REINFORCING GOOD DECISIONS FOR GLOBAL MULTIPLE-OBJECT TRACKING

by

Taher Anjary

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

June 15, 2023

REINFORCING GOOD DECISIONS FOR GLOBAL MULTIPLE-OBJECT TRACKING

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Taher Anjary

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Dr. Fatma Güney (Advisor)

Date: _____

[Dedication here (Optional)]

ABSTRACT

REINFORCING GOOD DECISIONS FOR GLOBAL MULTIPLE-OBJECT TRACKING

Taher Anjary

Master of Science in Computer Science and Engineering

June 15, 2023

Multiple Object Tracking (MOT) involves the challenge of determining the paths followed by individual objects within a video. Recent advances in MOT achieve tracking using the tracking-by-detection paradigm. The challenge is in resolving ambiguous correspondences between detected objects across the temporal axis. This has lead to learning to extract and utilize more discriminative features. Such distinguishing cues can be extracted either visually, geometrically, spatio-temporally or from velocity. Most relevant to our work are approaches that utilize the global-reasoning capacity of Graph-Neural-Networks (GNNs). However, many networks that take this approach overlook the high number of False Negatives that hinder the tracking performance. In our work, we start with an abundant set of detections. The abundance of detections capture as many False Negatives as possible, but also increases the number of False Positives. We formulate our method with REINFORCE, and a task-specific reward formulation to reason through the vast combinatorial space and find the optimal trajectories. Taking inspiration from two-stage-object-detectors, we develop a *Track-of-Interest* head to generate track-proposals via a learnable random walk sampler, allowing the network to present the best set of tracks. We show that using a proposal-based approach in this way allows the agent to leverage the global scope of data available to GNNs rather than local, pairwise relations between detected objects. We show an improvement in commonly used tracking metrics on the MOT17 dataset. Our code will be made publicly available at `https://github.com/tanjary21/ReiTracker`.

ÖZETÇE

Yüksek Lisans Tez Başlığı
Taher Anjary
Bilgisayer Mühendisliği, Yüksek Lisans
15 Haziran 2023

Çoklu Nesne Takibi (MOT), bir video içindeki tek tek nesnelerin izlediği yolları belirleme zorluğunu içerir. MOT'daki son gelişmeler, tespit yoluyla izleme paradigmasını kullanarak izlemeyi mümkün kılıyor. Buradaki zorluk, zamansal eksen boyunca tespit edilen nesneler arasındaki belirsiz yazışmaları çözmektir. Bu, daha ayırt edici özelliklerin çıkarılması ve kullanılmasının öğrenilmesine yol açmıştır. Bu tür ayırt edici ipuçları görsel, geometrik, uzay-zamansal veya hızdan elde edilebilir. Çalışmamızla en alakalı olan, Grafik-Sinir Ağlarının (GNN'ler) küresel akıl yürütme kapasitesini kullanan yaklaşımlardır. Ancak bu yaklaşımı benimseyen birçok ağ, izleme performansını engelleyen çok sayıda Yanlış Negatifi gözden kaçırır. Çalışmamıza bol miktarda tespitle başlıyoruz. Tespitlerin çokluğu mümkün olduğu kadar çok sayıda Yanlış Negatif yakalar ancak aynı zamanda Yanlış Pozitiflerin sayısını da artırır. Yöntemimizi REINFORCE ve göreve özel bir ödül formülasyonu ile formüle ederek geniş kombinatoryal alanda akıl yürütüp en uygun yörüngeleri buluyoruz. İki aşamalı nesne algılayıcılardan ilham alarak, öğrenilebilir bir rastgele yürüyüş örnekleyici aracılığıyla izleme önerileri oluşturmak için bir *İlgi Alanı* başlığı geliştiriyoruz ve ağın en iyi izleme dizisini sunmasına olanak sağlıyoruz. Öneriye dayalı bir yaklaşımın bu şekilde kullanılmasının, aracının, tespit edilen nesneler arasındaki yerel, ikili ilişkilerden ziyade, GNN'ler için mevcut olan küresel veri kapsamından yararlanmasına olanak tanıdığını gösterdik. MOT17 veri kümesinde yaygın olarak kullanılan izleme ölçümlerinde bir gelişme olduğunu gösteriyoruz. Kodumuz şu adreste kamuya açıklanacaktır: `https://github.com/tanjary21/ReiTracker`

ACKNOWLEDGMENTS

Acknowledgements here

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

MLE	Maximum Likelihood Estimation
MOT	Multiple-Object Tracking
MOTA	Multiple-Object Tracking Accuracy
FP	False Positive
FN	False Negative
IDSw	Identity Switches
GNN	Graph Neural Network
GCN	Graph Convolutional Network
MPN	Message Passing Network
CNN	Convolutional Neural Network
ReID	Person Re-Identification
RoI	Region of Interest
ToI	Track of Interest

Chapter 1

INTRODUCTION

Multiple-Object-Tracking(MOT) is a key problem in Computer Vision. Solving it means that one can look at a video sequence, uniquely identify all the objects that exist in it and maintain each of the identified objects as they evolve through the video sequence. We focus only on tracking pedestrians in our work. MOT is a mid-level task within Computer Vision, but it provides a foundation from which higher-level tasks can be achieved. These tasks include practical applications such as virtual reality, pose estimation, human-computer interaction, action recognition, behaviour analysis, visual surveillance[Luo et al., 2021], autonomous driving[Chang et al., 2019, Wilson et al., 2021, Lambert and Hays, 2021].

MOT literature would show the following intuitive way of accomplishing MOT: using ever-improving object-detectors trained on large pedestrian datasets, one can extract features of objects detected at each frame, and use a similarity measure to find the corresponding objects in adjacent frames. Determining the correspondences i.e. deciding which detections from one frame associate to which detection in a subsequent frame, would mean an object has been successfully tracked from one frame to the next, and this can continue for several frames. Approaching MOT in this way, where one first detect objects independently at each frame and then associates them at a later step, is known as the *tracking-by-detection* paradigm.

1.1 Common Errors in Tracking

Tracking-by-detection works in ideal scenarios, where the objects in the video are always, clearly visible by a sensor, and are correctly detected with precise bounding boxes. Then, by computing a euclidean distance between each detected object in one

frame and and that in a subsequent frame, one could construct a distance matrix, and use a Hungarian algorithm to optimally assign a strict one-to-one matching. This is equivalent to optimally(within the local context) associating objects based on which object from a subsequent frame is closest to which object in the current frame. Repeating this for the entire video would give you perfect trajectories. However, in real world scenes;

- crowds may walk in all directions on a busy street
- crowds occlude each others' visibility to a camera
- crowds move with inconsistent speeds and directions
- other objects that my be mistaken for a human figure such as statues, reflections, shrubs and motorcyclists may exist
- the video may be captured at low frame rates

With such real-world scenarios, MOT becomes a significantly more complex optimization problem to solve.

When an object that needs to be detected is not visible by a camera, the detector may not be able to detect it. This is termed as a *False Negative (FN)*. FN's are a critical problem in Autonomous Driving because it is unacceptable to not detect an object that appears in front of a moving vehicle, especially if it is human. To alleviate this, object detectors are being trained on extensive datasets, constantly pushing the state-of-the-art in recall.

In an effort to capture FNs however, one may create overly-cautious detectors to

- predict objects that resemble pedestrians
- predict empty objects near the edge of a wall in anticipation of a child running onto the street
- predict multiple objects in the vicinity of one object in anticipation of occluded objects behind it

Such detections are termed as *False Positives (FP)*. This warrants the need for an algorithm such as *Non-Maximal Supression (NMS)*. NMS discards predictions that are below a certain confidence threshold and have significant overlap with other, more confident predictions. This is robust in that in the attempt to capture FNs, FPs increase, but can be discarded intelligently. Leaving high recall, high precision, hence highly accurate detections that can be tracked.

1.2 Challenges in Tracking

To track the given detections, one may intialize a track with each detection's bounding box, then compute euclidean distances to the bounding boxes of detections from the next frame, and perform one-to-one matching, then update the track with the new bounding box location. This can be repeated over subsequent frames.

Such association, where the criterion for computing distances between detections is based on geometric similarity in location, scale and time, are termed *geometric*, or *spatio-temporal* association cues and can be measured via *Intersection over Union (IoU)*. Spatio-temporal cues are an efficient distance criterion as they are cheap and easy to compute, especially in real time, but can fail when the frame rate is low and motion is fast. This is because spatio-temporal cues assume that an object will move only incrementally, given high enough frame rates, such that their IoU remains high, and warrants a confident association. This is unfortunately an unreliable assumption as motion may be high when the camera is also moving, frames rates may be low due to processing and bandwidth constraints.

Previous works exist to correct and account for camera motion and frame rate but they are not robust enough to adapt to a wide range of velocities. Furthermore, the trajectory is terminated if at any point a match cannot be made, which may happen due to temporary occlusions or FNs. Termination is necessary for scenarios such as a target leaving the field of view, but is undesirable if the target is only momentarily occluded and re-appears later. Such a tracking error is called a *Fragmentation (FM)*. Previous works address this by using patience-windows or buffers, such that unmatched trajectories may be stored in anticipation for re-association, if they re-

emerge within a specified number of frames, and then the trajectory can be re-activated. However, if the motion suddenly changes during the occlusion, Iou may not be a robust enough cue, especially in crowded scenes where there may be several possible candidates for the re-association.

To address ambiguous associations from spatio-temporal distances, *Person Re-Identification (ReID)*[Ristani and Tomasi, 2018] was found to be a robust cue. ReID[Ristani and Tomasi, 2018] aims to produce a normalized similarity score for two detections, based on feature vectors computed from their visual appearance i.e. by encoding the pixels within the bounding box. This allowed associations to be made effortlessly regardless of how long the target is occluded for. However, several targets may dress similarly, resulting in similar features. Highly occluded targets, even if detected, would contain similar pixels to the occluder, resulting in similar features. This made *visual* association cues prone to ambiguity as well, and a cosine distance between two candidate matching features would no longer suffice as the distinguishing cue.

A common practice in real-time MOT was to use Kalman filters[Bewley et al., 2016, Yu et al., 2016] to predict the next spatial position of a trajectory, which proved useful as not just a cue, but also to continue propagating a trajectory through an occlusion. This demanded that, upon re-emergence, a propagated trajectory would have to have not only a high IoU with the re-emergent detection, but also a high visual similarity. The issue however, was that several changes may occur under the occlusion: the trajectory may change their velocity, which affected their appearance, position and scale.

Since several trackers assumed linear motion, non-linear Kalman filters and/or learnable RNN/LSTM based motion models were explored. This also allowed the prediction of multiple possible future locations. Further advances noted that graphs allowed interactions between targets to be considered prior to making associations. Real-time trackers conjectured that the interactions could be the cause of non-linear motion, akin to charged particles attracting/repelling each other when in each others' vicinity, hence altering their velocities. Graphs naturally embedded these interactions, so real-time trackers modelled interactions by including hidden embeddings

from neighboring tracks as additional input cues to their motion predictors. But graphs also naturally allowed visual cues to undergo interaction, recognizing that an occluded trajectory would have high visual similarity to the occluding trajectory, and scrutinized visual embeddings could be learned via several graph-conv iterations. This corresponded to stronger, more discriminative visual embeddings to be learned in real-time trackers via contrastive learning methods, using nearby targets as contrastive samples to strengthen the discriminatability in the learned embedding space.

1.3 Graph Based Solutions

Meanwhile, graphs were gaining popularity in tracking because they allowed a global view of the scene in both spatial and temporal dimensions; all detections from the past and future can be simultaneously considered before making association decisions. A track that is temporarily occluded could immediately re-associate to its re-appearance ten time-steps in the future rather than greedily associating to the next best object in the next time-step. While offline-graph methods, by design, cannot operate in real-time tracking, they proved more robust in offline visual analysis tasks[Brasó and Leal-Taixé, 2020]. Furthermore, graph trackers provided clues as to how real-time trackers could be improved; by utilizing buffers/patience windows to mitigate greedy decisions, and more recently, segregating the association process into two rounds; associate high confidence objects first, then associate the low confidence and leftover objects. The buffers/patience-windows would allow real-time trackers to track through occlusions, alleviating IDSw and FMs, while two-round associations would allow capturing more FNs and carefully discarding FPs.

Graphs also showed that utilizing both visual and spatio-temporal cues was not only possible, but allowed synergistic advantages: visual similarities could be used when spatio-temporal cues became uninformative such as during temporary occlusions, and spatio-temporal cues could be used when visual similarities were ambiguous such as when two targets dressed the same. The disadvantage of offline graphs are that they cannot run in real-time, and that the increased, global scope created

a larger search space, hence increasing the label imbalance and making them susceptible to tracking errors from more ambiguities. Such were the ambiguities with graph trackers that violations were occurring; detections could associate to multiple other detections, implying that the trajectory split into multiple futures, and vice versa is true for the past. Hence, flow constraints and linear solvers had to be adapted to resolve conflicts/violations, but they were non-differentiable and simplified the constraints only on the associated decisions(edges) rather than also the detections(nodes). For both graphs trackers and real-time trackers, more discriminative visual and spatio-temporal cues were necessary.

1.4 Our Proposed Method

Our work hence began with MPN[Brasó and Leal-Taixé, 2020], a graph-tracker that utilized preprocessed detections from Traktor[Bergmann et al., 2019] to improve detection recall. We chose this approach due to the ability of graph-trackers to consider information from a global scope of the scene before making association decisions, consequently eliminating the need for hard-coded processes such as buffers/patience-windows to handle occlusions.

The popular observation in tracking-by-detection is that the quality of detections is a strong factor in determining the quality of the tracker. Furthermore, we conjecture that detection could benefit from tracking cues; an FP detection could be discarded not based on its confidence but on whether a visually, spatio-temporally similar detection exists in adjacent frames. Such is our motivation for utilizing unsuppressed detections; discard them only after all information has been considered, such that recall and precision can be optimized.

Upon closer investigation however, we found that quality detections are not easy to obtain on unseen data, implying overfitting. Furthermore, we found that state-of-the-art trackers on the benchmark had few FPs but struggled to capture FNs, implying a very cautious detector. Hence, we utilize a dense set of raw, unprocessed detections from QDTrack[Pang et al., 2021]. We chose QDTrack due to its power to exploit local scopes to not just generate high-quality detections, but also several con-

trastive targets for contrastively learning a discriminative embedding space to substitute ReID[Ristani and Tomasi, 2018]. The approach involves keeping all Regions-of-Interest(RoIs), prior to being filtered via Non-Maximum-Suppression(NMS) such that our detection set captures the majority FNs, but necessarily results in a surge of FPs. Nevertheless, we found the dense detection set significantly increases the oracle performance of our tracker; an upper bound on the accuracy that our tracker has the potential to obtain if it made all the right decisions with the given data. The tracker simply has to learn to identify and avoid tracking FPs. We found this challenging since the increased FNs comes at the cost of significantly more FPs that the tracker has to reason through.

MPN utilized a simplified flow-solver that only constrained the associations; a detection-node can have at most one edge(association) going in and one edge going out. While this resolved the violations, it assumed that the node is a TP. This was sufficient for MPN since they used cautious detections that contained a fairly low number of FPs. In the case of dense detections however, where FPs are plentiful, we found the solver to be susceptible to incorrect associations, since the solver's design assumed that every detection-node is a TP. Therefore, we extend MPN to predict not just probabilities on the edges, but also probabilities on the nodes. We design the solver such that both edge and node probabilities are used as part of the costs and constraints in the solver; if a detection-node has a low probability(predicted to be an FP), then no edges should go in or out of it, whereas if the detection-node has a high probability(predicted to be an TP), at most one edge can go in and out of it.

While our inclusion of additional, more flexible constraints on the solver were reasonable, the performance was not; the FPs were too many for the solver to resolve, the class imbalance was too high. Hence, we adopt focal loss to weigh and balance the supervision signals according to the positive-negative sample ratio. This helps slightly, but proves insufficient.

The literature showed that two-stage detectors such as Faster-RCNN[Ren et al., 2015] resulted in higher detection recall due to its innovative, trainable Region-of-Interest(RoI) head; it proposes an abundance of possible detection locations which

captured several FNs, to be filtered by NMS in an attempt to reduce the subsequent surge in FPs. Hence, we conjecture that a similar approach in tracking would prove advantageous. Hence, we create a *Track-of-Interest(ToI)* head to extract such track-proposals from within a window of T frames. We design the ToI head to enumerate every possible non-violating tracklet on the graph as a proposed tracklet of interest. The process is exhaustive in the combinatorial space, as is the RoI head in the spatial activation map. Furthermore, each tracklet is in itself non-violating; each tracklet has no more than one edge going in and out of a node. Similar to how RoI heads generated several overlapping bounding-boxes as possible detected objects, ToIs generated tracklets that had nodes that were common with other tracklets. This prompted a form of NMS to be designed for ToIs; so we adapted the de-overlapping algorithm from [Yang et al., 2019b, Dai et al., 2021] and optimized for GPU processing. The result would give a set of tracklets that did not conflict with each other; a node cannot participate in more than one ToI. The Track NMS also allowed final trajectories to be generated easily -via sliding window merging with a Track-IoU[Cetintas et al., 2022] criterion- without the need of a flow-solver to resolve conflicts. What remained was a way to utilize the ToIs as training samples.

Task-specific supervision such as with rewards and Reinforcement Learning(RL) algorithms are known to be effective at converging to an optimal solution when direct supervision on individual steps becomes not just impractical but also unknowable; a decision made in the past that led to an eventual success of an episode may lead to failure in other episodes due to varying scenarios; the search space for the optimal solution is not only vast, but can also be considered a combinatorial problem. For search spaces where several unfavorable states exist, exploration and variety encourages discovery and focus on a large portion of the rare favorable states[ben,].

Trackers that utilize RL methods rely on modelling multiple single-object-trackers(SOT) as agents, each responsible for tracking its target via a *Markov-Decision-Process(MDP)*. However, MDP approaches tend to exploit a good state once it is discovered, leading to possibly unexpected and undesirable behaviour[ben,]. Instead, we design a high-level, task-specific reward which encourages longer and purer tracklets, providing supervision to guide learning towards a high-level tracking goal rather than

directly supervising each decision. This complements our offline graph-tracker approach while avoiding the MDP model. Furthermore, the exhaustive enumeration of possible tracklets on a single graph provides the diverse set of samples required to explore rare favorable solutions that may exist in the large combinatorial space of unfavorable ones plagued by FPs.

Our contributions are hence 4-fold:

- exploiting the global scope of GNNs to make more informed tracking decisions
- exploiting the power of local scopes of real-time trackers to generate an abundant set of detections that are likely to capture FNs, and enabling tracking to benefit detection by following a disjoint tracking-by-detection paradigm to eliminate FPs
- a Minimum-Cost-Flow(MCF) solver adapted to provide additional constraints to tackle the surge of FPs in the data.
- a REINFORCE formulation that encourages tracking from proposals such that a high-level tracking-specific reward is maximized, while retiring the need for flow solvers and reasoning through a large combinatorial space of ubiquitous FPs.

Chapter 2

BACKGROUND

2.1 Deep Learning on Graphs

When performing 2D Convolution on image data, a central pixel in the subsequent activation layer is a weighted average of the central pixel in the in current activation layer and the pixels in the kernel's receptive field, where the weights are embedded as learnable parameters in the kernel. 2D Convolution is however only convenient on input that has regular, grid-like structure. Furthermore, it would not be possible for a pixel on one end of the image to include a pixel from the other end of the image, without sufficient layers. Such problems with irregular data exist in the fields of biochemical research, recommender systems, social networks, natural language and computer vision.

Applications. In computer vision, dynamic scenes such as objects moving in a video or irregular data such as point clouds can be represented as graphs [Wu et al., 2020]. The ability to recognize interactions between objects and represent the scene's semantic relationships [Xu et al., 2017, Yang et al., 2018, Li et al., 2018a] becomes critical to solve a task such as tracking on video data from a camera or segmenting point clouds from a 3D LiDAR sensor [Wang et al., 2019a, Landrieu and Simonovsky, 2018, Te et al., 2018]. Other vision tasks being tackled using GNNs include visual reasoning[Chen et al., 2018], semantic segmentation [Qi et al., 2017, Yi et al., 2017], few-shot image classification [Garcia and Bruna, 2017, Guo et al., 2018], question answering[Narasimhan et al., 2018] and human-object interaction[Qi et al., 2018].The syntactic dependency between words in a sentence from Natural Language can also be modelled on graph structures [Marcheggiani and Titov, 2017, Marcheggiani et al., 2018] and can then be used with GNNs to solve NLP tasks such as Neural Machine Translation [Bastings et al., 2017], or document classification [Kipf

and Welling, 2016, Hamilton et al., 2017, Veličković et al., 2017]. GNN's have also been applied to recommender systems [Gao et al., 2023, Berg et al., 2017, Ying et al., 2018, Monti et al., 2017] where a link prediction between an item and a user can score the importance of the item and provide customized recommendations to each user. GNN's have allowed researchers to not just model biochemical structures, but also synthesize new chemical compounds [Li et al., 2018b, De Cao and Kipf, 2019, You et al., 2018], predict molecular properties [Gilmer et al., 2017], and determine protein-protein interactions [Fout et al., 2017]. GNNs are becoming ubiquitous in not just the aforementioned domains but also for tasks such as combinatorial optimization[Li et al., 2018c], social influence[Qiu et al., 2018], brain networks[Kawahara et al., 2017] and event detection[Nguyen and Grishman, 2018].

Global Receptive Field. Graphs provide a natural way to represent irregular data structures such as social networks or molecular configurations. A Graph G, is made of N nodes and E edges. Nodes in a graph are akin to pixels on an image, while the edges describe which nodes(pixels) are immediately adjacent to which other nodes(pixels). Such an edge specification can be represented as an *adjacency matrix*, $A \in \mathbb{R}^{N \times N}$. The difference between graphs and images is that a node can be adjacent to any other node in the graph, while the pixel in an image can be adjacent to at most 8 other pixels. While a graph of N nodes can have N^2 different connections, an image with N total pixels can only have upto $8N$ total connections. This allows graph activations to consider any other activation in the previous layer, while 2D Conv is restricted to only the activations within the kernel window. Hence, graphs become very powerful and expressive networks, capable of capturing relationships and gathering knowledge from seemingly irrelevant regions in the data. They generalize Transformers' Attention mechanisms [Veličković et al., 2017] and Convolution in any dimension.

Graph Convolution. Graphs correspond to images while 2D Conv kernels correspond to Graph Conv kernels. However, Graph Conv kernels are simple MLP's that process a single node's embedding. The result can be used in *Neural-Message-Passing* and *Aggregation* steps. Attention weights can also be computed on every edge. Finally, depending on the connections specified by the adjacency matrix,

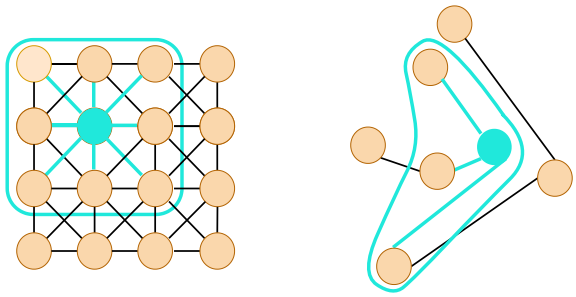


Figure 2.1: 2D Convolution(left): Shown in blue are the active elements under the kernel. The adjacency is implied by the intrinsic euclidean grid structure of images, hence, the values that get aggregated depend on the kernel's shape in CNNs. Graph Convolution(right): In GCNs however, the adjacency in the irregular space of graphs determine the kernel's shape. Instead of defining the kernel in CNNs, the adjacency on the graph is defined. In CNNs, receptive field can be increased with more layers, in GCN's hoewever, the receptive field is increased with more message-passing iterations. The global view of the graph can be accomplished by defining a fully dense graph, with every node having a direct link to every other node, while on CNN's, the same effect can be achieved with a kernel of the same spatial dimensions as its input[Wu et al., 2020].

embeddings of nodes in the neighborhood of a node of interest can be aggregated into its own embedding and is akin to attention mechanisms in Transformers and activation maps in 2D Convolution. formally, let $G = (V, E)$ be a graph composed of V nodes and E edges. Let $v_i \in V$ denote a node and $e_{ij} = (v_i, v_j) \in E$ denote a directed edge from v_j to v_i. Let $\mathbf{h}_i^0$ represent an initial node embedding for node v_i. Performing Graph Convolution is then:

$$\begin{aligned} \mathbf{h}_{ij}^l &= \theta_a(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) \\ \mathbf{m}_{ij}^l &= \theta_n(\mathbf{h}_{ij}^l, \mathbf{h}_j^{l-1}) \\ \mathbf{h}_i^l &= \sigma(\{\mathbf{m}_{ij}^l\}_{j \in N_i}) \end{aligned} \tag{2.1}$$

where l iterates L graph conv layers. θ_a is a learnable function with parameters, but can also be the dot product. θ_n is yet another learnable function, but could be

a simple multiplication. σ is then an order-invariant aggregation operation such as sum, mean, max or min. Observe how graph conv operations can be generalized to attention mechanisms, 2D convolution, and can even be extended to embed features on the edges.

Invariance Properties. Transformers do not use recurrence and therefore cannot utilize the order in which the tokens are arranged i.e. it is invariant to the order. In 2D Convolution however, the order of the pixels is implicit to the grid-shaped kernels and inputs. Similarly, graphs have no concept of the order of the nodes, hence, the aggregation function has to be invariant as well. This corresponds to summation, mean, minimum, maximum and weighted average functions as the choices of aggregation functions. Such functions will give you the same result regardless of the order in which its elements are i.e. the sum of 1+2+3 will be 6, but so will 3+1+2.

Position. While the attention mechanism in Transformers is order invariant, the task of language processing requires awareness of syntax, which demands an understanding of the order of the tokens. Therefore, Vaswani et al [Vaswani et al., 2017] embed 1D sequence positions into the tokens themselves. Similarly in tracking, while the aggregation functions are order invariant, certain physical rules need to be modelled, such as how objects can move through space and time and how objects may influence each others motion to avoid collision. Therefore, we may embed 3D spatio-temporal positions of each node into itself. Better yet, we embed normalized, relative 3D spatio-temporal magnitudes on the edges.

2.2 REINFORCE

In problems where what is correct and what is not cannot be accurately determined, Reinforcement Learning becomes an appropriate optimization method on learnable networks. For example, when playing games, making a decision may not immediately lead you to a good or bad state. But a sequence of decisions may ultimately lead to a final high or low score. Which decision(s) led to an agent's success/failure may not be obvious or deterministic, since some may have been good, some bad, and those definitions depend on the state. Hence, determining the responsible decisions that

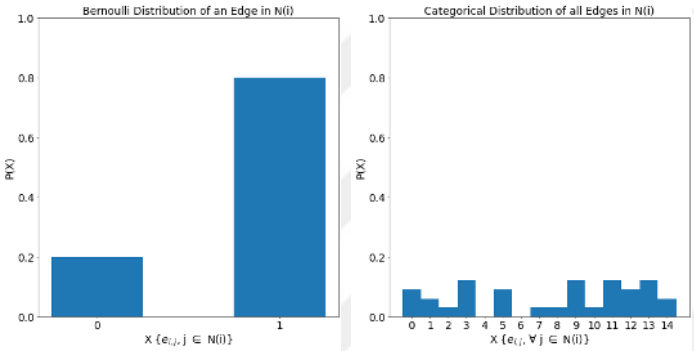
one would like to optimize is known as the *credit assignment problem*. The only way to assess the agent is via its final score, which may not be differentiable with respect to the agent's learnable parameters. This is where the REINFORCE[Williams, 1992] algorithm becomes useful.

Assume we have an agent π, with learnable parameters θ, which takes as input a representation of its environment, history and/or states, s. At each time-step, the agent predicts a probability distribution $p(\mathbf{a}|\pi^{\theta}(\mathbf{s}))$, over its action set i.e. the set of possible actions it can take, and receives a reward. Optimizing the parameters such that overall expected reward is maximized is then performed by minimizing the loss via gradient descent:

$$r \times -log(p(\mathbf{a}|\pi^{\theta}(\mathbf{s}))) \tag{2.2}$$

2.3 Stochasticity

For our use case, we require the chosen action to be more stochastic; we want the actions to be sampled such that exploration is encouraged in early stages. Furthermore, as will be described in more detail in Section 4.4.1, we require the tracker to be able to simultaneously explore multiple actions at each time-step/frame, since our method relies on exhaustively exploring through an abundance of varying possible futures. Therefore, we use Bernoulli sampling rather than Multinomial/Categorical sampling. The benefits are 2 fold; first, the sampling allows for variation, hence encouraging exploration. The sampling is not directly differentiable, but is made so via the reparameterization trick [Schulman et al., 2015]. backward computation are also elegantly built into PyTorch [Paszke et al., 2017]. Secondly, it allows for multiple actions to be taken, from a set of actions that are always changing. Note that in our application, the action set corresponds to the set of outgoing edges at a particular node. Hence, utilizing REINFORCE with Bernoulli sampling on graphs is an elegant approach to modelling the complexity of the MOT task.



(a) (b)

Figure 2.2: Showing the difference between Categorical and Bernoulli distribution based sampling. (a): Bernoulli sampling allows independently sampling an edge, allowing multiple such edges to be simultaneously sampled with their respective distributions. (b): Categorical sampling normalizes the probabilities of all the edges in the neighborhood of node i, and allows sampling of strictly one edge only

Chapter 3

RELATED WORK

The majority of cutting-edge MOT (Multiple Object Tracking) methods adhere to the tracking-by-detection approach, in which the problem is breaken down into two distinct stages: First, identifying the positions of pedestrians in each frame independently, with neural networks currently being the most advanced method [Ren et al., 2015, Yang et al., 2016, Redmon and Farhadi, 2016, Yang et al., 2016], and secondly, establishing connections between the identified detections across different frames to create trajectories.

In the following sections, we first describe how graphs are adapted to tracking, and the association cues that are required in Section 3.1. Then, we describe how deep learning is utilized to solve the various components of the tracking problem in Section 3.1.1. We proceed to describe trends in recent graph tracking approaches that use learnable methods and provide a summary of recent graph tracking works in Section 3.1.2. In Section 3.2, we describe other tracking approaches that mostly adapt detectors to perform tracking and highlight what complexities of tracking other methods emphasize. Here, we describe the complementary benefits that jointly training a detection and tracking pipeline have, how they learn better features and how FP-FN trade-off can be handled. We highlight that while we do not jointly train our detector and tracker, we couple them by delegating the detector refinement to the tracker, such that global cues from the temporal domain can be exploited to improve detection quality. In Section 3.3, we describe how reinforcement learning and reward based methods have been adapted to tracking. We report that the trend in reinforced tracking is to adapt multiple, single-object-trackers(SOT) which do not always model interactions. We highlight the importance of task-specific rewards and how others either use immediate, local feedback or revert to more high level, but slower reward evaluation methods. We distinguish ourselves via a novel reward

design that encourages tracking on a more high level, global context, while being easy to compute and approximates tracking accuracy metrics. Finally, in Section 3.4, we describe other methods that resemble our approach to tracking by proposing. We highlight the importance of having excessive data to provide a rich and diverse set of training samples from the limited, annotated tracking data that is available. We liken our approach to the performance boost that two stage detectors obtain when using Region-of-Interest(RoI) heads, and introduce our Track-of-Interest(ToI) head.

3.1 Tracking as a Graph Problem.

Correspondences between samples can be performed online either via a frame-by-frame processing pipeline for online tasks [Breitenstein et al., 2009, Ess et al., 2008, Pellegrini et al., 2009] or track-by-track for offline applications of video understanding[Berclaz et al., 2006]. Batch methods are generally preferred for offline tasks due to their increased robustness against occlusions. The standard approach to modeling data correspondences offline involves using a graph, where each detection is represented as a node, and edges indicate potential connections between them. How the cues are used to predict the associations vary greatly throughout the literature e.g network flow formulations [Berclaz et al., 2011], multi-hypothesis tracking [Henschel et al., 2017], conditional random fields [He et al., 2015] or quadratic pseudo-boolean optimization[Ess et al., 2008].

Association cues. The set of cues that lead to enhanced tracking can be categorized into two; motion in 2D[Bewley et al., 2016, Bochinski et al., 2017, Feichtenhofer et al., 2017, Xiao et al., 2018, Yu et al., 2016, Held et al., 2016] or 3D[Hu et al., 2021, Osep et al., 2017, Sharma et al., 2018, Mitzel and Leibe, 2012, Luiten et al., 2019], or visual similarity[Bergmann et al., 2019, Wojke et al., 2017, Milan et al., 2016b, Kim et al., 2015, Sadeghian et al., 2017, Leal-Taixé et al., 2016, Son et al., 2017, Yang and Nevatia, 2012]. The literature is abundant with works that argue that one cue is more discriminative than the other, or that other cues are unnecessary; [Pang et al., 2021] claim geometric, spatio-temporal cues are not beneficial

while [Zhang et al., 2022] prefer them over visual cues.

In [Saleh et al., 2020, Qin et al., 2023, Dendorfer et al., 2022], online MOT is approached by modelling the motion. By exchanging interactions with geometric cues, they predict trajectories. During occlusions, [Saleh et al., 2020, Dendorfer et al., 2022] employ a probabilistic model to propagate several plausible futures of the track in anticipation of their geometric reappearance. Instead, [Guo et al., 2021, Ren et al., 2023] focus on scrutinizing the detections' features by learning to predict masks. The masks attenuate regions in the bounding box that occlude the person, thereby only using relevant pixels to generate more refined, informative visual/ReID embeddings. Such refined features that only embed specific pixels that belong to the segment alleviate occluded objects from having high visual similarity in the feature space.

With graphs however, utilizing all available cues can be modelled rather naturally [Brasó and Leal-Taixé, 2020, Cetintas et al., 2022]. While other works [Wojke et al., 2017, Wang et al., 2019b, Zhang et al., 2020, Wang et al., 2021b, Wang et al., 2021a] fuse appearance and motion features, Hyun et al [Hyun et al., 2023] claim that they are not discriminative enough in crowded scenes and require higher order representations from multiple message passing iterations.

The choice cue(s) can then be used to compute learned costs [Leal-Taixé et al., 2014] and formulated as either a maximum flow [Berclaz et al., 2011] or a minimum cost problem. Both formulations can be efficiently and optimally solved. In our work, we focus on tracking offline, using graphs, with visual and spatio-temporal motion cues and follow the tracking-by-detection paradigm.

3.1.1 Learning in Tracking.

It is widely recognized that neural networks have become the dominant approach in various vision tasks, especially after [Krizhevsky et al., 2012] demonstrated their potential in image classification. This trend has also made its way into the field of tracking, where learning has primarily been used to establish a mapping from images to optimal costs for the graph algorithms mentioned earlier. For instance,

Laura et al[Leal-Taixé et al., 2016] learn to predict the cost of associating a pair of detections by employing a symmetric network, while a combination of CNNs and RNNs is utilized in the same fashion by Sadeghian eta l[Sadeghian et al., 2017]. More advanced techniques, such as attention networks [Zhu et al., 2018], have resulted in improved outcomes.

In [Ristani and Tomasi, 2018], Ristani et al highlight the significance of learned ReID (re-identification) features for multi-object tracking. Braso et al[Brasó and Leal-Taixé, 2020] then claim that the aforementioned methods disjoint the optimization from the learnable costs. In contrast, [Schulter et al., 2017, Kim et al., 2012, Wang and Fowlkes, 2015] integrate the solvers into the learning process, and Li et al[Li et al., 2022] advance by formulating a differentiable flow solver that allows gradients to pass through it. Li et al[Li et al., 2022]'s pipeline however can only accomplish the differentiability by formulating a comprehensive two-step framework that pre-compute the required gradients during the forward pass in order to bypass the solver's non-differentiability, with limited success. Braso et al's method propose they can directly learn a solver. However, their method does not guarantee that the trajectories formed from edge classification are free from violations, and therefore revert to a post-processing pipeline that uses non-differentiable solvers.

Furthermore, [Brasó and Leal-Taixé, 2020, Li et al., 2022] simplify the solver by ignoring the cost of the nodes, hence assuming that the detection set will always exclusively consist of True Positives. This works in MPN[Brasó and Leal-Taixé, 2020]'s favor as they train on the ground-truth annotation detection set, but leads to false associations when running on unseen data. To address this, we train our model with an abundance of raw, unfiltered detections that consist of several False Positives. We then formulate a Minimum-Cost-Flow(MCF)[Zhang et al., 2008] solver that, in contrast to [Brasó and Leal-Taixé, 2020, Li et al., 2022], constrains both nodes and edges jointly. We further implement a proposal-to-trajectory inference pipeline that directly resolves conflicts, yields non-violating trajectories and is devoid of solvers, yet is comparable to the MCF solver.

3.1.2 Recent Tracking Approaches on Graphs.

Offline graph trackers. Graph trackers such as [Brasó and Leal-Taixé, 2020, Cetintas et al., 2022, He et al., 2021] provide a structure that naturally allow detections(represented as nodes) to communicate to each other and update themselves, adaptively and selectively, such that they may encode identity, similarity and dissimilarity. More recent works [Dai et al., 2021, Girbau et al., 2022] implement hierarchical clustering methods to group smaller, local graphs into longer, global trajectories. These graph based trackers run offline by predicting association probabilities on the edges to produce tracks as a groups of connected nodes. By parameterizing graph hierarchies, Cetintas et al[Cetintas et al., 2022] are able to access larger time-spans in a learnable way, similar to how the receptive field of CNNs increases with more layers. At each hierarchy level, tracklets from the previous level are clustered into larger ones, until they form full-length trajectories.

Online graph trackers. Graph trackers that run online [Wang et al., 2021b, Chu et al., 2023, He et al., 2021, Xu et al., 2019, Shan et al., 2020], rather than offline, can be accomplished by formulating bi-partite graphs; nodes in a track-graph associate to nodes in a detection-graph after performing intra and inter-graph neural message passing. Wang et al[Wang et al., 2021b] were the first to propose a jointly trained detector and tracker with a GNN in an online setting, while modelling relations. They integrate the Centernet[Duan et al., 2019] detector with a DLA-34 backbone[Yu et al., 2018] and a Re-ID module directly into their pipeline. They then leveraged spatio-temporal object interactions by using GNNs to extract inter-object relations such that better features that improved both detection and data association could be learned. They also found GraphConv[Morris et al., 2019] to be better at embedding relations than other GNN processes; AGNNConv[Thekumparampil et al., 2018] and EdgeConv[Wang et al., 2019a].

He et al[He et al., 2021] embed track-nodes by taking the mean of the detections' features that have been associated upto frame t. They do not use spatio-temporal cues, but propose a differentiable, convex, quadratic programming layer within their graph convolutions to construct a quadratic affinity matrix of the edges, and an

affinity matrix of the nodes to produce an optimal score map to assign detections to tracks.

Xu et al[Xu et al., 2019] instead propose to utilize not just appearance and spatio-temporal cues, but also topological cues from the scene to model interactions using spatio-temporal relational networks. Shan et al [Shan et al., 2020] utilize appearance and emphasize the role of motion predicting to generate more consistent features for a progressing target. They achieve this by re-extracting the features of the tracklets in the current frame based on motion predicting. The adaptive graph neural network[Shan et al., 2020] is designed to fuse not just appearance and locations but also historical information, and is crucial in distinguishing different objects.

Chu et al[Chu et al., 2023] implement an online graph tracker with transformer inspired modules; A buffer of track graphs for the past T frames are maintained and undergo self attention across the spatial domain. The output of which is transposed such that self attention across the temporal domain may be performed. A detection graph from the current frame also undergoes spatial self attention, before being cross attended to the track graph. The result is an extended assignment matrix with additional special rows/columns to handle occlusions, entering and terminating tracks. Unlike transformer trackers however, [Chu et al., 2023] mask the attention operations such that only targets in close vicinity to each other can attend to each other, both inter and intra graph, and both spatially and temporally, allowing the network to focus on the challenging, ambiguous tracking scenarios.

Hyun et al[Hyun et al., 2023] implement a joint detector and online graph tracker; they maintain top-K detections from the current frame, such that the several spurious objects may capture objects that could be missed due to low confidences. They then claim that by performing several message-passing iterations, higher order features such as interactions and occlusions yield more robust association matrices despite having high confidence FPs and low confidence near-misses, all to capture long-term dependencies without a motion model.

In our work, we employ an offline graph tracker such that global knowledge can be conveyed across the graph before predicting associations. This allows us to track

through missed detections and occlusions while anticipating and avoiding identity switches. However, rather than using graph-hierarchies to increase the global scope, we compress the embeddings, allowing us to jointly access temporally larger, global scopes with a single monolithic graph. This avoids having to train the hierarchies incrementally and simplifies the training process.

3.2 Improvements in Tracking Approaches with Deep learning.

Learning more discriminative features. Wu et al[Wu et al., 2021] claim that ReID is trained to scrutinize inter-identity variance for given detections, while detection -under a contrastive loss- could enhance inter-identity difference and minimize intra-identity variance. Works by [Wu et al., 2021, Feichtenhofer et al., 2017] were the first of their kind to recognize that detection and tracking could benefit from each other when trained jointly. Detectors were then adopted to directly predict the displacement of objects in subsequent frames [Bergmann et al., 2019, Zhou et al., 2020, Peng et al., 2020], hence learning tracking jointly on the detector. However, these architectures were prone to tracking errors caused by short-term occlusions.

To track through occlusions required cues that could be extracted from subsequent frames rather than motion cues that simply extrapolated history. By adding an extra head to a detector, [Zhang et al., 2020, Yang et al., 2019a, Wang et al., 2019b, You et al., 2023] were able to extract visual cues from a frame and match them to another frame in the future. This allowed similarity scores to be made between visible sections of a partially occluded trajectory. These methods however, were prone to identity switches in ambiguous scenarios where people dressed the same or walked together. To alleviate visual ambiguity, [Pang et al., 2021, Yu et al., 2022, Kim et al., 2021] propose to utilize distractor objects as contrastive targets to drive discriminative feature learning. [Sadeghian et al., 2017] model occupancy grid maps for every target to allow interactions of nearby targets to influence its own motion. Kim et al[Kim et al., 2021] pool embeddings of distractor tracks into the affinity computation, while Pang et al[Pang et al., 2021] use dense distractor and background detections to do the same. According to Yu et al[Yu et al., 2022],

computing contrastive embeddings for both tracks and detections are not feasible in online settings, but can be gotten around by storing trajectories as a single embedding vector.

Tracking with Transformers. Recent adaptations of Attention in Computer Vision have led to Transformer based tracking. Zhou et al [Zhou et al., 2022] adopt a transformer architecture to group detection tokens directly into trajectories in an offline fashion. Different from [Zhou et al., 2022], in online transformer-trackers such as [Zeng et al., 2021, Xu et al., 2021, Sun et al., 2020, Meinhardt et al., 2022, Cai et al., 2022], a frame is tokenized, and learnable object queries attend to different regions in the image, each responsible for detecting a unique object entering the scene.

Object queries evolve into track queries once they detect an object. The track queries then naturally interact through attention mechanisms, before attending to queries from the next frame, where each track-query is responsible for identifying and encoding its new appearance, position and motion in subsequent frames. The transformer based detector-tracker is the first formulation that truly unified detection and tracking, while allowing interactions to be considered thanks to the attention mechanisms, which we intuit is responsible for learning more discriminative latent features, as is in graphs. However, unlike graphs, such transformer-trackers could not receive global context from across the temporal domain and remained prone to not just detection errors, but also tracking errors.

False positive and negative trade-off. By focusing on being able to capture as many of the objects as possible, tracking metrics tend to improve. To track the abundance of objects, recent works [Yang et al., 2022, Aharon et al., 2022, Zhang et al., 2022, Pang et al., 2021] perform association in two rounds; Detector-based trackers maintain buffers of low-confidence detections and utilize them to associate to left-over(trajectories or detections that failed to associate due to low confidences) objects in the future. This two-step association qualifies low-confidence detections -which would have otherwise been discarded- into tracked objects, driving up recall and tracking accuracy metrics.

Inspired by Khurana et al[Khurana et al., 2021], Tokmakov et al[Tokmakov et al.,

2021] attempt to train a detector to explicitly continue detecting occluded targets by using synthetic datasets because they could be annotated automatically. Their method, unlike Cao et al[Cao et al., 2023], avoided the need of Kalman filters or post processing(interpolation) by incorporating a head to quantify target occlusion and prevent being penalized for FPs. However, [Stadler and Beyerer, 2023] then claim that some incorrect, high-confidence objects may be associated over correct, low-confidence ones. So they contribute a joint-association method that alleviates this issue by considering all objects -regardless of their confidence- simultaneously. We found [Stadler and Beyerer, 2023]'s observations to agree with ours. Hence, via message passing, the benefits of multiple rounds of association is embedded by our multiple message passing steps. The buffers to store and possibly reactivate low confidence trajectories is instead delegated to our proposal generation step and are stored only as likelihoods and detection indices, eliminating the need to maintain large trajectory embeddings. In recognition of [Wu et al., 2021, Feichtenhofer et al., 2017], we conjecture that by performing a Non-Maximum-Suppression later, at the tracker level rather than at the detector level, we allow global information from across the temporal domain to supply discriminative latent cues used in detecting False Positives, while increasing the recall.

3.3 Reinforcement Learning in MOT.

Online trackers tend to suffer from tracking drift, due to false association errors that accumulate [Xiang et al., 2015]. Despite this fact, previous attempts at Reinforcement Learning based tracker designs are online methods [Xiang et al., 2015, Yun et al., 2017, Rosello and Kochenderfer, 2018, Ren et al., 2018, Jiang et al., 2019, Wang et al., 2022] because online tracking can be modelled as Markov-Decision-Processes(MDP)[Xiang et al., 2015, Rosello and Kochenderfer, 2018]. They hence model each track as an agent in an environment with multiple Single-Object Trackers(SOT) [Xiang et al., 2015, Yun et al., 2017, Rosello and Kochenderfer, 2018, Ren et al., 2018, Jiang et al., 2019, Wang et al., 2022], only some of which model interactions between agents [Ren et al., 2018, Jiang et al., 2019]. In Ren et al[Ren

et al., 2018]'s approach, each track-agent learns its own trajectory via a Q-Network and an Action selection networks. At each timestep, the Q-Networks consider input from the environment's hidden state, containing a representation of all other agents at the previous timestep. Each Q-Network in turn updates the environment, before making an action decision. Similarly, Jiang et al[Jiang et al., 2019] model a Deep Q-Network for each agent using LSTMs. The LSTMs receive messages from other agents before selecting an action.

An issue with the multi-SOT approaches to MOT is that the number of trajectories need to be known so that corresponding agents may be initialized [Xiang et al., 2015]. A solution to this involves using a detector at the start of a tracking sequence, and initializing an agent for each detection. The agent then selects actions that transform the bounding box parameters such that they may capture the object in the next frame[Jiang et al., 2019, Yun et al., 2017]. Other actions are used by the agent to change its state to active-inactive, tracked-lost [Xiang et al., 2015] such that challenging and dynamic tracking scenarios such as occlusions, terminations and anticipated identity switches may be handled. An alternate solution to determining the required number of agents is the tracking-by-detection paradigm, where detections are handed to agents based on preliminary heuristics such as IoU or cosine distance between visual embeddings, and the agent decides whether to accept or drop the detection, wait or exit the scene, then the left-over detections intitialize new tracks [Wang et al., 2022].

In order to handle global context in such an approach, Wang et al[Wang et al., 2022] use a non-differentiable, initial round of assigning plausible detections to each agent. If the agents select the same detection, the environment resolves the conflict using linear programming, and assigns rewards at each time-step. The difference between these two approaches is that one's action set involves propagating initial detections, the other's action set involves choosing the best detection from a detector in its local vicinity. New agents in the latter can be initialized from left-over detections that do not get associated, while the former cannot initialize new agents, unless they use it jointly with other heuristics such as signals from environment interactions [Ren et al., 2018] or jointly regressing to the detection set[Rosello and

Kochenderfer, 2018, Ren et al., 2018]

Since reinforcement learning based trackers are online [Xiang et al., 2015, Yun et al., 2017, Rosello and Kochenderfer, 2018, Ren et al., 2018, Jiang et al., 2019, Wang et al., 2022] and can be modelled as an MDP, most works assign rewards at each time-step after the agents select an action. The reward for it is determined by considering the ground truth trajectories using Inverse Reinforcement Learning, providing some level of immediate, fine-grained supervision [Xiang et al., 2015, Ng and Russell, 2000]. On the other hand, Rosello et al[Rosello and Kochenderfer, 2018] use a variant of the MOTA metric as the reward itself. This requires the episode to end before it can be computed since MOTA's design by definition runs on a collection of tracking results over a time span. Therefore, when training, [Rosello and Kochenderfer, 2018] run their tracker for an episode, collect the predictions, compute the MOTA[Bernardin and Stiefelhagen, 2008], then perform a backward pass. This provides rewards that are more task-specific from a high level, but is a bottleneck because MOTA is not differentiable, is slow to evaluate and cannot be specialized to supervise individual components [Rosello and Kochenderfer, 2018].

Wu et al's [Wu et al., 2022] work on using a reward-based approach to learn the *Person Re-Identification task (ReID)*. They employ an agent to compare a query object to a sequence of key objects in subsequent frames. The agent receives rewards based on whether it selects a TP, TN, FP or FN. They target these metrics, and show in the ablation that with the reinforced gradients, not only does their method perform more accurately, but also learns better feature representations, scrutinizing subtle cues and ignoring redundant, uninformative activation regions. This motivates us to explore the effect of rewards on tracking on graphs with excessive data.

In our work, we instead propose a reward function that encourages longer and purer tracklets. We specially engineer sparse matrix operations that run on the GPU that allow us to evaluate every proposed tracklet efficiently when training on graph structures; we compute a specific reward for each proposal. This allows a high level, task-specific reward function from a more global context to directly encourage favorable tracking behaviour, with the added benefit of supplying fine-grained gradients to individual components of the proposals i.e. nodes and edges.

Our action set is stochastic during training, and corresponds to a single probability on each edge, and is set to deterministic upon inference, eliminating the need for hand-crafted action sets that require micro-level supervision. In addition, our method stochastically explores all options rather than select one edge when training, complementing our exhaustive-proposal-exploration approach to learn to select one good decision while recognizing the unfairly large number of bad ones.

The enumeration of possible tracklets leads to conflicts which are handled efficiently at inference time via our Track-NMS, an alternate method to linear programming that can also run on the GPU. Furthermore, while other works use refined detections from other detectors, we utilize the raw detection set to tackle the well-known fact that tracker accuracy depends on detection accuracy. Hence, we train our tracker in anticipation of scenarios where the detector fails to generalize well.

Lastly, we highlight that by performing message passing on graph structures, we model interactions in the graph's latent feature space that span more global contexts, allowing more information about the objects in the scene to be embedded, and considered before making a decision.

3.4 Proposal based MOT.

The method of proposing targets and filtering them based on confidence has been shown to detect better by two-stage object detectors such as Faster-RCNN[Ren et al., 2015]. Our method takes inspiration from Ren et al[Ren et al., 2015]'s work in detection applied to tracking; propose to track better by considering several possible tracklets and keep the most confident ones. While earlier works [Huang et al., 2013, Huynh et al., 2011] attempted to track by proposing smaller tracklets and iteratively grouping them into larger ones, they lacked a single training pipeline; instead, they trained different hierarchies separately and had to collect training data from lower levels to train subseqeunt higher levels [Huang et al., 2013] and resolved conflicting proposals in favor of purity[Huynh et al., 2011], since impurities could not be recovered from. This idea was explored by Yang et al [Yang et al., 2019b] to cluster similar faces on graphs by generating proposal clusters. More recently, Dai et al[Dai

et al., 2021] adapt similar ideas into tracking; proposals are generated by a deterministic pre-processing algorithm that utilizes pretrained visual, spatio-temporal cues. They train a classifier to score the proposals, optimizing for a binary purity and normalized length score. Differently from [Dai et al., 2021], our proposals are generated via differentiable random walk sampler based on learned association probabilities. Furthermore, rather than avoiding impure proposals altogether, our reward formulation encourages longer and purer proposals using a continuous function, allowing our network to utilize bad proposals to learn good ones via REINFORCE'd gradients. Additionally, we avoid the need for intricate training procedures of individual hierarchies.

Graph samplers were proposed by GraphSAINT[Zeng et al., 2020] in which induced sub-graphs are sampled by performing random-walks on a base graph. Such walks could be applied to generate tracklet proposals, but lacked the ability to meet flow constraints. Additionally, the probabilities used to perform these walks are based on the graph structure itself i.e. in/out degrees and are not differentiable. Hornkov et al [Hornáková et al., 2020] generate plausible walks using disjoint path solvers that meet the constraints, however, the constraints also restrict the same edge from being walked on more than once, limiting the enumeration of all possible walks. Furthermore, Hornkov et al[Hornáková et al., 2020] rely heavily on network flow solvers. Unlike GraphSAINT[Zeng et al., 2020]'s samplers, our random-walk sampler is based on learned, differentiable probabilities which returns each individual walk, separately. We implement the random-walk sampler in a highly parallelized way, enumerating all possible variants and plausible non-violating walks that can arise from the reduced search space of the sampled edges. We emphasize that each walk is in itself non-violating, i.e. no more than one edge can go into and out of a node, but each node can exist in multiple walks- we acknowledge that this is also a violation, but claim that these serve as additional training samples, and violations are resolved at inference devoid of flow-solvers.

Chapter 4

METHODOLOGY

4.1 Obtaining Dense Detections

In order to capture the high number of FNs, we use raw, un-suppressed *dense detections* from QDTrack[Pang et al., 2021] with a ResNet-50 backbone and a Faster-RCNN architecture. We take their COCO-pretrained version and fine-tune it on the CrowdHuman dataset[Shao et al., 2018] using *Static Image Training*, as in CenterTrack[Zhou et al., 2020]. We further train it on tracking datasets such as MOT17 since they contain crowded scenes with several occluded trajectories, in various driving scenarios. QDTrack trains the detector via *quasi-dense similarity learning*, where a method that can distinguish distinct targets while associating to identical targets in a contrastively trained embedding space is optimized for online MOT. This is performed by utilizing dense RoIs rather than discarding them.

4.1.1 Quasi Dense Similarity Learning

Rather than discarding low confidence detections, the RoI samples can be utilized for contrastive learning, pulling the embedding vectors closer for the same objects, and pushing them further for other objects. Given a key image I_1 in training, a reference image I_2 is sampled from a bounded temporal neighborhood. The feature maps of each RoI is extracted via RoI Align [He et al., 2017] which are compressed down to a 256D representation via a learnable embedding layer, trained as follows.

Let an RoI be considered positive if it sufficiently overlaps with a ground-truth object else negative. If two RoIs from two frames correspond to the same ground truth object, the matching between them is positive (i.e. the association has a label of 1), else negative(association label 0). Let $\mathbf{V}$ be the number of RoI samples on the key frame, and $\mathbf{K}$ the number of RoI samples from the reference frame to be

used as contrastive samples. The embedding loss, L_{embed}, used to learn the desired embedding space is optimized via,

$$L_{embed} = log\left[1 + \sum_{k^+}\sum_{k^-} exp(\mathbf{v}\cdot\mathbf{k}^- - \mathbf{v}\cdot\mathbf{k}^+)\right] \tag{4.1}$$

where $\mathbf{v} \in [1, V], \mathbf{k} \in [1, K]$ and $\mathbf{v}, \mathbf{k}^+, \mathbf{k}^-$ are the 256D feature vectors of the training sample, its positive and negative targets, respectively, in K. This encourages $\mathbf{v} \cdot \mathbf{k}^- \rightarrow 0$ for false associations, and $\mathbf{v} \cdot \mathbf{k}^+ \rightarrow 1$ for true associations. Once this space is learned, similarity can then be quantified with a single value between [0,1].

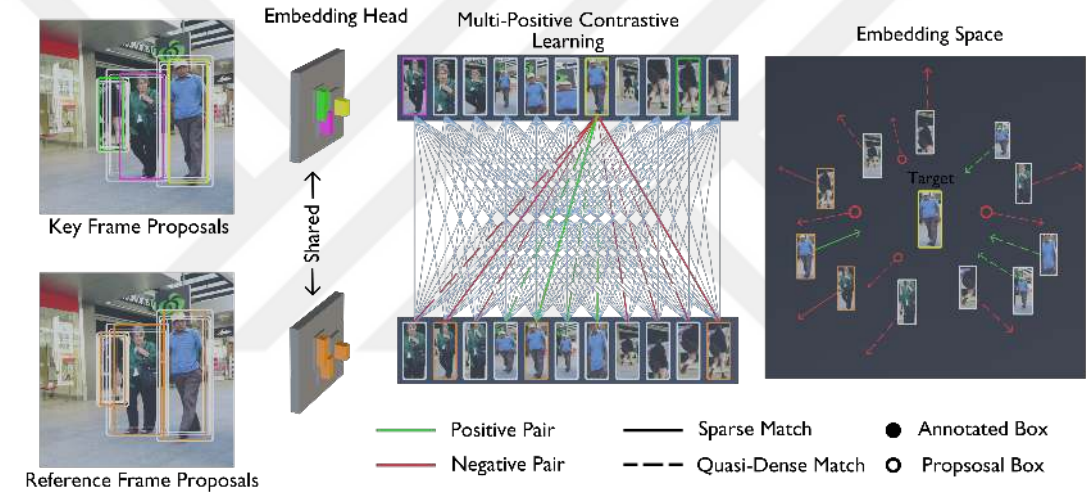


Figure 4.1: QDTrack Training Pipeline. Given a pair of key and reference images, feature vectors for all quasi-dense samples are extracted. By applying dense matching between them and optimizing the learned representation with multiple positive contrastive learning, the resulting feature space is capable of distinguishing different instances [Pang et al., 2021].

4.1.2 Object Association

Once the embedding layer is trained, it can be used for computing associations. However, tracking during inference may lead to challenging scenarios where one track could match optimally to multiple detections. To address this, Pang et al[Pang et al., 2021] formulate a *bi-directional softmax* to mimic a two-way agreement. Let

$\mathbf{m}$ be the feature embeddings for M trajectories tracked upto frame *t-1*, and $\mathbf{n}$ be the feature embeddings for N detected objects at frame t. Then, the similarity between a candidate match between detection i and track textitj is computed:

$$\mathbf{f}(i,j) = \frac{1}{2}\left[\frac{exp(\mathbf{n}_i \cdot \mathbf{m}_j)}{\sum_{k=0}^{M-1} exp(\mathbf{n}_i \cdot \mathbf{m}_k)} + \frac{exp(\mathbf{n}_i \cdot \mathbf{m}_j)}{\sum_{k=0}^{N-1} exp(\mathbf{n}_k \cdot \mathbf{m}_j)}\right] \tag{4.2}$$

A high $\mathbf{f}$ score using bi-softmax implies that the two objects are each other's nearest neighbor in the embedding space.

In our work, we do not use the bi-softmax association since a similar, bi-directionality is modelled in our graph as undirected edges. Rather, we notice how the embedding space, when trained under multiple-positive contrastive losses, with the bi-softmax, is a more tracking-specific alternate to ReID. We show an improvement in performance when using QDTrack's jointly-trained embedding layer over MPN's pretrained ReID network in Table 5.3.

4.2 Tracking via Graphs

Formulating the tracking problem as a graph is the first step. In this work, we approach tracking via the *tracking-by-detection* paradigm, where the set of detections from a video are collected from one network, then associated by another to form tracks.

We define an MOT graph as the set of nodes and edges, where nodes correspond to the detections, and edges to possible association hypotheses between said nodes. The nodes encode features about the detection, such as an appearance feature vector, while the edges encode relative features between the two nodes they connect, such as the relative distance in spatial(pixel) domain, temporal(frame) domain. The task is to, once initialized, predict the probability of all the edges in the MOT graph. Ideally, the two nodes with the most similar visual, spatial and temporal cues would contribute to higher activations on their corresponding edges, and vice versa is true. But, there can be ambiguities, caused by scenarios where people wear similarly colored clothing, walk together, visually occlude others. In such scenarios, minute dissimilarities would have to be relayed across the graph, to allow knowledge

to be scrutinized, and appropriate cues exploited, to allow more global informed decisions to be made on the edge. This has the advantage of allowing occluded tracklets the opportunity to re-identify themselves and associate correctly, avoiding greedily associating to the next-best-thing. This also provides opportunity to learn robustness to identity switches.

4.3 Message Passing Network

Let $G = (V, E)$, where V is the set of detection-nodes and $\mathbf{E} \in \mathbb{R}^{V \times V}$ the complete set of possible hypothesized associations, represent the MOT graph. This graph is the input to the Message-Passing-Network from [Brasó and Leal-Taixé, 2020]. Node embeddings are initialized using [Brasó and Leal-Taixé, 2020]'s pretrained ReID network, which is responsible for generating identical embeddings for people with people with the same identity. Edge embeddings are parameterizations of relative timestamps, bounding box location and sizes(see details in section 5.4.5).

Neural Message Passing[Brasó and Leal-Taixé, 2020] is then performed for several steps, allowing nodes and edges to relay information, and update themselves to encode, we can intuit, information about their neighbors, who is most similar, who isn't, how far away they are. Intuitively, if we perform 12 message passing steps, then two nodes that are *12-hop neighbors* apart will be *aware* of not just each other, but also the neighborhood between them.

To formalize, given an MOT graph $G = (V, E)$, let h_i^0 be the initial ReID embedding for the node $i \in V$, and $h_{i,j}^0$ the the initial, parameterized edge embedding for every edge $(i, j) \in E$. A forward pass is performed for every node:

$$\begin{aligned} \mathbf{h}_{i,j}^l &= \phi_e([\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}, \mathbf{h}_{i,j}^{l-1}]) \\ \mathbf{m}_{i,j}^l &= \phi_v([\mathbf{h}_i^{l-1}, \mathbf{h}_{i,j}^l]) \\ \mathbf{h}_i^l &= \sigma(\{\mathbf{m}_{i,j}^l\}_{j \in N_i}) \end{aligned} \tag{4.3}$$

where l iterates the message-passing step, ϕ_e and ϕ_v are learnable functions that encode latent features from the nodes and edges respectively, $\mathbf{m}_{i,j}^l$ is the latent message received by the target node i from a source node j in i's neighborhood,

denoted by N_i, and σ is an order-invariant operation such as addition, maximum or average, which combines all the incoming messages.

At each message passing step, every node and edge embedding is simultaneously updated from the its neighbors, passing knowledge along, and encoding higher order latent features.

In practice, for a node at time step t, we segregate messages coming from nodes $< t$ from nodes $> t$ and process them with segregated functions, the result of which get concatenated into single embeddings during the update. This is to make the network *aware* of messages coming from future nodes and past nodes, and assist in learning temporal dynamics, motion, cause and effect. See [Brasó and Leal-Taixé, 2020] for further details.

In addition to the default MPN[Brasó and Leal-Taixé, 2020] which produces probabilities on every edge, we add one more head to predict probabilities for every node. These probabilities are a useful cue as to which nodes from the abundance are worth keeping and which are not. In practice, we keep them all and use the probabilities to help distinguish highly probable proposals from unlikely ones i.e. if a proposal is made of nodes/edges with low probabilities, it will rank lower, making it easier to purge out. Vice versa is true, see section 4.5.2 for details.

4.4 Track-of-Interest (ToI) Head

4.4.1 Random Walk Sampler

For an MOT graph $G(V, E)$, following message passing, each node $v_i \in \mathbf{V}$, and edge $e_{i,j} \in \mathbf{E}$ predicts a probability. A node v_i with $p(v_i) \approx 1$ indicates a likely detected object, and an edge $e_{i,j}$ with $p(e_{i,j}) \approx 1$ indicates a likely association between nodes v_i and v_j.

We pass the edge probabilities $p(e_{i,j}), \forall e_{i,j} \in \mathbf{E}$ to a *differentiable Bernoulli* sampler, to select a subset of edges $e'_{i,j} \subset \mathbf{E}$, yielding an induced subgraph $G' \subset G$.

We then initialize walks on nodes in G' that have an *out degree* larger than 1 and *in degree* of 0. Each walk grows progressively and independently using a process inspired by *artificial cell division*[Chatterjee et al., 2015] and *nuclear fission*.

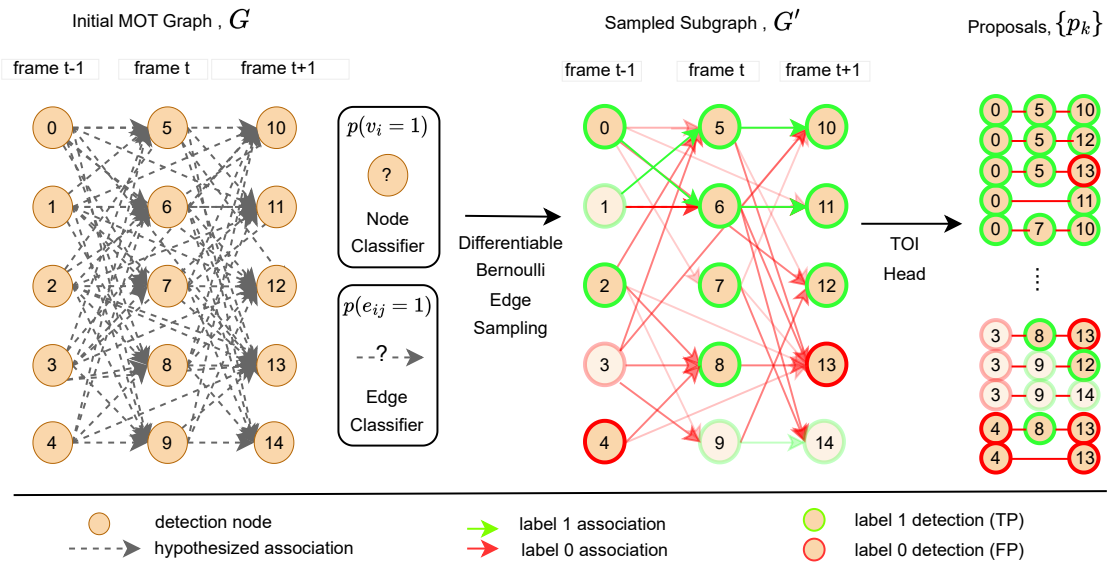


Figure 4.2: Main Training Pipeline. After initializing the MOT graph, G, it undergoes Neural-Message-Passing. The node and edge embeddings then go through their respective classifiers to predict $p(v_i = 1)$,$p(e_{ij} = 1)$, i.e. whether they are a True detection and association, respectively. We then use $p(e_{ij} = 1)$ to probabilistically sample a subgraph, G'. Shown in varying opacities of green and red are the predicted probability of the node and edge being the ground truth label. The ToI Head then performs parallelized walks to yield a wide variety of track-proposals, from which the losses and rewards are computed. The whole pipe is differentiable.

By utilizing sparse Tensor representations and functions, we are able to robustly represent multiple proposal lengths, simultaneously. If a walk encounters a node which has more than one outgoing edge, the walk first clones itself into as many outgoing edges as necessary, and each one of those clones differentiates by walking on one of the outgoing edges. In this way, we achieve *debranching*, such that each walk is a *unique* track-proposal that can be evaluated independently.

By performing random-walk-sampling in this way, we enumerate all possible walks that can be generated from the probabilistically sampled subgraph G'. This necessarily results in a huge imbalance of several *impure* and rare *perfect* track-proposals. We address this imbalance with the assigned score and our REINFORCE

formulation.

Our sampler is highly parallelized, and generates proposals of different lengths. All unique walks evolve simultaneously, but the evolution process itself is sequential, and has an upper bound equal to the number of frames in the graph. This allows us to achieve unprecedented sampling speeds, exhaustiveness and differentiability.

4.4.2 Proposal Score Assignment

To train our network using REINFORCE, we design a score for each proposal. The design of the score is crucial to direct the learning towards better proposal generation. To do so, we assign each proposal a score based on how long it is, and how pure it is in terms of the identities of detections that comprise it.

Let each node $v_i \in \mathbf{V}$ have an assigned ground-truth identity $g_\tau, \tau \in [1, T]$, where T is the total number of ground-truth trajectories. Let a proposal $\mathbf{p_k}$ be a set of associated nodes $\{v_{k,1}, v_{k,2}, ...v_{k,L}\}$ from the sampler, i.e. a proposed tracklet, where L is the number of nodes in $\mathbf{p_k}$. Let the dominating ID of the proposal $\mathbf{p_k}$ be the g_τ which occurs most frequently in the proposal. Let L_τ represent the number of nodes that make the ground truth trajectory τ, and L_k the number of nodes that make the proposed trajectory. The *length score* of proposal $\mathbf{p_k}$ is then defined as:

$$\frac{L_k(L_k+1)}{L_\tau(L_\tau+1)} \tag{4.4}$$

This allows longer proposals to receive increasingly longer scores, thereby encouraging the network to tend towards longer trajectories.

Let $c_{k,\tau}$ be the count of nodes in proposal k that have an assigned ground-truth identity g_τ . The *purity score* of the proposal is then defined as:

$$\frac{c_{k,\tau}(c_{k,\tau}+1)}{L_k(L_k+1)} \tag{4.5}$$

The score s_k of proposal $\mathbf{p_k}$ is then the sum of the *purity* and the *length* score and ranges between 0 and 2. Formulating a score in this way allows the proposal to be scored increasingly higher as it gets longer, while being increasingly rewarded for proposing tracklets with fewer ID switches.

4.4.3 REINFORCE Loss

For every proposal $\mathbf{p_k}$, consisting of nodes and the corresponding edges between adjacent nodes in the proposal, $\{\{v_{k,1}, v_{k,2}, ...v_{k,L}\} \cup \{e_{k,1,2}, e_{k,2,3}, ...e_{k,L-1,L}\}\}$, we compute the sum of log-probabilities of both the nodes and edges that constitute the proposal. Formally, the *proposal-loss*, l_k of one proposal $\mathbf{p_k}$ is:

$$l_k = \sum_{v_{k,i} \in \mathbf{p_k}} -log(p(v_k)) \quad + \sum_{e_{k,ij} \in \mathbf{p_k}} -log(p(e_{k,ij})) \tag{4.6}$$

We then weigh each proposal-loss in the batch with its assigned score, giving us the final *REINFORCE* supervision signal:

$$L_{reinf} = \frac{1}{K} \sum_{\mathbf{p_k}} s_k l_k \tag{4.7}$$

Where K is the total number of proposals in the batch. Our loss formulation is differentiable through the Bernoulli sampler, up through the probabilities, Message-Passing-Network aggregation functions, to the embeddings, making us, to the best of our knowledge, the first and only differentiable, proposal-based and REINFORCE'd tracker. This allows the network to learn tracking by learning to propose confident tracklets.

4.5 Inference Modes

4.5.1 Tracking via Minimum-Cost-Flow

The *Time-Aware-Message-Passing* ensures that 99% of the predictions are *non-violating*, which is measured by quantifying the *Flow-Constraint Satisfaction Rate*[Brasó and Leal-Taixé, 2020]. To handle the remaining 1%, MPN[Brasó and Leal-Taixé, 2020] utilize a *Flow-Solver*. Their formulation enforces that at most, only one edge can go in and out of a node. They use the predicted edge probability as the costs for the solver. We extend this further by utilizing the predicted node probabilities as well, to ensure that learned False Positives can be enforced to not have any edges

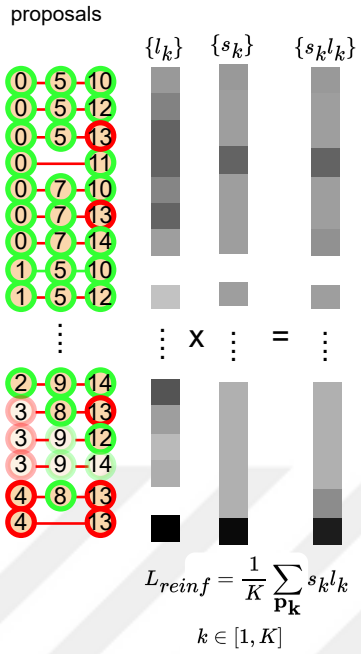


Figure 4.3: REINFORCE formulation to train from proposals. Using predicted probabilities from the edge classifier, we sample edges. The ToI head then enumerates all possible walks on the sampled subgraph, while maintaining differentiability. For each proposal, a ground-truth score and loss is computed; the average of their product is hence the REINFORCE supervision signal that allows us to perform fine-tuning from the proposals

going in or out of it. We implement a full *Minumum-Flow-Cost Solver (MCF)*[Zhang et al., 2008].

Following the predictions on the nodes and edges, the graph is then passed to a solver to enforce flow-constraints such that a node cannot be linked to more than one other node in either the future or the past. Formally, let $G = (V, E)$ be the graph representing a batch-sample MOT graph where each $v_i \in V, i \in [1, N_V]$ is a node-detection and each $e_{i,j} \in E, i, j \in [1, N_E]$ represents a possible association between nodes i and j. Recall that each prediction on a node, denoted v_i, is a logit representing whether it is a valid/invalid detection ie. a True Positive(TP) or a

False Positive(FP). Similarly, each prediction on an edge, denoted $e_{i,j}$, is a logit representing whether nodes i and j are associated or not. Then, to ensure that no more than one edge connects out of(to a future) or into(from a past) a node of interest, we apply a linear solver to find an optimal combination of associations by optimizing the objective:

$$min_{\hat{y}_{i,j}} ||\hat{y}_{i,j} - \hat{e}_{i,j}|| \tag{4.8}$$

with the following constraints:

$$\begin{aligned} \sum_{j,i \in E, t_i > t_j} \hat{y}_{j,i} \leq 1 \\ \sum_{i,k \in E, t_i < t_k} \hat{y}_{i,k} \leq 1 \end{aligned} \tag{4.9}$$

Where $i, j, k \in [1, N_V]$ and represent a node of interest at frame t_i, a node from a future frame t_j and a node from a past frame t_k, respectively, within the MOT graph. The solver returns the variables $\hat{y}_{i,j}$ that represent the optimal associations such that the constraints are met. Note that $\hat{y}_{i,k}$ are constrained to be integer values 1 or 0, representing an association or not, respectively. By optimizing the objective with such constraints, we also ensure that the chosen optimal is not too far from the logits predicted by the network.

The formulation above assumes that each node is valid, i.e. TP, which then unintentionally enforces that possible FPs(rogue detections) are considered in tracking. Such naive constraints can hinder tracking performance by causing more ID switches and/or fragmentation. To remedy this, *node-sense* is incorporated to the constraints such that the solver can optimize robustly despite the FPs. Formally, we specialize the constraints to:

$$\begin{aligned} \sum_{j,i \in E, t_i > t_j} \hat{y}_{j,i} &\leq \hat{x}_i \\ \sum_{i,k \in E, t_i < t_k} \hat{y}_{i,k} &\leq \hat{x}_i \end{aligned} \tag{4.10}$$

Where $\hat{x}_i$ are the solver's integer variables 0 or 1, representing whether a node is an FP or a TP, respectively, and are returned at the end of the optimization. This gives the solver more flexibility by allowing it to dynamically avoid forming associations to and from a node of interest-thought to be an FP-while maintaining the original task of allowing at most one association into or out of a node of interest if it is thought to be a TP.

We further specialize the objective to:

$$min_{\hat{y}_{i,j}} \sum_{i,j \in E} C_{i,j}(\hat{y}_{i,j} - \hat{e}_{i,j})^2 + \sum_{i \in V} C_i(\hat{x}_i - \hat{v}_i)^2 \tag{4.11}$$

Where C represents the negative-log-probability of the corresponding node, $C_i = -log(\hat{v}_i)$ or edge, $C_{i,j} = -log(\hat{e}_{i,j})$. Incorporating the negative-log-probabilities into the objective allows it to take into account the predicted logits from the network and ensures that the optimal does not diverge too much from it. The optimization is accomplished with a linear-programming solver that implements a *Minumum-Flow-Cost Solver (MCF)*[Zhang et al., 2008]. The MCF solver takes in node and edge probabilities and yields integer predictions over the nodes and edges. In other words, *node-sense* refers to the solver taking into account the validity of the node in the objective function.

4.5.2 Tracking from Proposals

Track NMS. The proposals generated by the *ToI head* enumerates all the possible walks that can be made on the sampled edges, making it complete set of unique walks. However, this means that a detection-node that participates in one walk may appear again in several others. This is considered a violation because it implies that the same detection has formed multiple trajectories. While such scenarios

are essential for training the network exhaustively, it is not sensible at inference time.

Therefore, similar to how excessive, redundant detections from an object detector on one frame undergo *Non-Maximum-Suppression*, we perform *de-overlapping*[Dai et al., 2021] on the proposal-tracklets formed by the graph of a short temporal window. This is performed by first sorting the proposals by decreasing negative-log-likelihood, l_k. It is important that the criteria for the sorting are the proposal negative-log-likelihoods, since the proposal score is derived from ground-truth annotations which will not be available during inference.

We then process each proposal by removing nodes if they have appeared in any proposal that is ranked higher. By doing so, we allow a detection-node to remain in the highest probable proposal, while purging it from any other proposal, hence resolving violations. This has the added benefit of purging out entire unlikely proposals since it is likely that all its nodes have appeared at higher ranked proposals. This operation is also implemented with GPU-optimized operations. We analyze this idea in section 5.4.1.

Track Merging. After processing the proposals into non-violating tracklets, they must be then merged to subsequent tracklets from subsequent windows into larger trajectories. To do this, an association cue is required, such as *track IoU*[Cetintas et al., 2022], which measures the ratio of detection-nodes that two tracklets from different temporal windows share in common, defined as:

$$IoU_{track}(p_i, p_j) = \frac{|(p_i \cap p_j)|}{|(p_i \cup p_j)|} \tag{4.12}$$

Where $i \in I$ iterates tracklets from one window and $j \in J$ another. This demands that there be an overlap between the two temporal windows. We use 50% overlap, and restrict the IoU_{track} to operate within the scope of the overlap region only. This

guarantees that the IoU_{track} is 1 when there is a perfect overlap, and 0 when there is none

The association is then performed via *Hungarian* matching with the following cost matrix, $cost \in \mathbb{R}^{I \times J}$:

$$cost[i,j] = \begin{cases} 1 - IoU_{track}(p_i, p_j) & if|(p_i \cap p_j)| \geq 0 \\ \infty & otherwise \end{cases} \tag{4.13}$$

The *Hungarian* algorithm performs one-to-one optimal assignment using the cost matrix, which we use to merge two associated tracklets into one. This is then performed again and again over subsequent overlapping windows by sliding it over the entire video sequence. The end result are a set of complete trajectories.

In practice, we implement a recursive solution. We also maintain the tracklets even if they don't associate at any given step, to allow it to re-associate after an occlusion.

Chapter 5

EXPERIMENTS

In the following sections, we first describe the tracking dataset that we train on. Following it, we describe implementation details such as the losses we use, the pretraining and finetuning specifications and then describe the different visual embeddings that we extract and use. Following that, we perform quantitative analyses on how our tracker performs relative to our baselines[Brasó and Leal-Taixé, 2020, Pang et al., 2021] in different validation and test settings. We then ablate our design choices by first quantitatively describing the learned proposals. We describe how well the proposals can be synthesized into trajectories using our inference strategy and highlight why a correlation between objective proposal quality and proposal negative-log-likelihoods must be learned. We further ablate how our REINFORCE formulation benefits the tracking objective, and how this can be further improved with different visual, and spatio-temporal embedding choices.

5.1 Dataset

MOT17: This dataset[Milan et al., 2016a] consists of 14 videos of pedestrians walking in different scenarios such as in a shopping center, a crowded street, or recorded by a moving camera, providing a diverse set of tracking challenges. The dataset is split into 7 training videos and 7 test videos, totalling at over 15,000 image frames, over 1,600 trajectories formed from over 300,000 annotated bounding boxes. Annotations for the test set are not available, and performance metrics on the test set are obtained by submitting inference results on the benchmark[Milan et al., 2016a]. There is a limit on how many times one may submit. Therefore, validation splits have to be designed from the training data.

Other works tend to divide the training set into half-train and half-validation

i.e. the first half of every video is in the training split, while the second half -which are continuations of the first half- are in the validation set. However, when running [Pang et al., 2021] in a 3-fold-cross validation setting, it was found to struggle to generalize. This suggested overfitting in half-train-half-validation setting. Therefore, we perform most of our MOT17 experiments by diving the dataset into 3 splits, and perform 3-fold cross validation. Each split contains full videos, avoiding data-distribution-leaks between train-val splits, allowing us to reliably test generalizability and overfitting.

Other pedestrian tracking datasets exist on the MOTChallenge benchmark, such as MOT15[Leal-Taixé et al., 2015], MOT16[Milan et al., 2016a] and MOT20[Dendorfer et al., 2020]. We avoid training on MOT15 as the dataset is relatively small compared to MOT17. Furthermore, the average density is 7.3 detections per frame, which is not as occluded/challenging as MOT17, with an average density of 21.1. MOT16, on the other hand, has an average density of 20.8, but has three times fewer images than MOT17. Furthermore, MOT17 already includes MOT16 sequences, along with updated annotations. Hence we refrain from using a dataset that MOT17 renders as redundant and not as challenging.

MOT20, however, has an average density of 149.7 boxes per frame, but has half as many frames as MOT17 and only 4 training sequences. Therefore, while MOT20 does have a significantly more challenging set of sequences, the dataset has significantly less variety, causing concerns for diverse training samples and overfitting. Further concerns involve GPU constraints; our goal is to push the global reasoning capacity of offline graph trackers by exploiting the temporal context- but with dense sequences, the temporal window would have to be shortened. Hence, we avoid MOT20 as well, but consider it in future work.

5.2 Implementation Details

Losses: We supervise the nodes and edges using *focal loss*[Lin et al., 2017]. We compute focal loss with $\alpha = 0.5$, $\gamma = 2.0$ for both nodes and edges. Higher α values weigh the loss from positive predictions more, while higher γ values weigh *hard pre-*

dictions(predictions whose labels were predicted with a low confidence) more. This allows the network to focus more on sifting through the excessive dense detections, while also paying attention to predictions that are difficult. We apply the supervision to the latent logits at all message-passing layers. Our final loss is an equal composite of three losses: $L_{total} = L_{reinf} + L_{node} + L_{edge}$.

Pretraining and Finetuning: We use QDTrack[Pang et al., 2021] with a Resnet-50 backbone, with COCO pretrained weights, further trained on *CrowdHuman*[Shao et al., 2018] using *static-image-training*[Bergmann et al., 2019], then on *MOT17* using default hyperparameters. We extract the raw, unprocessed detections to use as input to our *ReiTracker*. We first pretrain *ReiTracker* using just $L_{node} + L_{edge}$ with a learning rate of 0.001, for 30 epochs. We then finetune for 30 more epochs by activating L_{reinf}, and reduce the learning rate to 0.0002. Without pretraining, the TOI-head initially samples too many edges, resulting in an explosion of proposals that cannot fit on an RTX A6000 with batch size of 1. By pretraining, the network first learns a reasonable edge predictor which reduces the number of edges that get sampled when reinforcing. When pretraining, we use a batch size of 10 graphs, each of 12-15 frames, then use a batch size of 1 per GPU when reinforcing. 6 Tesla T4's are sufficient for finetuning.

Embeddings: MPN[Brasó and Leal-Taixé, 2020] train a separate ResNet backbone ReID network to generate embeddings for the detections. The input to this network are RGB patches of the detections, resized to a consistent size. We believe this limits the ability to embed visual cues from the vicinity of the detection. Furthermore, for one detection, the network outputs a 2048D vector used to initialize node embeddings, and a 256D ReID vector, strictly used for computing cosine similarity and as the criterion for edge construction and pruning. Instead, we use an encoder, jointly trained with the detector, using explicit *Contrastive Loss* and *Bi-directional Softmax*[Pang et al., 2021]. The resulting 256D vector hence embeds visual cues from the vicinity of the detection due to the *receptive-field* phenomenon of CNNs, which we use as the node embedding, and the edge construction criterion. This reduced vector size allows us to increase our network parameters and the temporal range of the input graph samples.

5.3 Main Results

The *MOT17Challenge benchmark*[Milan et al., 2016a] provides a server onto which we can submit results on the test set. Shown in Table 5.1 section (a) is a comparison of our results against our baselines, using the publicly available detections provided by the benchmark. We get outperformed. However, the benchmark has a separate section for submitting results based on a custom set of detections; the private section. Here, our method really shines. As can be observed in Table 5.1 section (b), not only does it surpass MPN, but MPN struggles when subjected to dense detections. This highlights the specific advantage of our design which focuses on sifting through excessive data. It shows the importance of showing the network, exhaustively, what not to do. We were unable to beat QDTrack due to its detector-tracker design, which focuses on trainnig a powerful detector with several more parameters than MPN. Note however, how QDTrack struggles to perform well in a 3-fold-cross-validation setting, shown in Table 5.1 section (c).

5.4 Ablation Studies

5.4.1 Learned Proposal Quality Analyses

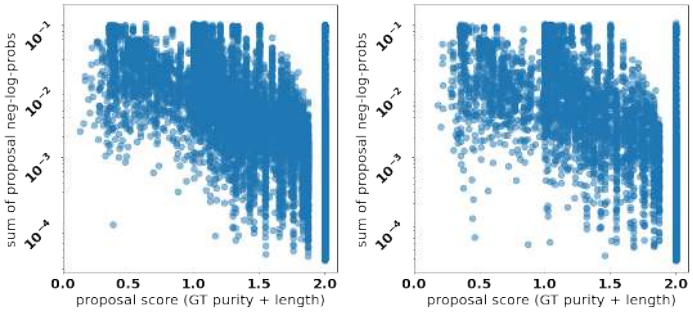
Tracking by generating proposal tracklets relies on the ability to have high quality proposals, and the ability to discard bad ones. To measure the quality of a proposal, we look at the score assigned to it(sec 4.4.2). However, the score is computed from the ground truth annotations of the detections' identities, meaning the score cannot be a cue during test time. Therefore, it is important that the network learns proposal likelihoods that correlate positively with the score, and this should generalize to unseen data. We analyse this in figure 5.1, which shows the learned correlation between proposal quality and the *sum-of-log-probabilities* of the nodes and edges that make the proposal. Further, we show that it generalizes this on unseen data as well. We can then convert the *sum-of-log-probabilities* to likelihoods by taking the exponent of its negative.

Table 5.1: **Benchmark Performance** We compare our approach to the state of the art on the: (a) MOT17 test set, Public Detection setting, (b) MOT17 test set, Private Detection setting, (c) MOT17 3-fold-cross-validation training set. † represents our method when using contrastively trained features, else we use ReID embeddings.

Method	MOTA↑	IDF1↑	FP↓	FN↓	IDSw↓
MPN	58.8	61.7	17,413	213,594	**1,185**
QDTrack	**64.6**	**65.1**	14,103	**182,998**	2,652
Ours	57.7	59.1	**10,901**	226,261	1,505
MPN	50.3	58.4	156,090	**119,475**	4,581
QDTrack	**68.7**	**66.3**	**26,589**	146,643	3,378
Ours	62.4	60.6	59,289	149,880	3,165
Ours†	62.8	61.9	58,368	149,094	**2,499**
MPN	64.4	70.8	**5,087**	114,460	**504**
QDTrack	38.0	43.1	16,569	190,296	2,055
Ours	69.1	67.7	29,025	72,258	2,823
Ours†	**69.7**	**71.2**	21,954	**40,041**	789

5.4.2 Proposal Enrichment Inference Strategy

We further show the wide variety of proposals that the ToI head generates, crucial for training the network to reason through the abundance of bad proposals, and eventually only propose the rare best. In figure 5.2, we show the wide range of proposals that the network is subjected to during training. We generate these by performing *Bernoulli Sampling*4.4.1 using the edge probabilities. At inference, we binarize the probabilities using 0.5 thresholding, allowing only confident edges to be walk-able. We show that this discards several bad proposals, leaving only highest quality proposals. Furthermore, we show that by specifying the walker to only start walks at nodes with no incoming edges (in-degree=0) and only at least one outgoing



(a) (b)

Figure 5.1: Showing the sum-of-log-probabilities vs score of proposals over the 3-fold dataset. The correlation shows that learned likelihoods are a useful sorting criterion. (a) shows the correlation for trained proposals. (b) shows the correlation on unseen videos

edge (out-degree > 1), we purge more bad proposals, enriching the set of proposals even more and shifting the distribution closer to perfection.

5.4.3 Effect of Losses

Our L_{reinf} formulation provides goal-oriented gradients, specific to the batch and the graph structure by focusing on presenting the longest and purest track proposals. We show in Table 5.2 up to a 10% increase in tracking accuracy(MOTA) when L_{reinf} is activated over its non-reinforced counterparts. However, all 3 losses are crucial to obtain identity preservation(IDF1), and we see that it correlates to lowering the False Negatives, thereby achieving our goal of capturing as many of them as possible.

5.4.4 Effect of Embedding

As described in section 5.2, we show the effect of using contrastive features, trained jointly with the detector. We showcase in Table 5.3 upto a 10% increase in tracking accuracy and identity preservation. We also boast significant improvement in FP, FN and IDSw. We attribute this to the power of compressed feature vectors trained by encoding not just the detection-pixels, but also the latent cues collected from the vicinity of the detection. The learned embeddings are hence more aware of

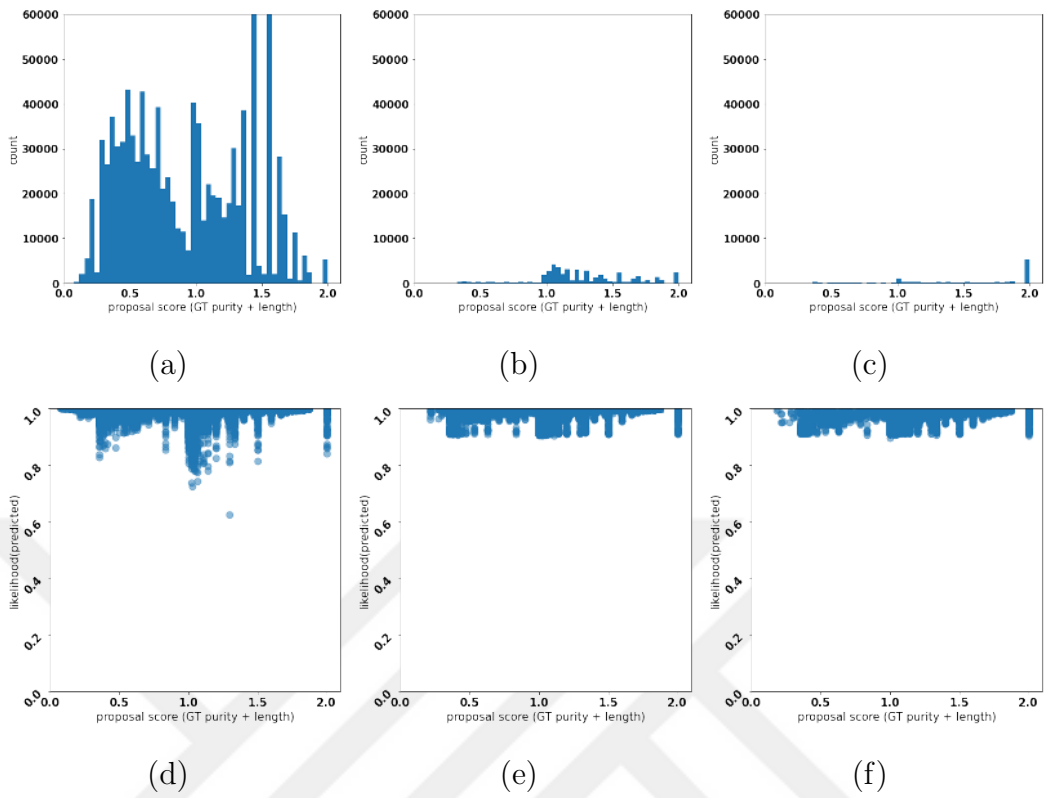


Figure 5.2: Investigating proposal quality distribution during (a): training, (b): inference and (c): when starting walks at nodes with out-degree > 1. Showing correlation between proposal likelihoods and ground truth score during (d): training, (e): inference and (f): when starting walks at nodes with out-degree > 1. All results are on sequences unseen when training. We show an enrichment in proposal quality by purging at the node level rather than thresholding at the proposal level.

their immediate surroundings which complement the GNNs ability to pass global knowledge and exploit subtle cues.

5.4.5 Effect of Different Spatio-Temporal Embeddings

While association based on appearance alone is a strong cue, it can fail in challenging crowded scenarios. This is because if one pedestrian is being occluded by another, their appearance will be similar since the processed pixels will be very similar, leading to possible ID switches. Therefore, along with the appearance embeddings being passed around the graph to relay similarity/dissimilarity knowledge, spatio-temporal

Table 5.2: **Ablating losses** We ablate REINFORCE training with and without node and edge supervision on the training set of MOT17 using 3-fold cross-validation. We use pretrained ReID embeddings for the appearance features

Loss							
L_{edge}	L_{node}	L_{reinf}	MOTA↑	IDF1↑	FP↓	FN↓	IDSw↓
✓			63.5	63.9	41,280	78,639	3,039
✓	✓		59.4	61.2	18,588	115,914	2,259
		✓	67.6	66.8	**18,378**	88,383	**2,253**
	✓	✓	**69.4**	66.9	26,319	73,977	2,574
✓		✓	63.7	64.1	37,842	81,462	2,880
✓	✓	✓	69.1	**67.7**	29,025	**72,258**	2,823

Table 5.3: **Ablating effect of different visual embeddings:** Pretrained ReID embeddings from MPN[Brasó and Leal-Taixé, 2020] vs contrastively trained features from QDTrack[Pang et al., 2021]

Embedding	MOTA↑	IDF1↑	FP↓	FN↓	IDSw↓
ReID	69.1	67.7	29,025	72,258	2,823
Contr.	**69.7**	**71.2**	**21,954**	**40,041**	**789**

knowledge about the detections could prove to be a useful, distinguishing cue for the association prediction. Consider the situation when two trajectories cross each other's paths. At the intersection frame, visual association would be ambiguous. In such a scenario, the relevant nodes and edges could potentially consider the aggregated, higher order embeddings on either side, and choose to associate to the trajectory with similar motion.

To determine the best method for embedding such spatio-temporal data into

the graph, we ablate them. We consider them jointly at pretraining and finetuning stages.

Baseline Embedding. As the baseline, we use [Brasó and Leal-Taixé, 2020]'s formulation for encoding relative geo-spatio-temporal positions between any source and target nodes, u, v, from different frames:

$$[t_j - t_i, \frac{2(x_j - x_i)}{h_j + h_i}, \frac{2(y_j - y_i)}{h_j + h_i}, log\frac{h_i}{h_j}, log\frac{w_i}{w_j}, \alpha] \tag{5.1}$$

where α is the appearance similarity, computed as a coine similarity between the ReID embeddings of the pixels from the source and target nodes' detection bounding boxes, computed from [Brasó and Leal-Taixé, 2020]'s pretrained ReID network. t is the frame number converted to a timestamp in seconds, computed using the fps at which the video is recorded. x, y, w, h are derived from the bounding box

Sinusoid Positional Embedding. MLPs have limited expressive ability to fit high frequency, nuanced information, such as in ambiguous scenarios, where the slightest difference could be the distinguishing factor. To address this, we consider *Sinusoid Positional Embeddings*,[Vaswani et al., 2017]. By embedding position scalars as fourier values with a variety of frequencies, the network is suddenly armed with more discriminative input, for values that may be highly similar.

We embed absolute visual, geo-spatio-temporal positions and add them to the node embeddings. Similarly, we initialize edge embeddings by encoding relative positions between two nodes:

$$\begin{aligned} PE_{node} = & [PE_{sin}(f), \\ & PE_{sin}(x_c), PE_{sin}(y_c), \\ & PE_{sin}(w), PE_{sin}(h), \\ & PE_{sin}(100 \times conf)] \end{aligned} \tag{5.2}$$

where f is the frame number of the detection, x_c, y_c are the center pixel coordinates of the detection bounding box, w, h the width and height, and $conf$ is

the detection confidence with a continuous value [0,1]. We multiply it by 100 since positional embeddings require integer inputs as the position. $P.E_{sin}$ is the sinusoid positional embedding function used by [Vaswani et al., 2017]. Each of the 6 attributes is encoded into a 6D embedding, i.e. $d_{model} = 6$ as per [Vaswani et al., 2017]'s notation. The result is concatenated to give a 36D node-positional embedding that is added to the feature vector of the node.

Similarly, the edges between any two source and target nodes, u,v, are initialized as:

$$\begin{aligned} PE_{edge} = & [PE_{sin}(f_j - f_i), \\ & PE_{sin}(x_j - x_i), \\ & PE_{sin}(y_j - y_i), \\ & \alpha] \end{aligned} \tag{5.3}$$

where α is the cosine similarity between the ReID embeddings of the nodes u,v. This embeds the appearance similarity as a continuous value.

Learned Positional Embedding. Considering that sinusoid positional embeddings allow extrapolatability such that any position can be encoded, they lack the ability to flexibly learn embeddings that would prove more useful for downstream tasks, such as message passing and aggregation for spatio-temporal and motion discrimination. Therefore we attempt to learn the positional embeddings.

This formulation is similar to 5.4.5, except that we replace $PE_{sin}(arg)$ with a learnable embedding matrix, from which arg indexes.

Table 5.4: **Ablating effect of different positional embeddings** We ablate the effect of the baseline, sinosoid and learnable positional embeddings encoded bsolutely on the nodes, relatively on the edges, and both. Evaluated on the training set of MOT17 using 3-fold cross-validation. We use pretrained ReID embeddings for the appearance features

	Encoded on						
PE type	*node*	*edge*	MOTA↑	IDF1↑	FP↓	FN↓	IDSw↓
Baseline		✓	**69.1**	**67.7**	9,675	**24,086**	941
Sinusoid	✓		67.3	66.9	9,281	26,558	837
		✓	67.2	64.3	5,618	30,451	691
	✓	✓	66.8	65.7	8,577	27,802	851
Learned	✓		54.7	55.2	4,496	45,755	580
		✓	53.8	54.0	**3,609**	47,614	**574**
	✓	✓	53.8	54.0	3,609	47,614	574

Chapter 6

CONCLUSION

We show that the baseline MPN is precision-specific, at the cost of recall; it struggles to handle a high label imbalance and relies on cautious, confident detections. Furthermore, with current memory constraints, jointly training an offline tracker with a detector is infeasible. Therefore, we learn tracking from raw, un-suppressed detections and show that it not only simulates joint detection and tracking, but also simulates learning in novel scenes where detectors tend spur several regions of interest, in favor of safety.

We specialize the graph-tracker to handle a high number of FPs while capturing more FNs. We also show that QDTrack struggles in a 3-fold setting, but is specialized to learn more discriminative features, and captures more of the FNs due to its quasi-dense detection approach. We utilize the best of both MPN and QDTrack to synergize each others strengths and compensate each others' weaknesses. Additionally, we show that FPs are better identified if suppressed at the tracker level, rather than at the detector level, proving the advantage of additional consistency cues from the temporal domain(global), rather than just the spatial(local) overlap.

We show that REINFORCE'd gradients, in conjunction with a carefully designed, high-level, yet task-specific reward expands the tracking capabilities of an offline graph tracker. It allows for stronger identity preservation, which translates to better tracking. It is capable of reasoning through a large combinatorial space with a large label imbalance to find the sparse optimal solutions. Furthermore, we implement a novel, stochastic and exhaustive random-walk process that complements the vast but sparse search space for optimal trajectories. By formulating the sampler in a parallel process, we avoid the MDP process and perform RL techniques such that offline tracking is complemented while being significantly faster.

Finally, we describe the violations that are introduced due to the offline approach,

and utilize a track NMS process on the GPU to resolve them without resorting to CPU flow solvers.

6.1 Future Work

Scaling the tracker to run efficiently on several other datasets such as MOT20[Dendorfer et al., 2020], MOTSynth[Fabbri et al., 2021], BDD100k[Yu et al., 2020], TAO[Dave et al., 2020] and DanceTrack[Sun et al., 2022], could test our method's ability to generalize on multiple classes such as vehicles, riders, any other object and perhaps even out-of-distribution objects. This would also allow testing whether our method can generalize to different domains such as synthetic data, different weather conditions, poor lighting conditions, fast moving objects, low frame rates, aerial/birds-eye view perspectives, multi-camera set ups or scenes where targets are dressed in uniforms. Currently, the system is capable of running on MOT20, MOTSynth and BDD100k. Unfortunately however, due to time constraints, we were unable to train and extract the required features. Nevertheless, scaling the system efficiently has proven to be an exciting engineering exercise, and further development with a focus on flexibility is encouraged.

Noting the power of contrastive targets to learn more discriminative features from Pang et al[Pang et al., 2021]'s work, it was observed that the contrastive samples tend to be in close spatial proximity to the target of interest. The idea of spatial proximity has been utilized in conjunction with GNNs to tackle the NMS algorithm by [Shen et al., 2022]. Shen et al[Shen et al., 2022] propose to form connected components of graphs on a single frame based on spatial proximity, and propagating each detections' confidences to relay beliefs about which detection(s) should be preserved and which should be discarded. We note that confidence is an unreliable criterion for suppression as several good detections may have low confidences, and Shen et al[Shen et al., 2022]'s work appears to be prone to FNs. However, detection confidence may still be a useful cue in conjunction with other cues to form a globally consensual, suppression criterion. Furthermore, such graph connected components may prove useful in jointly learning a discriminative, contrastively trained

embedding space.

Recent trends are tending to the real-time, online processing approach. Online approaches suffer from the lack of a global perspective, and accounting for anticipated re-emergence in the future demands patience-windows and algorithmic post-processing pipelines that depend on a fixed, non-trainable hyper-parameter threshold. Real-time approaches also require Hungarian-algorithm based association decisions which -while being as locally optimal as they can be- remain prone to greedy decisions. That being said, real-time Transformer-trackers -specifically the ones where track queries directly attend to the image to perform both detection(for uninitialized track queries) and tracking(for already initialized track queries)- bypass the algorithmic engineering efforts of Hungarian matching. This delegates the matching to the latent space within the attention layers, making joint-detection-and-tracking, truly joint, and fully learnable. Such attention layers may also benefit from the injection of contrastive weights directly in conjunction with the attention weights, however, more thorough design and experimentation would be required to ensure sanity of the desired operational objective. Additionally, recent trends involve graph-trackers adapted for real-time tracking; graphs are general enough to operate as CNNs and Transformers. Therefore, exploration into adapting the benefits of Transformers in Vision on to Graphs could prove to be a more flexible implementation method.

Tracking a target through an occlusion is performed via patience windows for real-time trackers. For offline graph-trackers, this is entirely unnecessary due to the edge construction criteria across the temporal axis. However, by delegating each target to a track-query(using real-time transformer trackers), a simple head can be implemented to indicate when a target is occluded. This relieves the reliance on patience windows and thresholds. Such a method can be used to explicitly allow models to track through occlusions. However, such annotations on tracking datasets are not plentiful, unless synthetically generated and annotated data is considered. Such an approach has been shown to prove effective at learning *object permanence*[Tokmakov et al., 2021], a concept that describes the illusion of smooth, continuous vision from discretized, choppy input in living organisms. Tokmakov et al[Tokmakov et al.,

2021] further demonstrate that the model may transfer well from synthetic to real data if simultaneously trained on both types of data. Such specific training for object permanence, in conjunction with an occlusion-prediction head, allows adaptive, learnable tracking of a target through an occlusion, without hard-coded patience windows.

Real-time trackers may revert to motion based propagation until visual re-emergence in order to track a target through an occlusion. A common strategy is to utilize Kalman filters or learned recurrent networks to predict a future location, or a variety of possible future locations [Saleh et al., 2020, Dendorfer et al., 2022]. However, such methods lack the ability to model complex, sudden changes in motion, while accounting for possible influences from other targets. A worthwhile exploration into utilizing GFlowNets[ben,], Neural Ordinary-Differential-Equations(ODEs) or *Liquid-Time Neural Networks*[Hasani et al., 2021] may prove useful as they promise significantly increased expressiveness with fewer parameters.

Parallax describes the visual effect of an object appearing in two distinct locations and directions when viewed from two different positions in space(and/or time), such as a stereo camera setup, or human eyes. The discrepancy can not only be utilized to estimate depth, but also the change in occlusion; when a target becomes occluded from one viewpoint, it may still be visible from another. Such discrepancy can further be utilized to indicate that a target is either about to be occluded or about to re-emerge. The depth may also be a useful cue to resolve ambiguities of highly overlapping detections in 2D space. This may direct exploration towards 3D tracking and may prove more robust.

Lastly, we note trackers that learn to embed precise pixels that belong to a target; Wu et al[Wu et al., 2022] use an agent to learn ReID by attending to specific pixels in the queried detections until a satisfactory conclusion about correspondence can be made while Guo et al[Guo et al., 2021] use attention mechanisms in their tracker to amplify pixels of the target while suppressing other pixels from distractor/occluding targets. In both works, segmentation-like attention maps are learned on each detection, allowing for more precise and discriminative embeddings to be extracted. Exploration into utilizing detectors with powerful instance segmentation

abilities may allow similar focus on pixels to be scrutinized. The segmentation masks may provide easy and established solutions to embed specific pixels only.

BIBLIOGRAPHY

[ben,]

[Aharon et al., 2022] Aharon, N., Orfaig, R., and Bobrovsky, B.-Z. (2022). Bot-sort: Robust associations multi-pedestrian tracking. *ArXiv*, abs/2206.14651.

[Bastings et al., 2017] Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., and Sima'an, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. *ARXIV*.

[Berclaz et al., 2006] Berclaz, J., Fleuret, F., and Fua, P. V. (2006). Robust people tracking with global trajectory optimization.

[Berclaz et al., 2011] Berclaz, J., Fleuret, F., Türetken, E., and Fua, P. V. (2011). Multiple object tracking using k-shortest paths optimization.

[Berg et al., 2017] Berg, R. v. d., Kipf, T. N., and Welling, M. (2017). Graph convolutional matrix completion. *ARXIV*.

[Bergmann et al., 2019] Bergmann, P., Meinhardt, T., and Leal-Taixé, L. (2019). Tracking without bells and whistles. In *ICCV*.

[Bernardin and Stiefelhagen, 2008] Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10.

[Bewley et al., 2016] Bewley, A., Ge, Z., Ott, L., Ramos, F. T., and Upcroft, B. (2016). Simple online and realtime tracking.

[Bochinski et al., 2017] Bochinski, E., Eiselein, V., and Sikora, T. (2017). High-speed tracking-by-detection without using image information.

[Brasó and Leal-Taixé, 2020] Brasó, G. and Leal-Taixé, L. (2020). Learning a neural solver for multiple object tracking. In *CVPR*.

[Breitenstein et al., 2009] Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., and Gool, L. V. (2009). Robust tracking-by-detection using a detector confidence particle filter.

[Cai et al., 2022] Cai, J., Xu, M., Li, W., Xiong, Y., Xia, W., Tu, Z., and Soatto, S. (2022). Memot: Multi-object tracking with memory. In *CVPR*, pages 8090–8100.

[Cao et al., 2023] Cao, J., Pang, J., Weng, X., Khirodkar, R., and Kitani, K. (2023). Observation-centric sort: Rethinking sort for robust multi-object tracking. In *CVPR*, pages 9686–9696.

[Cetintas et al., 2022] Cetintas, O., Bras'o, G., and Leal-Taix'e, L. (2022). Unifying short and long-term tracking with graph hierarchies. *ArXiv*, abs/2212.03038.

[Chang et al., 2019] Chang, M.-F., Lambert, J. W., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Chatterjee et al., 2015] Chatterjee, S., Hore, S., Paladhi, S., and Dey, N. (2015). Counting all possible simple paths using artificial cell division mechanism for directed acyclic graphs.

[Chen et al., 2018] Chen, X., Li, L.-J., Fei-Fei, L., and Gupta, A. (2018). Iterative visual reasoning beyond convolutions. In *CVPR*, pages 7239–7248.

[Chu et al., 2023] Chu, P., Wang, J., You, Q., Ling, H., and Liu, Z. (2023). Transmot: Spatial-temporal graph transformer for multiple object tracking. In *WACV*, pages 4870–4880.

[Dai et al., 2021] Dai, P., Weng, R., Choi, W., Zhang, C., He, Z., and Ding, W. (2021). Learning a proposal classifier for multiple object tracking. In *CVPR*.

[Dave et al., 2020] Dave, A., Khurana, T., Tokmakov, P., Schmid, C., and Ramanan, D. (2020). Tao: A large-scale benchmark for tracking any object. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 436–454. Springer.

[De Cao and Kipf, 2019] De Cao, N. and Kipf, T. (2019). Molgan: An implicit generative model for small molecular graphs. arxiv 2018. In *ICML*. DOI.

[Dendorfer et al., 2020] Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., and Leal-Taixé, L. (2020). Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.

[Dendorfer et al., 2022] Dendorfer, P., Yugay, V., Osep, A., and Leal-Taix'e, L. (2022). Quo vadis: Is trajectory forecasting the key towards long-term multi-object tracking? *ArXiv*, abs/2210.07681.

[Duan et al., 2019] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., and Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *ICCV*, pages 6569–6578.

[Ess et al., 2008] Ess, A., Leibe, B., Schindler, K., and Gool, L. V. (2008). A mobile vision system for robust multi-person tracking.

[Fabbri et al., 2021] Fabbri, M., Brasó, G., Maugeri, G., Cetintas, O., Gasparini, R., Osep, A., Calderara, S., Leal-Taixé, L., and Cucchiara, R. (2021). Motsynth: How can synthetic data help pedestrian detection and tracking? In *ICCV*.

[Feichtenhofer et al., 2017] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2017). Detect to track and track to detect.

[Fout et al., 2017] Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. In *NIPS*, volume 30.

[Gao et al., 2023] Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., et al. (2023). A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM*.

[Garcia and Bruna, 2017] Garcia, V. and Bruna, J. (2017). Few-shot learning with graph neural networks. In *ICLR*.

[Gilmer et al., 2017] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *ICML*, volume abs/1704.01212.

[Girbau et al., 2022] Girbau, A., Marqu'es, F., and Satoh, S. (2022). Multiple object tracking from appearance by hierarchically clustering tracklets. In *BMVC*.

[Guo et al., 2018] Guo, M., Chou, E., Huang, D.-A., Song, S., Yeung, S., and Fei-Fei, L. (2018). Neural graph matching networks for fewshot 3d action recognition.

[Guo et al., 2021] Guo, S., Wang, J., Wang, X., and Tao, D. (2021). Online multiple object tracking with cross-task synergy.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. volume 30.

[Hasani et al., 2021] Hasani, R., Lechner, M., Amini, A., Rus, D., and Grosu, R. (2021). Liquid time-constant networks. In *AAAI*, volume 35, pages 7657–7666.

[He et al., 2021] He, J., Huang, Z., Wang, N., and Zhang, Z. (2021). Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In *CVPR*.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask r-cnn. In *PAMI*.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[Held et al., 2016] Held, D., Thrun, S., and Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. In *ECCV*.

[Henschel et al., 2017] Henschel, R., Leal-Taixé, L., Cremers, D., and Rosenhahn, B. (2017). Improvements to frank-wolfe optimization for multi-detector multi-object tracking. volume abs/1705.08314.

[Hornáková et al., 2020] Hornáková, A., Henschel, R., Rosenhahn, B., and Swoboda, P. (2020). Lifted disjoint paths with application in multiple object tracking. In *ICML*.

[Hu et al., 2021] Hu, H.-N., Yang, Y.-H., Fischer, T., Darrell, T., Yu, F., and Sun, M. (2021). Monocular quasi-dense 3d object tracking. volume 45.

[Huang et al., 2013] Huang, C., Li, Y., and Nevatia, R. (2013). Multiple target tracking by learning-based hierarchical association of detection responses. volume 35, pages 898–910.

[Huynh et al., 2011] Huynh, L., Nguyen, D., Dinh, T. B., and Dinh, T. B. (2011). Online multiple object tracking by hierarchical association of detection responses.

[Hyun et al., 2023] Hyun, J., Kang, M., Wee, D., and Yeung, D.-Y. (2023). Detection recovery in online multi-object tracking with sparse graph tracker. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4850–4859.

[Jiang et al., 2019] Jiang, M., Hai, T., geng Pan, Z., Wang, H., Jia, Y., and Deng, C. (2019). Multi-agent deep reinforcement learning for multi-object tracker. *IEEE*, 7:32400–32407.

[Kawahara et al., 2017] Kawahara, J., Brown, C. J., Miller, S. P., Booth, B. G., Chau, V., Grunau, R. E., Zwicker, J. G., and Hamarneh, G. (2017). Brainnetcnn: Convolutional neural networks for brain networks; towards predicting neurodevelopment. *NeuroImage*, 146:1038–1049.

[Khurana et al., 2021] Khurana, T., Dave, A., and Ramanan, D. (2021). Detecting invisible people. In *ICCV*, pages 3174–3184.

[Kim et al., 2021] Kim, C., Fuxin, L., Alotaibi, M., and Rehg, J. M. (2021). Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9553–9562.

[Kim et al., 2015] Kim, C., Li, F., Ciptadi, A., and Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *ICCV*, pages 4696–4704.

[Kim et al., 2012] Kim, S., Kwak, S., Feyereisl, J., and Han, B. (2012). Online multi-target tracking by large margin structured learning. In *ACCV*.

[Kipf and Welling, 2016] Kipf, T. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *ARXIV*, abs/1609.02907.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *ACM*.

[Lambert and Hays, 2021] Lambert, J. and Hays, J. (2021). Trust, but verify: Cross-modality fusion for hd map change detection. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*.

[Landrieu and Simonovsky, 2018] Landrieu, L. and Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, pages 4558–4567.

[Leal-Taixé et al., 2016] Leal-Taixé, L., Canton-Ferrer, C., and Schindler, K. (2016). Learning by tracking: Siamese cnn for robust target association. In *CVPRW*.

[Leal-Taixé et al., 2014] Leal-Taixé, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., and Savarese, S. (2014). Learning an image-based motion context for multiple people tracking.

[Leal-Taixé et al., 2015] Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

[Li et al., 2022] Li, S., Kong, Y., and Rezatofighi, H. (2022). Learning of global objective for network flow in multi-object tracking.

[Li et al., 2018a] Li, Y., Ouyang, W., Zhou, B., Shi, J., Zhang, C., and Wang, X. (2018a). Factorizable net: an efficient subgraph-based framework for scene graph generation. In *ECCV*, pages 335–351.

[Li et al., 2018b] Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018b). Learning deep generative models of graphs. In *ICML*.

[Li et al., 2018c] Li, Z., Chen, Q., and Koltun, V. (2018c). Combinatorial optimization with graph convolutional networks and guided tree search. In *NIPS*, volume 31.

[Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *PAMI*.

[Luiten et al., 2019] Luiten, J., Fischer, T., and Leibe, B. (2019). Track to reconstruct and reconstruct to track. In *RAL*, volume 5, pages 1803–1810.

[Luo et al., 2021] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., and Kim, T.-K. (2021). Multiple object tracking: A literature review. *ARXIV*.

[Marcheggiani et al., 2018] Marcheggiani, D., Bastings, J., and Titov, I. (2018). Exploiting semantics in neural machine translation with graph convolutional networks. *ARXIV*.

[Marcheggiani and Titov, 2017] Marcheggiani, D. and Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. *ARXIV*.

[Meinhardt et al., 2022] Meinhardt, T., Kirillov, A., Leal-Taixé, L., and Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers. In *CVPR*.

[Milan et al., 2016a] Milan, A., Leal-Taixé, L., Reid, I. D., Roth, S., and Schindler, K. (2016a). MOT16: a benchmark for multi-object tracking. *ARXIV*, 1603.00831.

[Milan et al., 2016b] Milan, A., Rezatofighi, S. H., Dick, A. R., Reid, I. D., and Schindler, K. (2016b). Online multi-target tracking using recurrent neural networks. In *AAAI*.

[Mitzel and Leibe, 2012] Mitzel, D. and Leibe, B. (2012). Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items. In *ECCV*.

[Monti et al., 2017] Monti, F., Bronstein, M., and Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. In *NIPS*, volume 30.

[Morris et al., 2019] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, volume 33, pages 4602–4609.

[Narasimhan et al., 2018] Narasimhan, M. G., Lazebnik, S., and Schwing, A. G. (2018). Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *NIPS*.

[Ng and Russell, 2000] Ng, A. Y. and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *ICML*.

[Nguyen and Grishman, 2018] Nguyen, T. and Grishman, R. (2018). Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*, volume 32.

[Osep et al., 2017] Osep, A., Mehner, W., Mathias, M., and Leibe, B. (2017). Combined image- and world-space tracking in traffic scenes.

[Pang et al., 2021] Pang, J., Qiu, L., Li, X., Chen, H., Li, Q., Darrell, T., and Yu, F. (2021). Quasi-dense similarity learning for multiple object tracking. In *CVPR*.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[Pellegrini et al., 2009] Pellegrini, S., Ess, A., Schindler, K., and Gool, L. V. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking.

[Peng et al., 2020] Peng, J., Wang, C., Wan, F., Wu, Y., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., and Fu, Y. (2020). Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking.

[Qi et al., 2018] Qi, S., Wang, W., Jia, B., Shen, J., and Zhu, S.-C. (2018). Learning human-object interactions by graph parsing neural networks. In *ECCV*, pages 401–417.

[Qi et al., 2017] Qi, X., Liao, R., Jia, J., Fidler, S., and Urtasun, R. (2017). 3d graph neural networks for rgbd semantic segmentation. In *ICCV*, pages 5199–5208.

[Qin et al., 2023] Qin, Z., Zhou, S., Wang, L., Duan, J., Hua, G., and Tang, W. (2023). Motiontrack: Learning robust short-term and long-term motions for multi-object tracking. *ArXiv*, abs/2303.10404.

[Qiu et al., 2018] Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., and Tang, J. (2018). Deepinf: Social influence prediction with deep learning. In *SIGKDD*, pages 2110–2119.

[Redmon and Farhadi, 2016] Redmon, J. and Farhadi, A. (2016). Yolo9000: Better, faster, stronger. In *CVPR*, pages 6517–6525.

[Ren et al., 2023] Ren, H., Han, S., Ding, H., Zhang, Z., Wang, H., and Wang, F. (2023). Focus on details: Online multi-object tracking with diverse fine-grained representation. *ArXiv*, abs/2302.14589.

[Ren et al., 2018] Ren, L., Lu, J., Wang, Z., Tian, Q., and Zhou, J. (2018). Collaborative deep reinforcement learning for multi-object tracking. In *ECCV*.

[Ren et al., 2015] Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *PAMI*.

[Ristani and Tomasi, 2018] Ristani, E. and Tomasi, C. (2018). Features for multi-target multi-camera tracking and re-identification. In *CVPR*.

[Rosello and Kochenderfer, 2018] Rosello, P. and Kochenderfer, M. J. (2018). Multi-agent reinforcement learning for multi-object tracking. In *AAM*.

[Sadeghian et al., 2017] Sadeghian, A., Alahi, A., and Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *ICCV*, pages 300–311.

[Saleh et al., 2020] Saleh, F. S., Aliakbarian, S., Rezatofighi, H., Salzmann, M., and Gould, S. (2020). Probabilistic tracklet scoring and inpainting for multiple object tracking. In *CVPR*.

[Schulman et al., 2015] Schulman, J., Heess, N. M. O., Weber, T., and Abbeel, P. (2015). Gradient estimation using stochastic computation graphs. In *NIPS*.

[Schulter et al., 2017] Schulter, S., Vernaza, P., Choi, W., and Chandraker, M. (2017). Deep network flow for multi-object tracking. In *CVPR*, pages 2730–2739.

[Shan et al., 2020] Shan, C., Wei, C., Deng, B., Huang, J., Hua, X., Cheng, X., and Liang, K. (2020). Tracklets predicting based adaptive graph tracking. In *CVPR*.

[Shao et al., 2018] Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., and Sun, J. (2018). Crowdhuman: A benchmark for detecting human in a crowd. *ARXIV*, 1805.00123.

[Sharma et al., 2018] Sharma, S., Ansari, J. A., Jatavallabhula, K. M., and Krishna, K. M. (2018). Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking.

[Shen et al., 2022] Shen, Y., Jiang, W., Xu, Z., Li, R., Kwon, J., and Li, S. (2022). Confidence propagation cluster: Unleash full potential of object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1151–1161.

[Son et al., 2017] Son, J., Baek, M., Cho, M., and Han, B. (2017). Multi-object tracking with quadruplet convolutional neural networks. pages 3786–3795.

[Stadler and Beyerer, 2023] Stadler, D. and Beyerer, J. (2023). An improved association pipeline for multi-person tracking. In *CVPR*.

[Sun et al., 2022] Sun, P., Cao, J., Jiang, Y., Yuan, Z., Bai, S., Kitani, K., and Luo, P. (2022). Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20993–21002.

[Sun et al., 2020] Sun, P., Jiang, Y., Zhang, R., Xie, E., Cao, J., Hu, X., Kong, T., Yuan, Z., Wang, C., and Luo, P. (2020). Transtrack: Multiple-object tracking with transformer. *ArXiv*, abs/2012.15460.

[Te et al., 2018] Te, G., Hu, W., Zheng, A., and Guo, Z. (2018). Rgcnn: Regularized graph cnn for point cloud segmentation. In *ACM*, pages 746–754.

[Thekumparampil et al., 2018] Thekumparampil, K. K., Wang, C., Oh, S., and Li, L.-J. (2018). Attention-based graph neural network for semi-supervised learning. *ARXIV*.

[Tokmakov et al., 2021] Tokmakov, P., Li, J., Burgard, W., and Gaidon, A. (2021). Learning to track with object permanence.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*.

[Veličković et al., 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks.

[Wang et al., 2021a] Wang, Q., Zheng, Y., Pan, P., and Xu, Y. (2021a). Multiple object tracking with correlation learning. In *CVPR*, pages 3876–3886.

[Wang and Fowlkes, 2015] Wang, S. and Fowlkes, C. C. (2015). Learning optimal parameters for multi-target tracking. In *BMVC*.

[Wang et al., 2022] Wang, S., Yang, D., Wu, Y., Liu, Y., and Sheng, H. (2022). Tracking game: Self-adaptative agent based multi-object tracking. In *ACM*.

[Wang et al., 2021b] Wang, Y., Kitani, K., and Weng, X. (2021b). Joint object detection and multi-object tracking with graph neural networks.

[Wang et al., 2019a] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019a). Dynamic graph cnn for learning on point clouds. *ACM*, 38(5):1–12.

[Wang et al., 2019b] Wang, Z., Zheng, L., Liu, Y., and Wang, S. (2019b). Towards real-time multi-object tracking. *ArXiv*, abs/1909.12605.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

[Wilson et al., 2021] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., Ramanan, D., Carr, P., and Hays, J. (2021). Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*.

[Wojke et al., 2017] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *ICIP*.

[Wu et al., 2021] Wu, J., Cao, J., Song, L., Wang, Y., Yang, M., and Yuan, J. (2021). Track to detect and segment: An online multi-object tracker.

[Wu et al., 2022] Wu, W., Liu, J., Zheng, K., Sun, Q., and Zha, Z. (2022). Temporal complementarity-guided reinforcement learning for image-to-video person re-identification. In *CVPR*.

[Wu et al., 2020] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE*, 32(1):4–24.

[Xiang et al., 2015] Xiang, Y., Alahi, A., and Savarese, S. (2015). Learning to track: Online multi-object tracking by decision making. In *ICCV*.

[Xiao et al., 2018] Xiao, B., Wu, H., and Wei, Y. (2018). Simple baselines for human pose estimation and tracking. *ARXIV*, abs/1804.06208.

[Xu et al., 2017] Xu, D., Zhu, Y., Choy, C. B., and Fei-Fei, L. (2017). Scene graph generation by iterative message passing. In *CVPR*, pages 5410–5419.

[Xu et al., 2019] Xu, J., Cao, Y., Zhang, Z., and Hu, H. (2019). Spatial-temporal relation networks for multi-object tracking. In *ICCV*, pages 3987–3997.

[Xu et al., 2021] Xu, Y., Ban, Y., Delorme, G., Gan, C., Rus, D., and Alameda-Pineda, X. (2021). Transcenter: Transformers with dense representations for multiple-object tracking. In *PAMI*.

[Yang and Nevatia, 2012] Yang, B. and Nevatia, R. (2012). An online learned crf model for multi-target tracking. In *CVPR*, pages 2034–2041.

[Yang et al., 2016] Yang, F., Choi, W., and Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, pages 2129–2137.

[Yang et al., 2022] Yang, F., Odashima, S., Masui, S., and Jiang, S. (2022). Hard to track objects with irregular motions and similar appearances? make it easier by buffering the matching space. In *WACV*.

[Yang et al., 2018] Yang, J., Lu, J., Lee, S., Batra, D., and Parikh, D. (2018). Graph r-cnn for scene graph generation. In *ECCV*, pages 670–685.

[Yang et al., 2019a] Yang, L., Fan, Y., and Xu, N. (2019a). Video instance segmentation.

[Yang et al., 2019b] Yang, L., Zhan, X., Chen, D., Yan, J., Loy, C. C., and Lin, D. (2019b). Learning to cluster faces on an affinity graph. In *CVPR*, pages 2293–2301.

[Yi et al., 2017] Yi, L., Su, H., Guo, X., and Guibas, L. J. (2017). Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 2282–2290.

[Ying et al., 2018] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, pages 974–983.

[You et al., 2018] You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. In *NIPS*, volume 31.

[You et al., 2023] You, S., Yao, H., Bao, B.-K., and Xu, C. (2023). Utm: A unified multiple object tracking model with identity-aware feature enhancement. In *CVPR*, pages 21876–21886.

[Yu et al., 2022] Yu, E., Li, Z., and Han, S. (2022). Towards discriminative representation: Multi-view trajectory contrastive learning for online multi-object tracking. In *CVPR*, pages 8834–8843.

[Yu et al., 2020] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645.

[Yu et al., 2016] Yu, F., Li, W., Li, Q., Liu, Y., Shi, X., and Yan, J. (2016). Poi: Multiple object tracking with high performance detection and appearance feature. *ARXIV*, abs/1610.06136.

[Yu et al., 2018] Yu, F., Wang, D., Shelhamer, E., and Darrell, T. (2018). Deep layer aggregation. In *CVPR*, pages 2403–2412.

[Yun et al., 2017] Yun, S., Choi, J., Yoo, Y. J., Yun, K., and Choi, J. Y. (2017). Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, pages 1349–1358.

[Zeng et al., 2021] Zeng, F., Dong, B., Wang, T., Chen, C., Zhang, X., and Wei, Y. (2021). Motr: End-to-end multiple-object tracking with transformer. *ArXiv*, abs/2105.03247.

[Zeng et al., 2020] Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. K. (2020). Graphsaint: Graph sampling based inductive learning method. *ArXiv*, abs/1907.04931.

[Zhang et al., 2008] Zhang, L., Li, Y., and Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *CVPR*.

[Zhang et al., 2022] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box. In *ECCV*.

[Zhang et al., 2020] Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2020). Fairmot: On the fairness of detection and re-identification in multiple object tracking. volume 129.

[Zhou et al., 2020] Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. *ArXiv*, abs/2004.01177.

[Zhou et al., 2022] Zhou, X., Yin, T., Koltun, V., and Krähenbühl, P. (2022). Global tracking transformers. In *CVPR*.

[Zhu et al., 2018] Zhu, J., Yang, H., Liu, N., Kim, M., Zhang, W., and Yang, M.-H. (2018). Online multi-object tracking with dual matching attention networks. In *ECCV*.