the same ID if it holds
- Discard old IDs after 5 frames of no matches

Performance of different thresholds can be seen in the table. We see that 0.5 is the best performing one.

Threshold = 0.5



Threshold	IDF1	IDP	IDR
0.3	0.6631	0.7049	0.6305
0.4	0.7888	0.8226	0.7607
0.5	**0.8412**	**0.8416**	**0.8405**
0.6	0.8257	0.8261	0.825
0.7	0.809	0.809	0.808
0.8	0.7063	0.7066	0.7057
0.9	0.5407	0.5409	0.5402

IDP: global min-cost precision
IRD: global min-cost recall
IDF1: global min-cost F1 score

Too low or too high thresholds don't seem to work. In values bigger than 0.4, Recall and Precision is nearly identical, but in smaller values recall drops.

Task 2,1: Tracking by Maximum Overlap (Team 4)

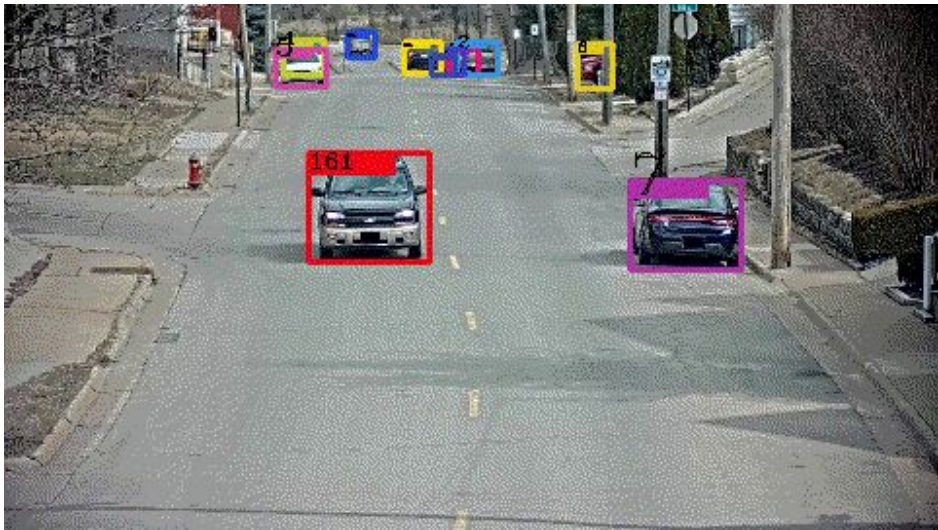
From the IDF1 results and what we see from visualizations, Maximum Overlap method works in most cases but there are some particular problems, for example tracking may fail for difficult to detect objects such as distant ones. A more common problem is that cars that are about the leave the camera is tracked as new objects.

Another problem is that when boxes overlap, track is generally lost. This especially occurs in cars that are in distance.

A car in distance that is about the leave the camera loses the track.

In this case, car is close but it is still loses the track.

Cars that are in distance loses their trackes because they overlap.



Task 2.1: Tracking by Maximum Overlap (Team 5) (1/2)

Tracking Workflow

Initial Frame:

1. Select a detection algorithm from **Task1** to obtain the detection bounding boxes of all vehicles.
2. A **tracking id** is assigned to each detected vehicles in the scene.

Tracking:

1. **Filtering**: Compute the IoU amongst the bounding boxes within the same frame and remove overlapping boxes. This is performed to remove overlapping parked car from afar. (**IoU>0.9**)
2. When given a new frame, compute the **IoUs** of all new detections with the current detections.
3. Assign the new detections to the current tracks with the maximum **IoU**, only if the **IoU** is higher than **0.5**.
4. If any new detections does not match any existing tracks assign a **new track id** to the new detections.

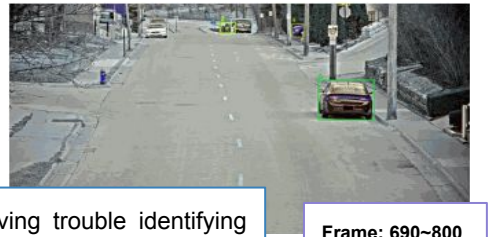
Metric	Fine-tuned Retina 101
IDF1	**0.724**
IDP	0.577
IDR	0.972
Precision	0.593
Recall	1

Consistent with what we observed earlier in the precision-recall curve, our **IDR** is a lot higher than **IDP**.

Task 2.1: Tracking by Maximum Overlap (Team 5) (2/2)

Frame: 750~850

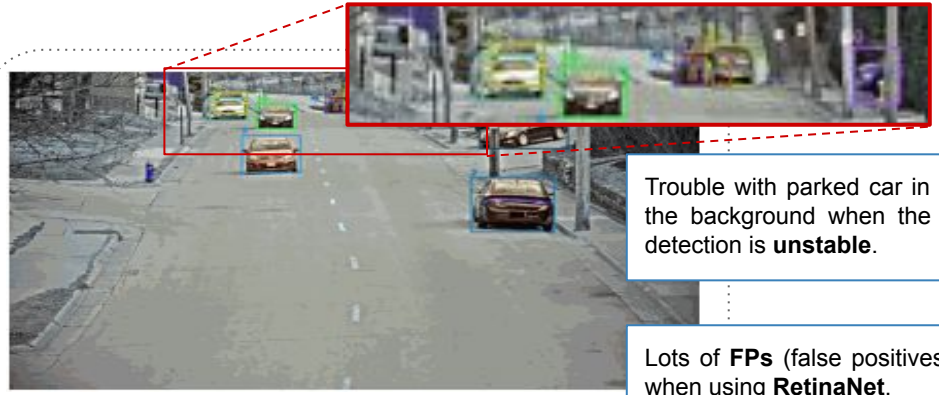
Maximum overlap method becomes problematic when tracked objects **overlap** with other objects.



Frame: 690~800

Detector having trouble identifying whether the detected vehicle is a car or a van when using a **pre-trained detector**. This led to tracking flickering of the same object.

Comparison



Tracking with fine-tuned RetinaNet101

Trouble with parked car in the background when the detection is **unstable**.

Lots of **FPs** (false positives) when using **RetinaNet**.

Tracking with pre-trained Mask R-CNN 101

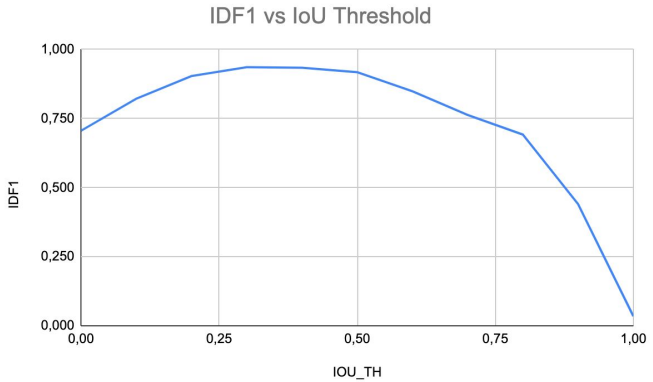
Fame: 500 ~1000

Task 2,1: Tracking by Maximum Overlap (Team 6 1/2)

Example of output with GT detections. Car labelled with its tracking Id. In this image we can see how the cars Ids are stable.

Cars with different IDs overlap between them. In some cases the id is generated a new value in other cases the id is transferred between cars. (IoU threshold = 0.4)

Task 2,1: Tracking by Maximum Overlap (Team 6 2/2)



Best IoU = 0.5 (IDF1 = 0,918)

Detector/Implementation	IoU (IoU = 0.5)	IoU (Direction angle, IoU = 0.5)
Annotations file	0,918	0,918
retinanet_R_50_FPN_3x	0,633	0,633
faster_rcnn_R_50_FPN_3x	0,711	0,711
retinanet_R_101_FPN_3x	0,585	0,585
faster_rcnn_X_101_32x8d_FPN_3x	0,705	0,705

- We implemented two versions of IoU, the second one includes a try to filter boxes that change the angle direction. How can see in the results the effect of this is nothing and it has to be reviewed.
- A high IoU threshold makes loss the tracking, the low threshold has a minor impact because the tracking of individual cars that do not overlap with another is kept.
- We use Ground truth to test our algorithm a get good scores, in the real problem (retinanet, faster) the score is lower because of the existence of the parked cars, which generate some errors.

Task 2.1: Tracking by Maximum Overlap (Xavi)

Repo	Feedback
Team 1	Explores the impact of acceptance threshold over IDF1. Parked cars generate tracks because of box instability. + Qualitative discussion backed by visualizations. - Explored with a single object detector. - Missing an explanation of the details of the implemented algorithm ? What object detector did you use ?
Team 3	+ Explain the way to keep tracks alive. + Spotted limitations with a qualitative & visual analysis. - The IoU threshold used is not reported, neither studied as a hyperparameter to tune. - Explored with a single object detector. ? What object detector did you use ?
Team 5	+ Introduce a step to remove park cars by considering IoU > 0.9. + Spotted limitations with a qualitative & visual analysis. ? Did you compute the IDF1 for Mask R-CNN as well ?

Task 2.1: Tracking by Maximum Overlap (Javi)

Repo	Feedback
Team 2	Qualitative and quantitative (IDF1=0.76) results provided. Highest overlap on IoU Good discussions.
Team 4	Qualitative and quantitative (IDF1=0.84) results provided. IoU threshold (tested 0.3 to 0.9) Good discussions.
Team 6	Qualitative and quantitative (IDF1=0.71) results provided. IoU threshold (tested 0.0 to 1.0) Good discussions but try to organize it better (first slide feels put it randomly without context) Bonus point: Filtering by direction angle.

Tasks

- Task 1: Object detection
- Task 2: Object tracking
  - Task 2,1: Tracking by overlap
  - **Task 2,2: Tracking with a Kalman Filter**
  - Task 2,3: IDF1 score

Task 2,2: Tracking with a Kalman Filter (Team X)

- Copy & paste (max 2, pages)

Task 2.2: Tracking with a Kalman Filter (Team 1)

We have used the SORT (Simple Online and Realtime Tracking) implementation, for more information go here. We have followed this tutorial.

The Kalman filter uses detected bounding boxes as as its inputs. We will use the bounding boxes detected using Faster R-CNN with FPN.

This implementation uses a constant velocity model with the top left and bottom right coordinates of the bounding box and the velocity of these coordinates as state variables.

We have had a problem with our implementation and we have not been able to obtain reasonable results as the IDF we got is 0.0.

Task 2.2: Tracking with a Kalman Filter (Team 2) - (1/2)

The **Kalman Filter** processes measurements to deduce the **optimal estimate** of the state of a **linear system** using a set of measurements and a **statistical model** of the system. We have used the implementation of Alex Bewley's [1].

The advantages of using this algorithm is that the system becomes more robust against missing bounding boxes and it allows to identify the objects along the video sequence making the prediction where a bounding box from the previous would be located at the moment.

To evaluate the methods we have used the IDP IDR and IDF1 from py-motmetrics [2].

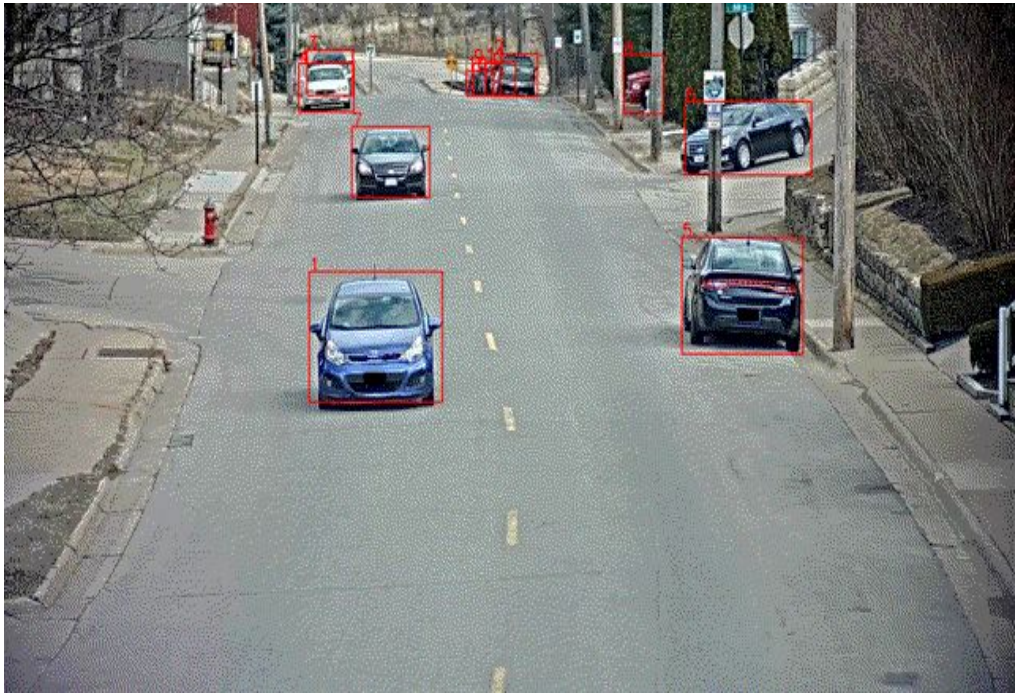
Model	$AP_{0.5car}$	IDP	IDR	IDF1
Off-the-shelf Faster R-CNN	0.4855	0.8920	0.6783	0.7706
Off-the-shelf RetinaNet	0.7269	0.2376	0.7059	0.3556
Fine-tuned RetinaNet	0.9700	0.3224	0.7209	0.4456
Fine-tuned Faster R-CNN	0.9852	0.7936	0.7687	0.7810

As expected the fine-tuned models performed better at tracking the cars since they have better predictions. However, RetinaNet performs worse than Faster R-CNN due to its predictions were much more noisier (analysed in task 1).

[1] https://github.com/abewley/sort [2] https://github.com/cheind/py-motmetrics

Task 2.2: Tracking with a Kalman Filter (Team 2) - (2/2)

Qualitative results:



Parked and isolated cars are tracker perfectly.

Nevertheless, the track IDs end up being extremely high due to occlusions since the algorithm is not capable of remembering the object during a short period of masking.

Frame 750
Track IDs: 2, 9, 10 and 11

Frame 910
Track IDs: 2, 29, 36 and 42

Task 2,2: Tracking with a Kalman Filter (Team 3) 1/2

Tracking

Kalman filter: constant velocity model

- State: $\mathbf{x}_t = (\mathbf{p}_t, \mathbf{v}_t)^T$
- Motion model:

we move in the direction of $\mathbf{v}_t$

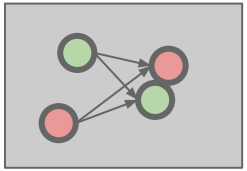
$$\mathbf{x}_t = \underbrace{\begin{bmatrix} \mathbf{1} & \Delta t \\ \mathbf{0} & \mathbf{1} \end{bmatrix}} \mathbf{x}_{t-1}$$

velocity remains constant

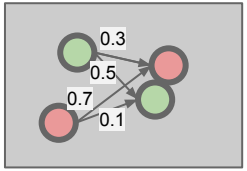
In SORT*, $\mathbf{p}_t$ is composed of the bbox center and area (which also grows or shrinks with constant velocity)

Association

For **multiple object tracking**, which detection on frame *t+1* corresponds to every track in frame *t*?



For every candidate pair, we can compute some **similarity** value: our implementation* uses the **IoU** of the detected boxes and the predictions.



The assignment problem can then be reframed as a max-flow min-cut problem: we maximize the total similarity score using the least number of edges. For **bipartite graph matching** (our case! two frames) there is a polynomial time method, the Hungarian algorithm.

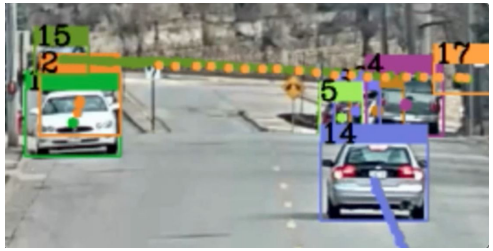
*We use the implementation used on this tutorial by Amaia Salvador, which is explained on this paper.

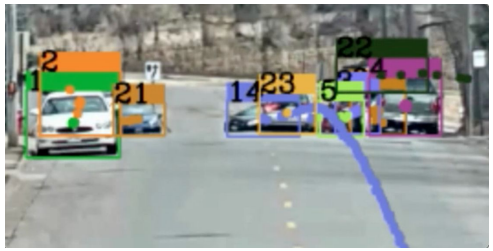
Task 2,2: Tracking with a Kalman Filter (Team 3) 2/2

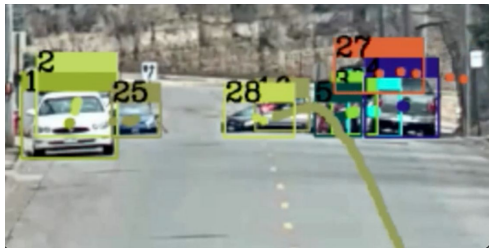
Maximum overlap

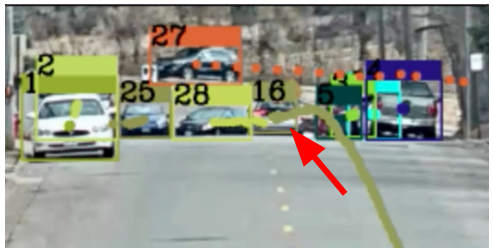
Kalman filter

time









The new car *steals* the track (14), and a new one is assigned to the previous car (23)

When isolated, the cars are tracked alright.

Two cars cross!

Kalman filter handles this situation well.

IDF1 scores with Kalman tracking

O.D. model	IDF1
MaskRCNN	0.7879
FasterRCNN mobile	0.7607
SSD	0.5731
FasterRCNN	**0.8122**
RetinaNet	0.7645

Task 2,2: Tracking with a Kalman Filter (Team 4)

We used SORT (Simple Online and Realtime Tracking).
For more information on SORT: https://arxiv.org/pdf/1602.00763.pdf

From the paper: *" Despite only using a rudimentary combination of familiar techniques such as the Kalman Filter and Hungarian algorithm for the tracking components, this approach achieves an accuracy comparable to state-of-the-art online trackers"*

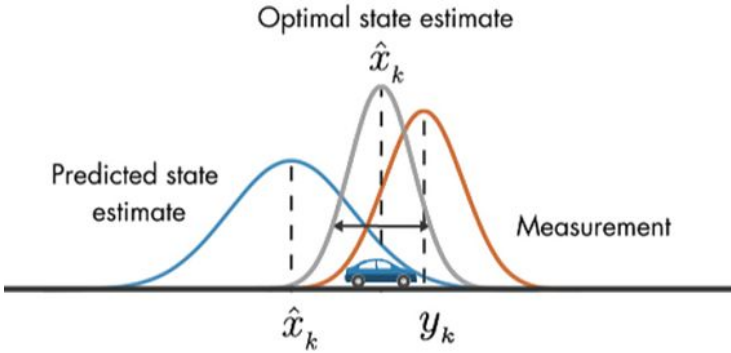
There are 2 main parameters to decide while using this library:

Max Age: Maximum number of frames to keep alive a track without associated detections.
Min Hits: Minimum number of associated detections before track is initialised.

We did a grid search on this parameter, trying values between 1 and 15 for both of them.

Results with the best parameters:

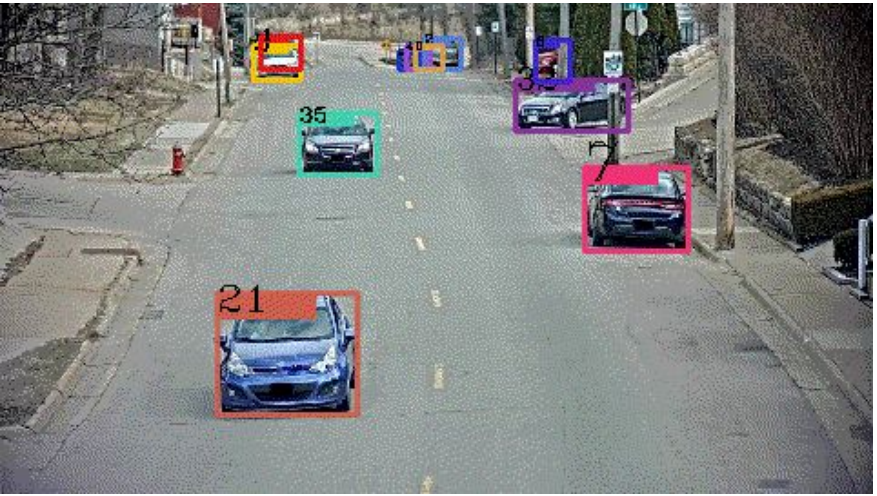
Max Age	Min Hits	IDF1	IDP	IDR
11	3	0.8659	0.8764	0.8553



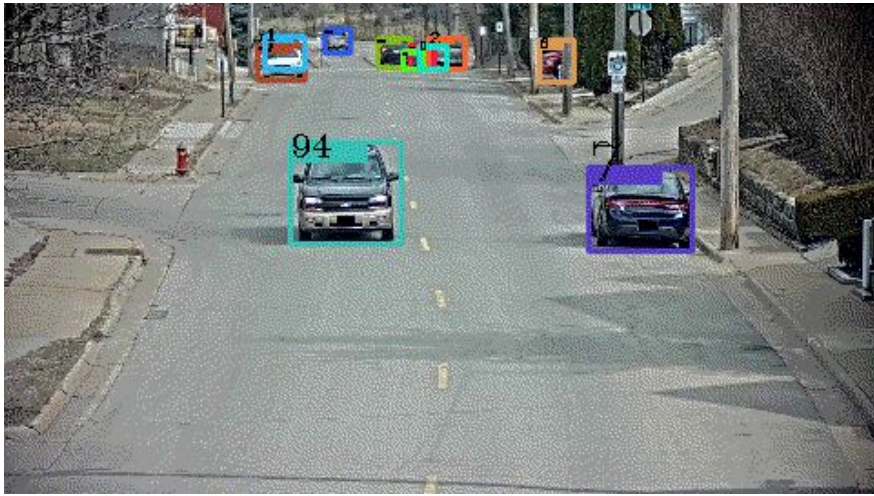
Task 2,2: Tracking with a Kalman Filter (Team 4)

Cars in distance is still harder to track, but Kalman handles this instances better than Maximum Overlap. Overall, we find Kalman as a more robust method.

3 cars that leaves the frame aren't tracked as new ones just before leaving, unlike Maximum Overlap.

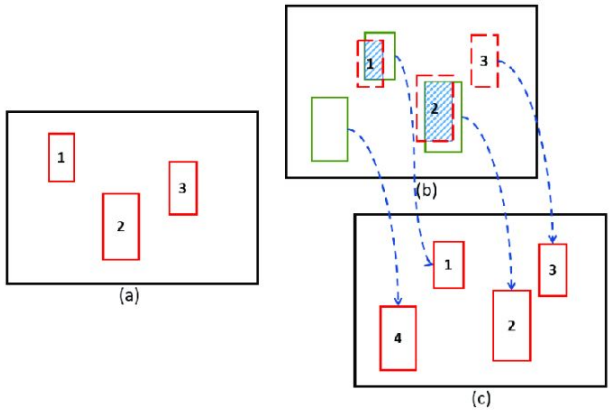


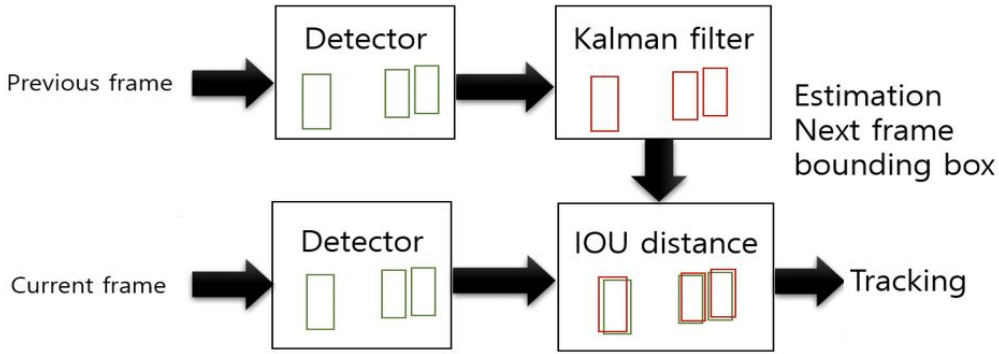
Kalman is able to track the 3 cars that are far away and overlap.



Task 2.2: Tracking with a Kalman Filter (Team 5)

- Inspired by Amaia Salvador tutorial[1], our implementation relies on SORT (Simple Online and Realtime Tracking)[2].
- SORT uses kalman filter (constant velocity) for object tracking without using ego-motion.
  - Information is corrected from associating objects in 2 adjacents frames (e.g. A detected at frame t and B detected at frame t+1.)
  - Some criteria have to be used to associate objects in adjacents frames (e.g. IoU).
  - Kalman Filter can use B's location at t+1 as a new measurement for A to update the states.
  - Association plays a critical role.
- Kalman Filter based methods perform better than Maximum Overlap based methods and fix the problem with faulty detections caused by overlapping objects.





[1] Amaia Salvador Tutorial
[2] Simple Online and Realtime Tracking (SORT) implementation

Task 2.2: Tracking with a Kalman Filter (Team 5)

The tracking using the Kalmar approach indeed seems to deliver better results in general if the displayed metrics are observed. Its IDF1 score is around 8% higher than that of the tracking using the maximum overlap.

This improvement most likely has to do with the ability of the model to better maintain the tracking of detections under occlusion situations, making it more robust and thus better performing in multiple object tracking

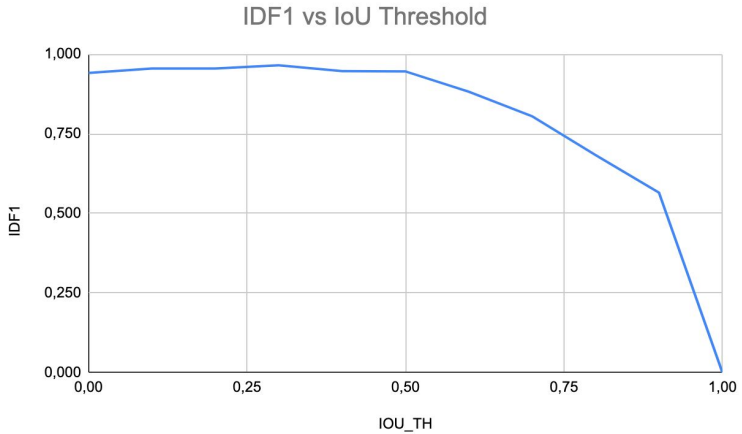
Metric	Fine-tuned Retina 101
IDF1	**0.808**
IDP	0.749
IDR	0.877
Precision	0.846
Recall	0.990

Task 2,2: Tracking with a Kalman Filter (Team 6 1/2)

- For this task we try to use two implementations of Kalman Filter, In the first one we have any errors (IDF1 0.8 with GT) and we focus the report on the second one (Sort implementation)

- Qualitatively we can conclude that Kalman Filter has the robustness to overlap boxes and maintains the correct labels with more accuracy than IoU implementation.

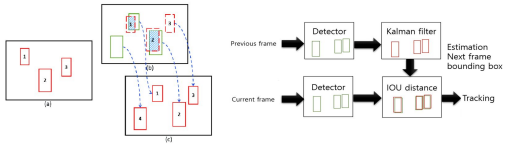
Task 2,2: Tracking with a Kalman Filter (Team 6 2/2)



Best IoU = 0.3 (IDF1 = 0,968)

Detector/Implementation	Kalman (max_age=5, IoU=0.3)
Annotations file	0,968
retinanet_R_50_FPN_3x	0,694
faster_rcnn_R_50_FPN_3x	0,770
retinanet_R_101_FPN_3x	0,617
faster_rcnn_X_101_32x8d_FPN_3x	0,791

Task 2.2: Tracking with Kalman filter (Xavi)

Repo	Feedback
Team 1	SORT - Missing some reminder of how Kalman filter works. - This part was not completed.
Team 3	SORT + Hungarian algorithm (Amaia Salvador) + Provide an overview of the Kalman filter. + Apply the Hungarian algorithm to solve the assignment problem. + Tests over 5 object detectors. - Still having problems with the parked cars.
Team 5	SORT + Hungarian algorithm (Amaia Salvador) + Provide an overview of the Kalman filter & SORT. - Qualitative results may have helped in better understanding. ? Did you create the figures in the slides yourselves ? 

Task 2.2: Tracking with Kalman filter (Javi)

Repo	Feedback
Team 2	SORT implementation No slides explaining/reviewing Kalman filtering Qualitative and quantitative results (IDF1=0.78) provided No discussion or comparison with maximum overlap.
Team 4	SORT implementation No slides explaining/reviewing Kalman filtering Performed a grid search for hyperparameters. Qualitative and quantitative results (IDF1=0.86) provided Comparison with maximum overlap but no values provided.
Team 6	Another implementation (your own?) + SORT implementation No slides explaining/reviewing Kalman filtering Qualitative and quantitative results (IDF1=0.79) provided Comparison with maximum overlap but no values provided. Have you got an explanation why the IoU threshold does NOT affect the IDF1 on low values? It did before on maximum overlap.

Tasks

- Task 1: Object detection
- Task 2: Object tracking
  - Task 2,1: Tracking by overlap
  - Task 2,2: Tracking with a Kalman Filter
  - **Task 2,3: IDF1 score**

Task 2.3: IDF1 for Multiple Object Tracking

Team ID	T2,1 Tracking by overlap	T2,2 Tracking with a Kalman filter
Team 1	0.552654	0.0
Team 2	0.7656	0.7810
Team 3	0.801	0.811
Team 4	0.8412	0.866
Team 5	0.724	0.808
Team 6	0.711	0,791

Task 2.3: IDF1 for Multiple Object Tracking (Xavi)

Repo	Feedback
Team 1	
Team 3	
Team 5	

Task 2.3: IDF1 for Multiple Object Tracking (Javi)

Repo	All show small improvements using Kalman filtering
Team 2	
Team 4	
Team 6	

Tasks

- Task 1: Object detection
- Task 2: Object tracking
- **(optional) Task 3: CVPR 2021 AI City Challenge**

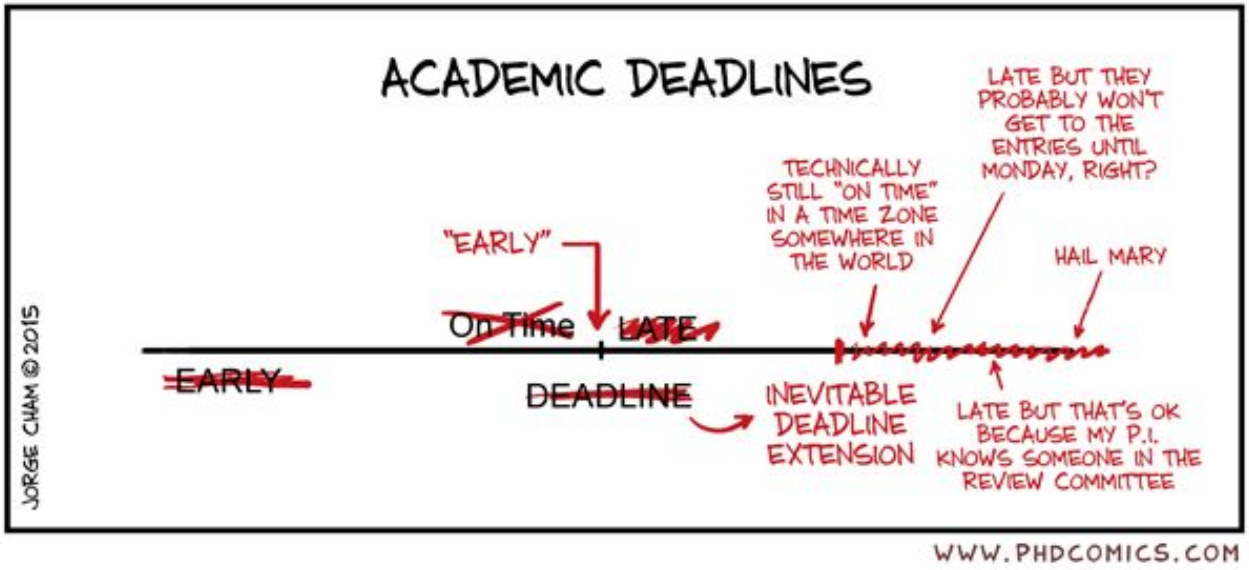
Task 3: CVPR 2021 AI City Challenge (Team X)
- Copy & paste (max 2, pages)

No team submitted the optional task

Scoring Rubric

Task	Description	Max, Score
T1	Object Detection	1
T1,1	Off-the-shelf	1
T1,2	Fine-tuning	2
T1,3	K-fold Cross validation	1
T2	Object Tracking	
T2,1	Tracking by Overlap	2
T2,2	Tracking with a Kalman Filter	2
T2,3	IDF1 for Multiple Object Tracking	1
T3	(optional) CVPR 2021 AI City Challenge	+1

Deliverables



- Deadline: **Wednesday March 30th at 3pm**,
- Deliverables:
  - Submit your report by editing these slides,
  - Provide feedback regarding the teamwork on this evaluation form,