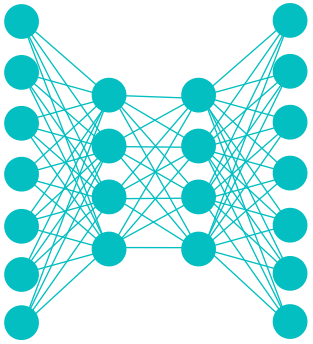
Lecture Notes for

# Neural Networks and Machine Learning

## Fully Convolutional Learning III: Object Tracking



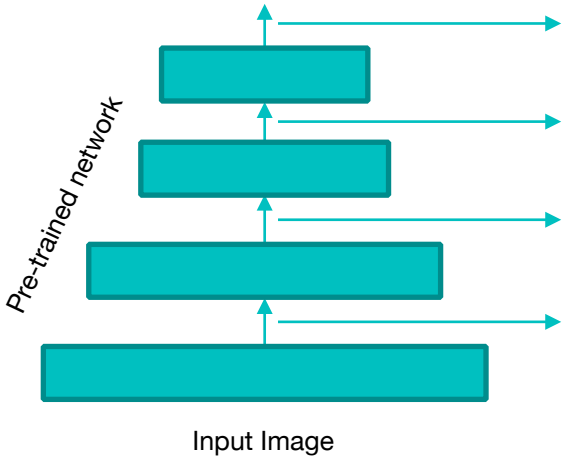
# Logistics and Agenda

- Logistics
  - Lab grading update
  - Next week update
- Agenda
  - Full Convolutional Architectures
    - Object Detection (last time):
      - RCNN, YOLO
    - Tracking from Detection (this time)
    - Instance Segmentation (start today, maybe):
      - Mask-RCNN, YOLACT
      - ViT Segmentation
      - SAM

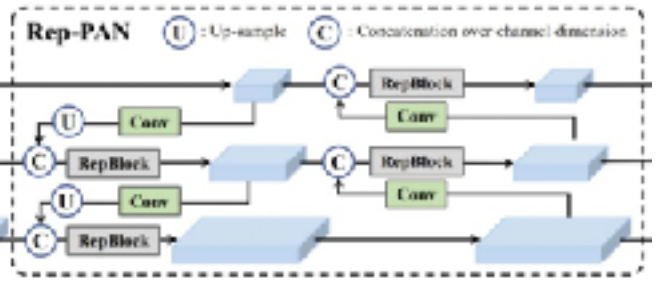


# YOLO Structures (Review)

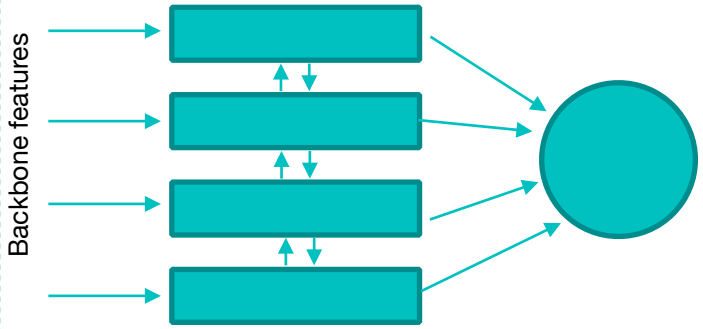
## Backbone



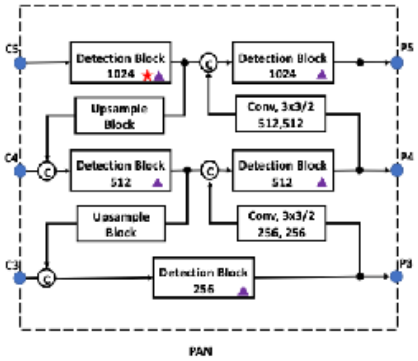

- YOLOv1-v4: Dark/ResNet
- Yv5: EfficientNet
- Yv6: EfficientNet-L2
- PP-Yv1-v2: ResNet50-vd
- Yv6 v3.0: RepVGG 🙄

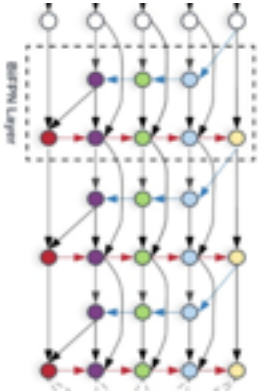



## Neck

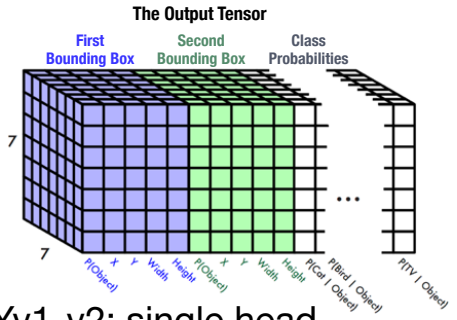



- Yv1-v2: Transfer Learning, Single Layer
- Yv3: Upsampling, Multiple Layers
- Yv4: Cross Stage Partial layers (CSPNet). Use multiple layers from backbone with weighted concat
- PP-Y: Path Aggregation Network
- PP-Yv2: Multi-scale PAN

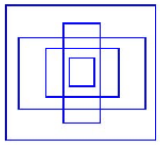


## Head(s)

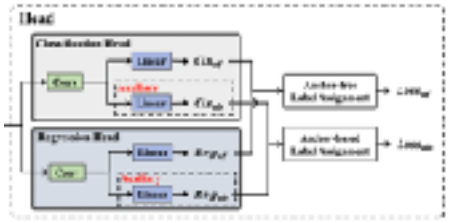



- Yv1-v2: single head
- Yv3: Multi-resolution heads
- Yv4: Clustered anchor boxes and new gradient penalty
- PP-Yv2: IoU prediction loss
- Yv6-v8: Dense Anchor Boxes (anchors as aux. task)
- Yv5-8: Mosaic Augmentation



$$\mathcal{L}_{IoU} = \begin{matrix} -t_{IoU} \cdot \log(\sigma(p)) + \\ (t_{IoU} - 1) \cdot \log(1 - \sigma(p)) \end{matrix}$$

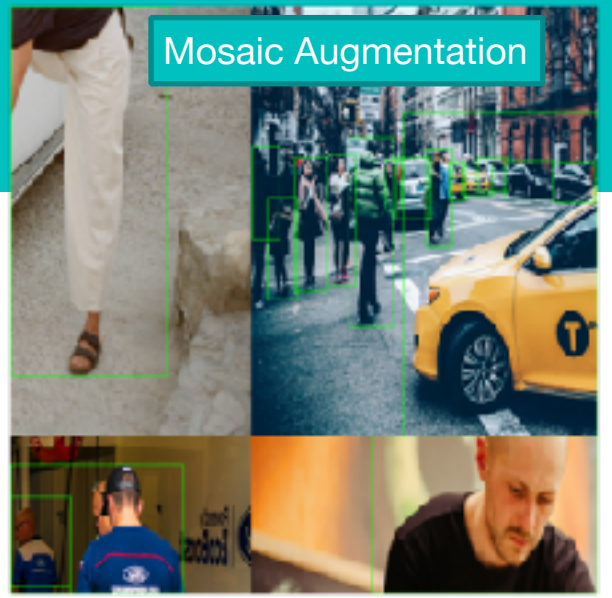
IoU Prediction

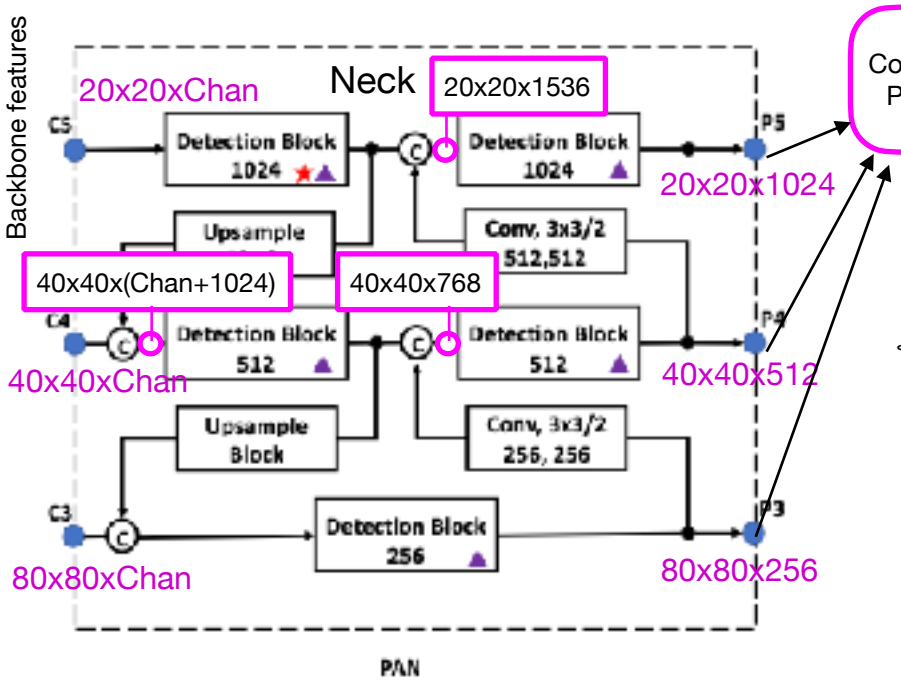


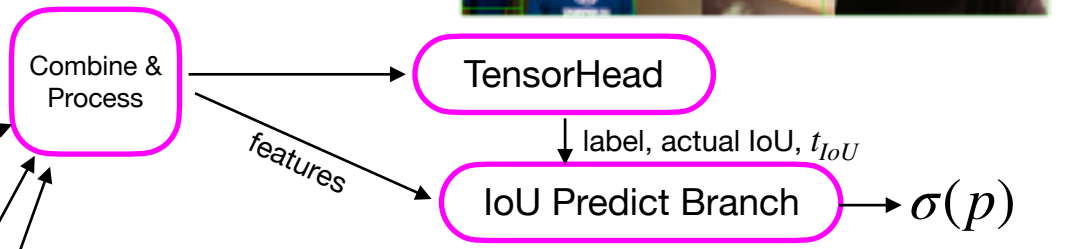


# One Example: PP-YOLOv2

- Path Aggregation Network (PAN) for multi-scale processing
- "IoU Aware Branch" that uses the IoU prediction as another loss




$$\mathcal{L}_{IoU} = -t_{IoU} \cdot \log(\sigma(p)) + (t_{IoU} - 1) \cdot \log(1 - \sigma(p))$$

$t_{IoU}$ : IoU between output of YOLO BBox and GT

$\sigma(p)$=predicted IoU from branch

Main idea is to predict the IoU in a separate branch, which makes the network better at finding the correct bounding boxes

Huang et al., 2021



# PP-YOLOv2: Results

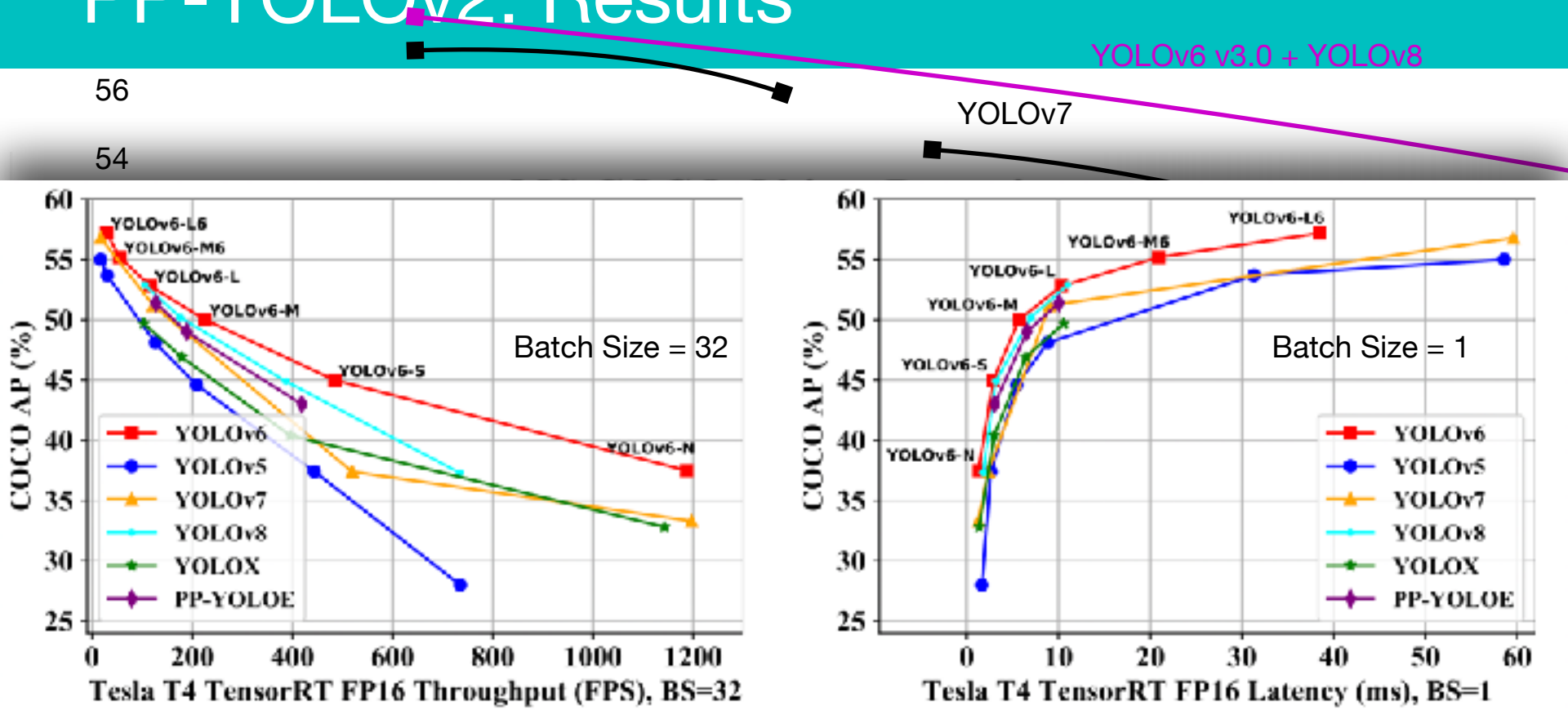
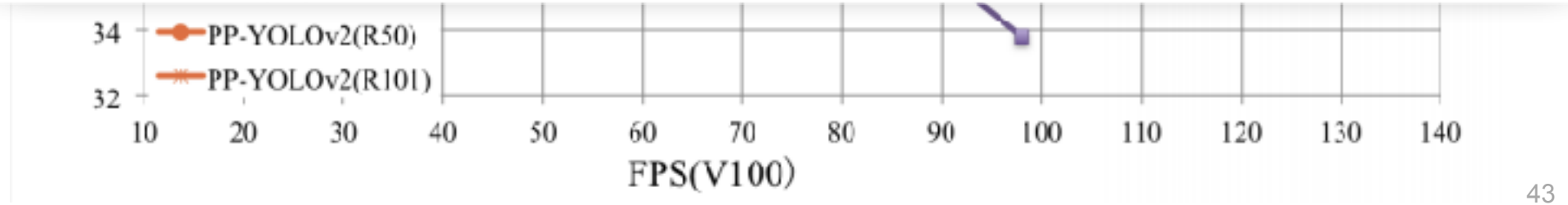



Figure 1: Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7.

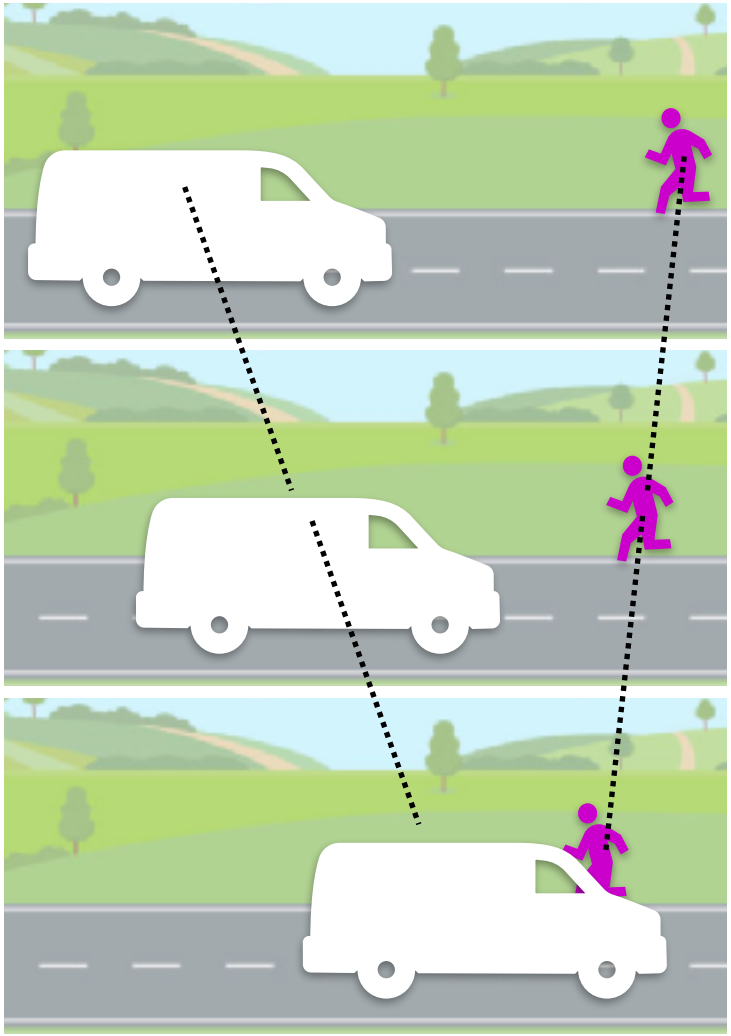





# Tracking From Object Segmentation Models

# Tracking, In General

- Typically separated into two tasks
  - Identify in frame
  - Track across frame
- Hardest aspects:
  - Identifying partially visible objects
  - Tracking same object across occlusion
  - Recognizing objects that leave frame and return
- Traditional Approaches:
  - Classifier followed by Hungarian assignment and Kalman filtering (through occlusion)





# Hungarian Algorithm, Chore Assignment

| | 1. Clean Bathroom | 2. Pickup Rooms | 3. Run Vacuum |
|---|---|---|---|
| Jameson | $15 | $5 | $10 |
| Cecilia | $12 | $4 | $9 |
| Emerson | $10 | $3 | $5 |

**Steps in Hungarian Algorithm:**

- Row reduction (subtract min in row)
- Column reduction (sub. min in col)
- Zero covering (repeated, as needed)
  - 1. Find minimum lines to cover all zeros (assignment)
  - 2. If number of lines does not equal number of needed assignments, perform additional reductions else go to final step.
  - 3. Find min uncovered value, subtract from all elements, add to double crossed elements. Go to step 2.
- Assignment (one zero per row), start from rows with only one zero

**Scratch Work:**

| S1 | 1. CB | 2. PR | 3. RV |
|---|---|---|---|
| J | | | |
| C | | | |
| E | | | |

| S2 | 1. CB | 2. PR | 3. RV |
|---|---|---|---|
| J | | | |
| C | | | |
| E | | | |

| S3 | 1. CB | 2. PR | 3. RV |
|---|---|---|---|
| J | | | |
| C | | | |
| E | | | |

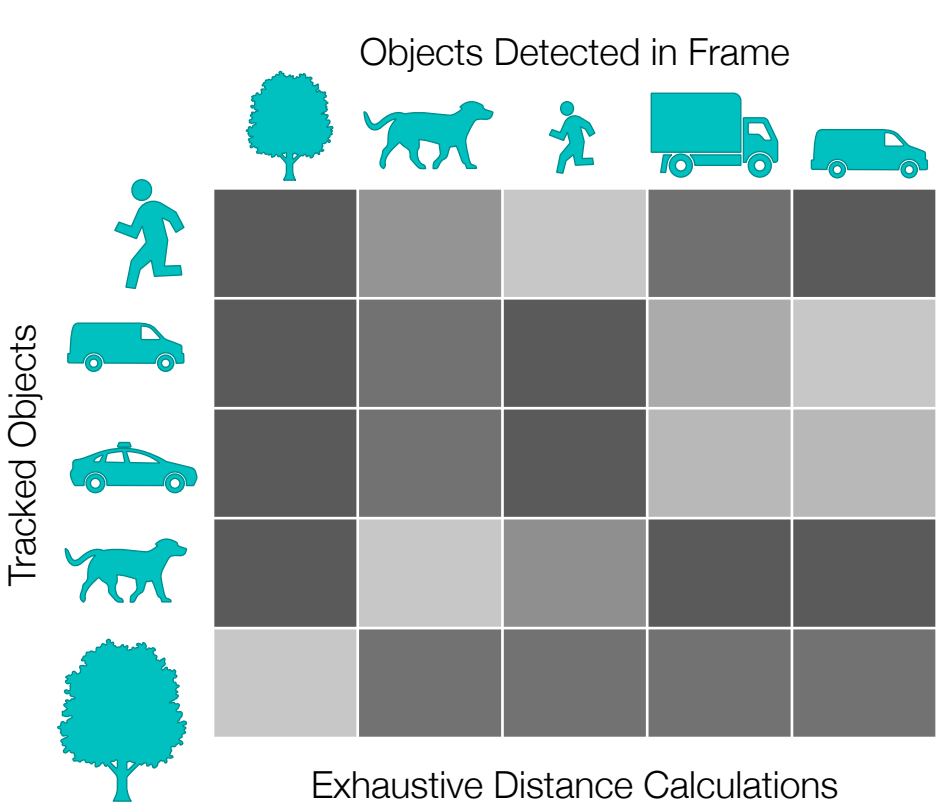
**Exhaustive Search, $O(n!)$**

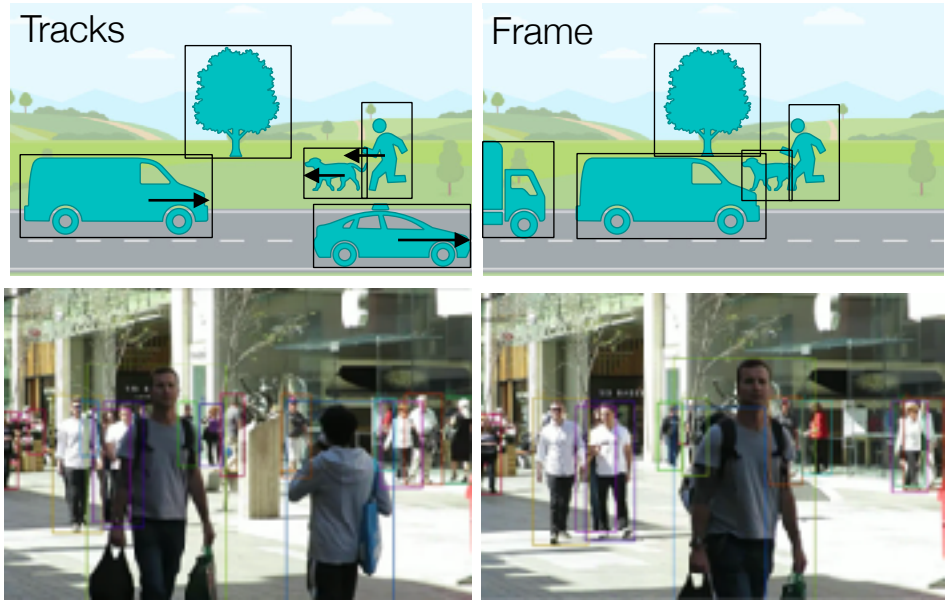
- J-1, C-2, E-3 = 15+4+5 = 24
- J-1, C-3, E-2 = 15+9+3 = 27
- **J-2, C-1, E-3 = 5+12+5 = 22**
- J-2, C-3, E-1 = 5+9+10 = 24
- J-3, C-1, E-2 = 10+12+ 3 = 25
- J-3, C-2, E-1 = 10+4+10 = 24



# Minimum Assignment, Vision

- Minimum Assignment problem: Match detections from to current frame that are “closest” to existing tracks.



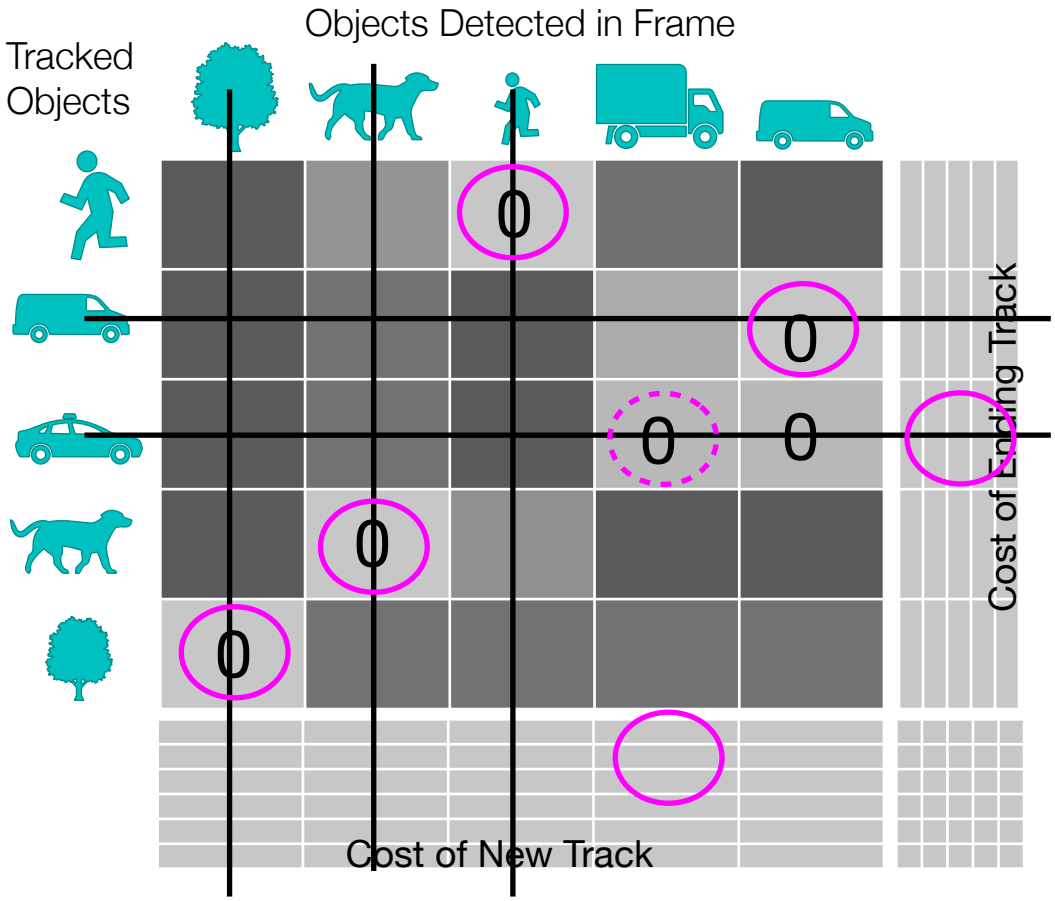

**Common Assignment Cost Metrics:**

- Difference in centers, prev/curr frame
- IoU of objects prev/curr frame
- IoU of “Future Position” (Kalman)
- Appearance (from detector)



# Hungarian Algorithm, in Vision Tracking

Brute force matching is $O(n!)$, looking at every possible assignment and selecting the minimum. *For 5 objects: 120 solutions*



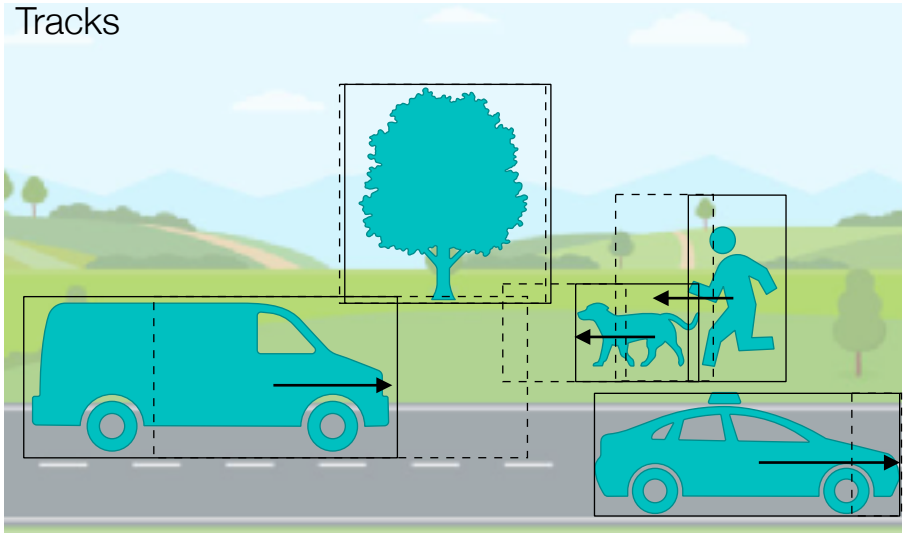

**Steps in Hungarian Algorithm:**

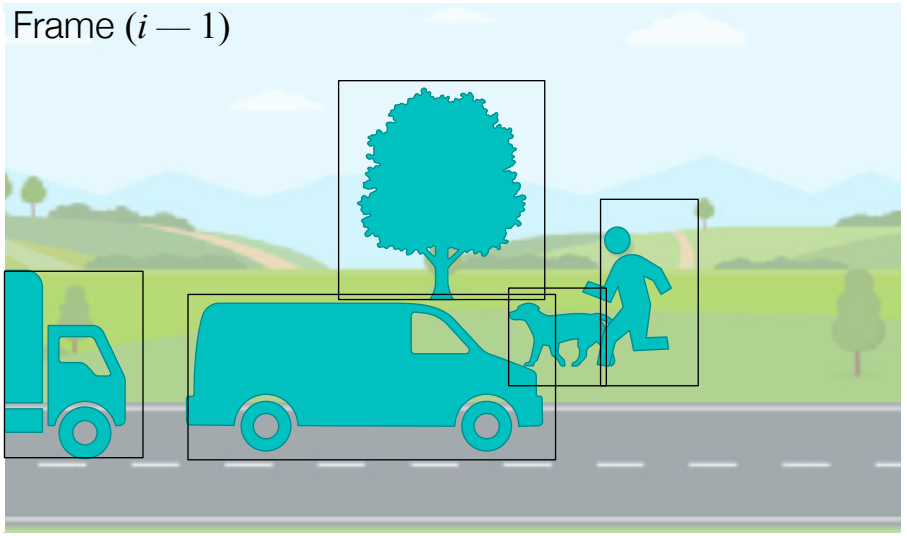
- Row reduction (subtract min in row)
- Column reduction (sub. min in col)
- Zero covering (repeated, as needed)
  - Find minimum lines to cover all zeros (assignment). $O(n^2)$
  - If number of lines does not equal number of needed assignments, perform additional reductions
  - Find min uncovered value, subtract from all elements, add to double crossed elements
- Assignment (one zero per line),
- Look at matching with New Track or storing for later (occluded)

Worst case becomes $O(n^3)$ and gives the optimal assignment!



# Kalman Filtering



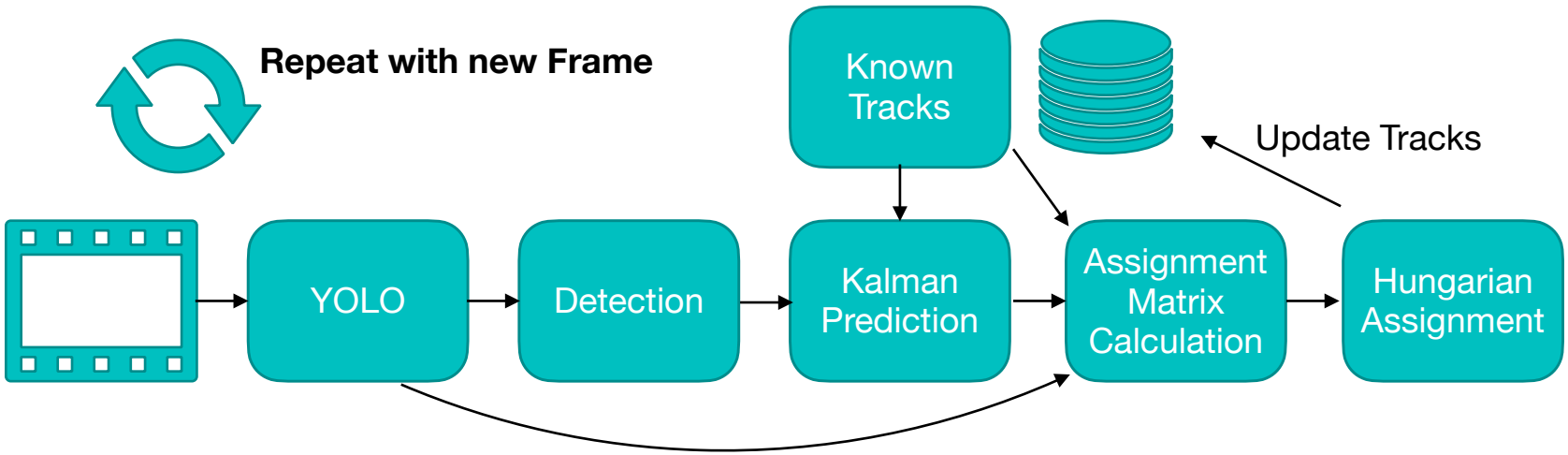




- For each object that has been tracked over multiple frames:
  - Estimate the trajectory of the bounding box center based on estimate of state space for each object (velocity)
  - Kalman Filtering computes filter coefficients in the state space that give speed and direction, *where it should be in next frame*
  - Can be used in computing cost matrix: $IoU\left(X_i^{(frame)}, X_{i-1}^{(kalman)}\right)$



# Tracking with YOLO, Deep-SORT

Simple Online Realtime Tracking




**Potentially Not Optimal:** Cannot Back Propagate through Hungarian Assignment

# Tracking with Transformers: Trackformer

Replace frame to frame assignment and tracking with Transformer. **Encoder** processes the CNN patches and **decoder** takes output and known tracks for assignment.
**Transformer Auto Regressive Output**: Track IDs+Track Features, Per Frame

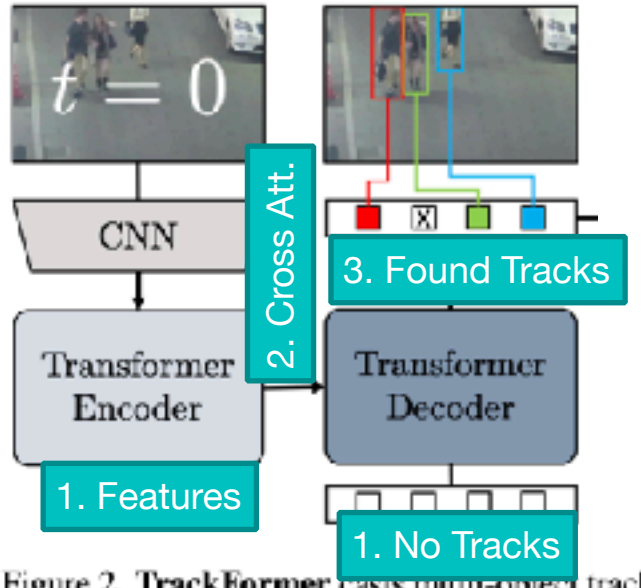
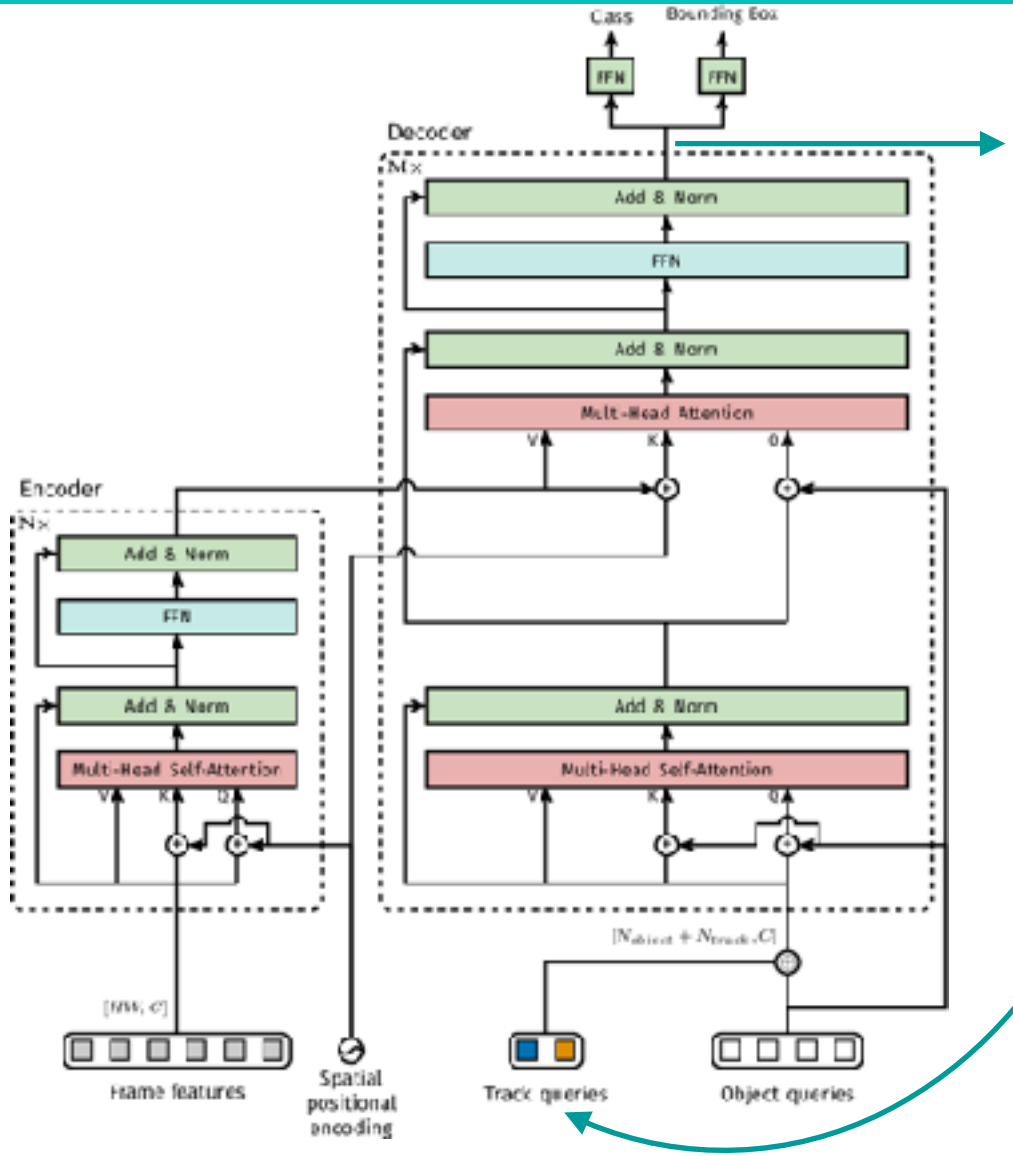



Figure 2. **TrackFormer** casts multi-object tracking as a set pre ion and **tracking-by-attention**. The architecture consists of a CNN for image feature extraction, a Transformer [50] encoder for image feature encoding and a Transformer decoder which applies self- and encoder-decoder attention to produce output embeddings with bounding box and class information. At frame $t = 0$, the decoder transforms $N_{\text{object}}$ object queries (white) to output embeddings either initializing new autoregressive **track queries** or predicting the background class (crossed). On subsequent frames, the decoder processes the joint set of $N_{\text{object}} + N_{\text{track}}$ queries to follow or remove (blue) existing tracks as well as initialize new tracks (purple).

Meinhardt, Tim, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. "Trackformer: Multi-object tracking with transformers." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8844-8854. 2022.



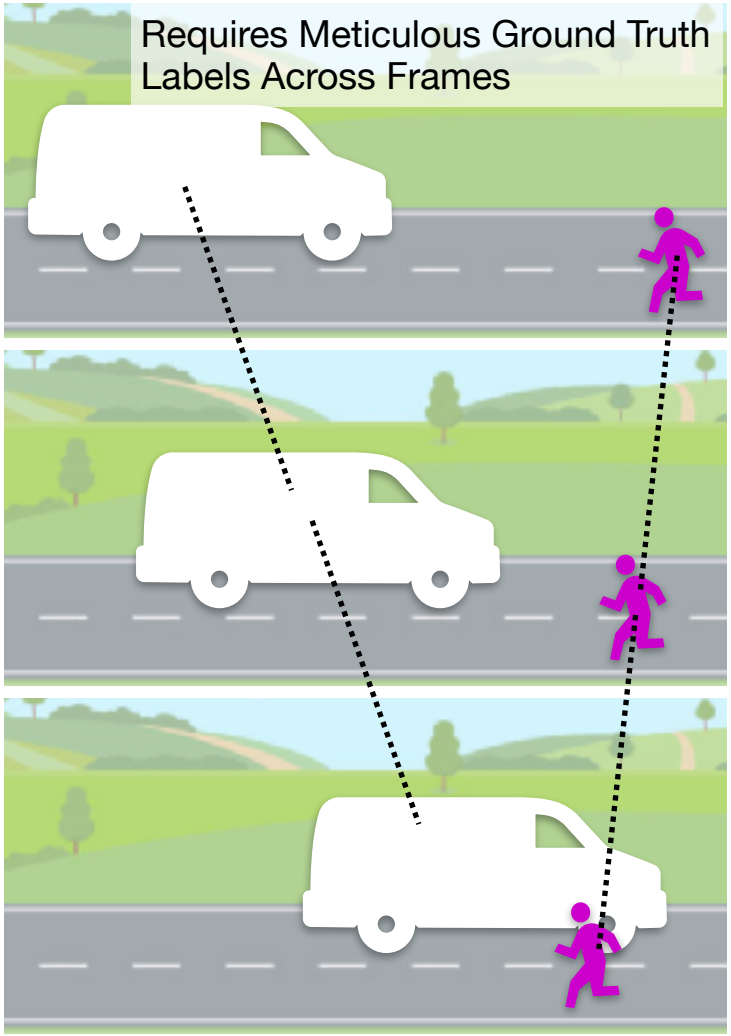
# Trackformer, more Details




Per object features
Saved, become features of track



# Measures of Performance




## Evaluation Measures

Lower is better. Higher is better.

| Measure | Better | Perfect | Description |
|---|---|---|---|
| Avg Rank | lower | 1 | This is the rank of each tracker averaged over all present evaluation measures. |
| MOTA | higher | 100 % | Multiple Object Tracking Accuracy [1]. This measure combines three error sources: false positives, missed targets and identity switches. |
| MOTP | higher | 100 % | Multiple Object Tracking Precision [1]. The misalignment between the annotated and the predicted bounding boxes. |
| IDF1 | higher | 100 % | ID F1 Score [2]. The ratio of correctly identified detections over the average number of ground-truth and computed detections. |
| FAF | lower | 0 | The average number of false alarms per frame. |
| MT | higher | 100 % | Mostly tracked targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at least 80% of their respective life span. |
| ML | lower | 0 % | Mostly lost targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at most 20% of their respective life span. |
| FP | lower | 0 | The total number of false positives. |
| FN | lower | 0 | The total number of false negatives (missed targets). |
| ID Sw. | lower | 0 | The total number of identity switches. Please note that we follow the stricter definition of identity switches as described in [3]. |
| Frag | lower | 0 | The total number of times a trajectory is fragmented (i.e. interrupted during tracking). |
| Hz | higher | Inf. | Processing speed (in frames per second excluding the detector) on the benchmark. |

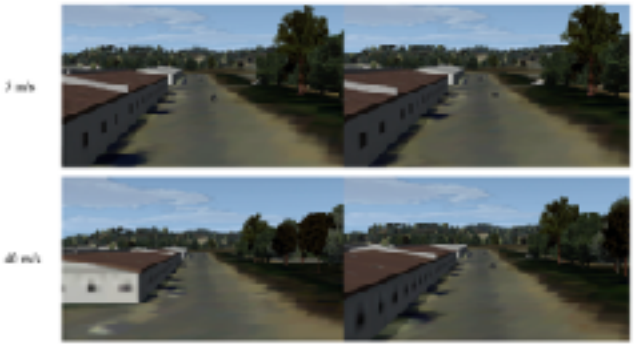
https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort/



# Evaluating Tracking with Transformer

Domain Mismatch

Source Movement



Distance

25 meters 300 meters 750 meters

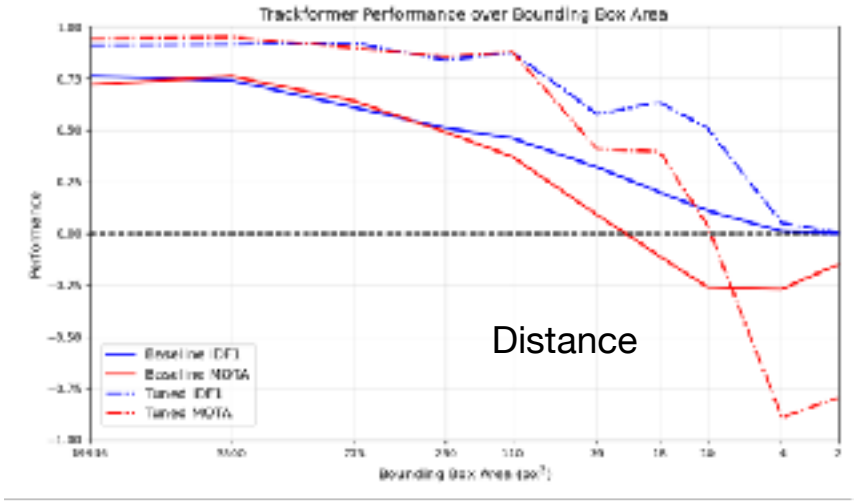
Lighting

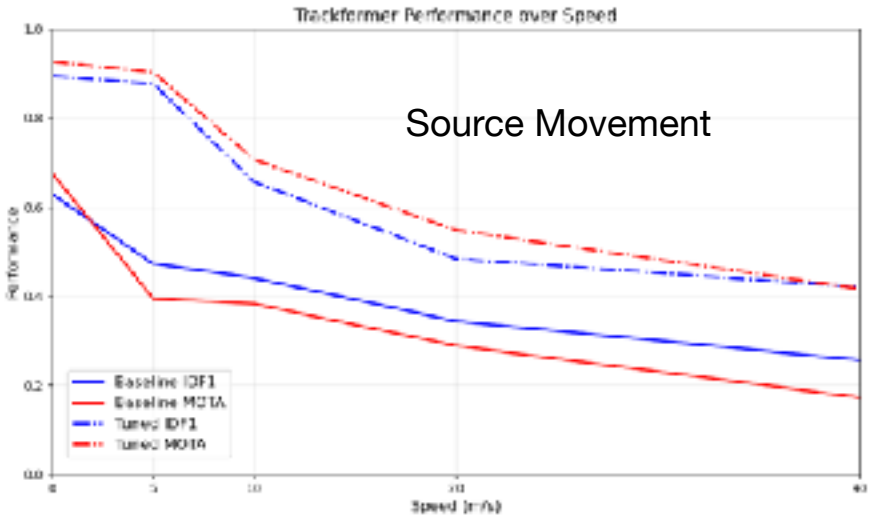
Impacts of Synthetically Generated Data on Trackformer-based Multi-Object Tracking

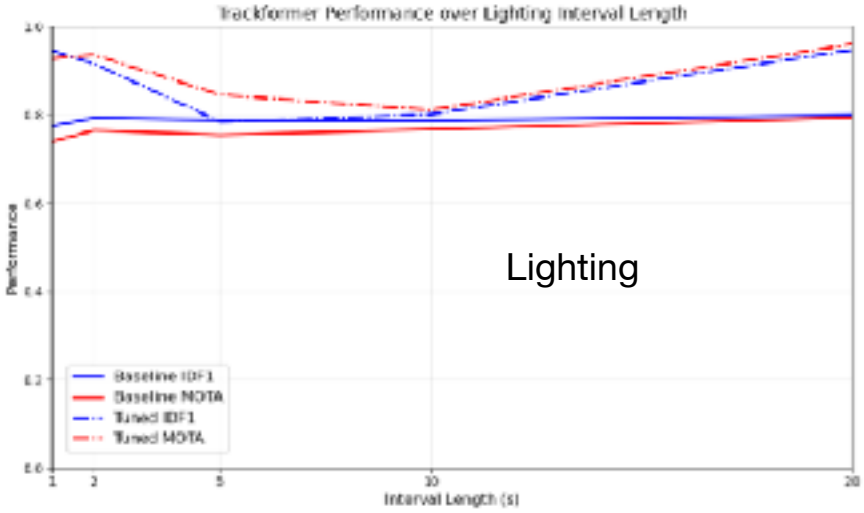
Matthew Lee, Clayton Harper, William Flinchbaugh, Eric C. Larson, and Mitchell A. Thornton



# Evaluating Tracking with Transformer




**Impacts of Synthetically Generated Data on Trackformer-based Multi-Object Tracking**

Matthew Lee, Clayton Harper, William Flinchbaugh,
Eric C. Larson, and Mitchell A. Thornton



# A closing thought from YOLOv3 Report


**Abstract**

*We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 $AP_{50}$ in 51 ms on a Titan X, compared to 57.5 $AP_{50}$ in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at* https://pjreddie.com/yolo/.


But maybe a better question is: "What are we going to do with these detectors now that we have them?" A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won't be used to harvest your personal information and sell it to.... wait, you're saying that's exactly what it will be used for?? Oh.

Well the other people heavily funding vision research are the military and they've never done anything horrible like killing lots of people with new technology oh wait.....[1]

I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

In closing, do not @ me. (Because I finally quit Twitter).

[1] The author is funded by the Office of Naval Research and Google.

Redmon and Farhadi, YOLOv3: An incremental Improvement



# A closing thought from YOLOv3 Report

## The Rebuttal I wish I could Write:

Reviewer #2 AKA Dan Grossman (lol blinding who does that) insists that I point out here that our graphs have not one but two non-zero origins. You're absolutely right Dan, that's because it looks way better than admitting to ourselves that we're all just here battling over 2-3% mAP. But here are the requested graphs. I threw in one with FPS too because we look just like super good when we plot on FPS.

Reviewer #4 AKA JudasAdventus on Reddit writes "Entertaining read but the arguments against the MSCOCO metrics seem a bit weak". Well, I always knew you would be the one to turn on me Judas. You know how when you work on a project and it only comes out alright so you have to figure out some way to justify how what you did actually was pretty cool? I was basically trying to do that and I lashed out at the COCO metrics a little bit. But now that I've staked out this hill I may as well die on it.

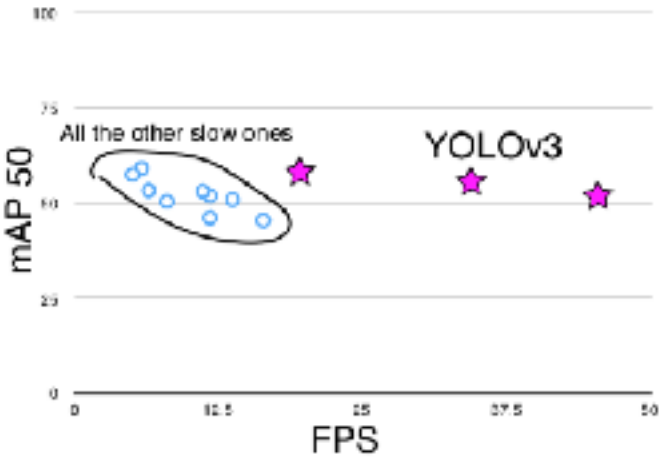




Figure 5. These two hypothetical detectors are perfect according to mAP over these two images. They are both perfect. Totally equal.

Now this is OBVIOUSLY an over-exaggeration of the problems with mAP but I guess my newly reteconned point is that there are such obvious discrepancies between what people in the "real world" would care about and our current metrics that I think if we're going to come up with new metrics we should focus on these discrepancies. Also, like, it's already mean average precision, what do we even call the COCO metric, average mean average precision?

Here's a proposal, what people actually care about is given an image and a detector, how well will the detector find and classify objects in the image. What about getting rid of the per-class AP and just doing a global average precision? Or doing an AP calcu-

Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them.



# YOLACT, one pass mask generation

**YOLACT**
**Real-time Instance Segmentation**

Daniel Bolya Chong Zhou Fanyi Xiao Yong Jae Lee
University of California, Davis
{dbolya, czhou, fyxiao, yongjaelee}@ucdavis.edu

**Abstract**

*We present a simple, fully-convolutional model for real-time instance segmentation that achieves 29.8 mAP on MS COCO at 33.5 fps evaluated on a single Titan Xp, which is significantly faster than any previous competitive approach. Moreover, we obtain this result after training on **only one GPU**. We accomplish this by breaking instance segmentation into two parallel subtasks: (1) generating a set of prototype masks and (2) predicting per-instance mask coefficients. Then we produce instance masks by linearly combining the prototypes with the mask coefficients. We find that because this process doesn't depend on repooling, this approach produces very high-quality masks and exhibits temporal stability for free. Furthermore, we analyze the emergent behavior of our prototypes and show they learn to localize instances on their own in a translation variant manner, despite being fully-convolutional. Finally, we also propose Fast NMS, a drop-in 12 ms faster replacement for standard NMS that only has a marginal performance penalty.*

**1. Introduction**

*"Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them."*
– Joseph Redmon, YOLOv3 [36]

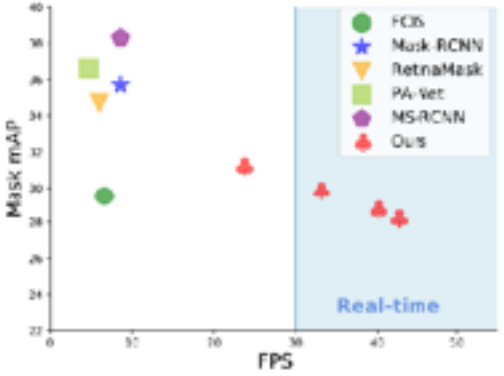



Figure 1: Speed-performance trade-off for various instance segmentation methods on COCO. To our knowledge, ours is the first *real-time* (above 30 FPS) approach with around 30 mask mAP on COCO `test-dev`.

However, instance segmentation is hard—much harder than object detection. One-stage object detectors like SSD and YOLO are able to speed up existing two-stage detectors like Faster R-CNN by simply removing the second stage and making up for the lost performance in other ways. The same approach is not easily extendable, however, to instance segmentation. State-of-the-art two-stage

Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "YOLACT: real-time instance segmentation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9157-9166. October 2019.



Lecture Notes for

# Neural Networks and Machine Learning

## FCN Learning: Detection

**Next Time:**
Instance Segmentation

**Reading:** None