GRAPH-BASED HIERARCHICAL TRACKLET MERGE FOR MULTIPLE OBJECT TRACKING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HALIL ÇAĞRI BILGI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

APRIL 2024

Approval of the thesis:

GRAPH-BASED HIERARCHICAL TRACKLET MERGE FOR MULTIPLE OBJECT TRACKING

submitted by **HALIL ÇAĞRI BILGI** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun Dean, Graduate School of **Natural and Applied Sciences**	__________
Prof. Dr. İlkay Ulusoy Head of Department, **Electrical and Electronics Engineering**	__________
Prof. Dr. A. Aydın Alatan Supervisor, **Electrical and Electronics Engineering, METU**	__________

Examining Committee Members:

Prof. Dr. Umut Orguner Electrical and Electronics Eng., METU	__________
Prof. Dr. A. Aydın Alatan Electrical and Electronics Eng., METU	__________
Prof. Dr. Hakan Çevikalp Electrical and Electronics Eng., Osmangazi University	__________
Assist. Prof. Dr. Elif Vural Electrical and Electronics Eng., METU	__________
Assist. Prof. Dr. Aykut Koç Electrical and Electronics Eng., Bilkent University	__________

Date:05.04.2024

ABSTRACT

GRAPH-BASED HIERARCHICAL TRACKLET MERGE FOR MULTIPLE OBJECT TRACKING

Bilgi, Halil Çağrı
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. A. Aydın Alatan

April 2024, 82 pages

The past decade has seen significant advancements in multi-object tracking, particularly with the rise of deep learning. However, many studies in online tracking have primarily focused on enhancing track management or extracting visual features, often leading to hybrid approaches with limited effectiveness, especially in scenarios with severe occlusions or crowded scenes. Conversely, in offline tracking, there has been a lack of emphasis on robust motion cues. This thesis proposes a novel solution to offline tracking by hierarchically merging tracklets, leveraging recent promising learning-based architectures. Our approach integrates motion cues and social interactions among targets using a joint Transformer and Graph Neural Network (GNN) encoder. The proposed solution is an end-to-end trainable model that does not require any handcrafted short-term or long-term matching processes. By representing tracklets across multiple frames using a graph structure, we enable collective reasoning of targets across different timestamps, leveraging advancements in graph representation learning. Furthermore, the Transformer encoder effectively captures the motion of each tracklet. By enabling bi-directional information propagation between these modalities, namely Transformer and GNN, we allow motion modeling to depend on interactions and, conversely, interaction modeling to depend on the motion of each target. Experimental results demonstrate the effectiveness of our approach, indicating that graph representation learning equipped with a joint Transformer encoder achieves results comparable to the state-of-the-art algorithms. These promising results emphasize the potential of the joint Transformer-GNN encoder architecture in multi-object tracking.

Keywords: Multi-object Tracking, Offline Tracking, Graph Representation Learning, Graph Neural Networks, Transformer

ÖZ

ÇOKLU HEDEF TAKİBİ İÇİN ÇİZGE TABANLI HİYERARŞİK İZ BİRLEŞTİRME

Bilgi, Halil Çağrı
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Nisan 2024 , 82 sayfa

Geçtiğimiz on yılda, özellikle derin öğrenmenin yükselişiyle birlikte çoklu nesne takibinde önemli ilerlemeler görüldü. Bununla birlikte, çevrimiçi izlemeyle ilgili birçok çalışma öncelikli olarak iz yönetimini geliştirmeye veya daha etkin görsel özellikleri çıkarmaya odaklanmıştır; bu da özellikle yoğun görsel örtmelerin veya kalabalık sahnelerin olduğu senaryolarda genellikle sınırlı etkililiğe sahip hibrit yaklaşımlara yol açmaktadır. Bunun yanında, çevrimdışı izleme metodları etkin iz yönetimine genellikle gereken önemi vermemiştir. Bu tez, son zamanlarda etkili olduğu gözlemlenen öğrenmeye dayalı mimarilerden yararlanarak, hedef izlerini hiyerarşik olarak birleştirerek çevrimdışı çoklu hedef takibine yeni bir çözüm önermektedir. Yaklaşımımız, ortak bir Transformer ve Çizge Sinir Ağı (Graph Neural Network) kodlayıcı kullanarak hedefler arasındaki hareket ipuçlarını ve sosyal etkileşimleri entegre eder. Herhangi bir varsayıma dayalı kısa veya uzun vadeli eşleştirme süreci gerektirmeyen, uçtan uca eğitilebilir bir modeldir. Önerdiğimiz model bir çizge yapısı kullanarak çoklu çerçevelerdeki hedef izlerini temsil eder, ve farklı zaman damgalarındaki hedeflerin kolektif olarak ele alınmasına olanak tanır. Ayrıca Transformer kodlayıcı, her bir hedefin hareket karakteristiğini etkili bir şekilde modellenmesine olanak tanır. Transformer ve GNN mimarileri arasında çift yönlü bilgi akışını etkinleştirerek, hareket modellemenin etkileşimlere bağlı olmasına ve bunun tersine etkileşim modellemenin de her hedefin hareketine bağlı olmasını sağlıyoruz. Deneysel sonuçlar, yaklaşımımızın etkinliğini ortaya koymakta ve ortak bir Transformer kodlayıcı ile donatılmış GNN mimarisinin, en son teknoloji algoritmalarla karşılaştırılabilir sonuçlar elde ettiğini göstermektedir. Bu umut verici sonuçlar, çoklu nesne takibinde ortak Transformer-GNN kodlayıcı mimarisinin potansiyelini vurgulamaktadır.

Anahtar Kelimeler: Çoklu Hedef Takibi, Çevrimdışı Hedef Takibi, Çizge Temsili Öğrenme, Çizge Gerin Öğrenme

to my family

ACKNOWLEDGMENTS

I am profoundly grateful to my advisor, A. Aydın Alatan, for consistently standing by my side, dedicating his time, and providing unwavering support throughout the entirety of this thesis. Our discussions at every step of the way have been invaluable in shaping the course of this work.

I would like to extend my gratitude to all the researchers at OGAM for their fruitful discussions and assistance. It has been a privilege to learn and grow together.

The numerical calculations reported in this paper were fully performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

Special thanks to my parents, Mehmet and Fadime, for their unwavering support, understanding, and encouragement during this journey. Their belief in me has been a constant source of strength. I would also like to express my heartfelt gratitude to my brother, Alp Giray, for being an exceptional sibling and for his continuous encouragement in every aspect of my life.

Last but certainly not least, I extend my utmost gratitude to my beloved wife, Ebrar. Throughout this journey, her exceptional patience, presence, and support have been priceless. I feel incredibly fortunate to have you by my side, growing, learning and exploring the world together.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
MOT	Multiple Object Tracking
GNN	Graph Neural Networks
GCN	Graph Convolutional Networks
MPNN	Message Passing Neural Networks
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
MLP	Multi-Layer Perceptron
GDL	Geometric Deep Learning
LLM	Large Language Model
IoU	Intersection over Union
ReLU	Rectified Linear Unit
NLP	Natural Language Processing

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The Multi-object tracking (MOT) involves the joint localization and tracking of each object in a scene, assigning each object a unique ID. The recent decade has witnessed significant development and advancement in multi-object tracking, especially with the advent of the deep learning. The methods and algorithms developed during this time have become exceptionally powerful. With the increasing availability of computing resources, these algorithms have found extensive applications in diverse fields, including autonomous driving, surveillance systems, sports analytics, and numerous others.

Most state-of-the-art multi-object tracking algorithms [13, 9, 14, 15, 16] have adopted the tracking-by-detection paradigm to address the multi-object tracking problem, especially with the advent of powerful object detectors [17]. In the tracking-by-detection approach, the initial step involves detecting every object in each frame of the scene using a robust, pre-trained object detector. Subsequently, the tracking process focuses on associating these detected bounding boxes across the temporal domain to extract plausible target trajectories. This temporal association is performed either *online*, on a frame-by-frame basis in real-time, or *offline*, where the entire video sequence is available from the beginning. In both cases, the problem of MOT involves data association, where multiple observations (or tracks) need to be assigned to multiple active object tracks to optimize some global criterion of choice. The task is challenging on its own due to missing or false detections, occlusions, and interactions between objects. In order to overcome those challenges, strong motion and visual models are required for long-term tracking.

To develop robust motion models, recent studies have embraced Kalman-filter-based approaches [18, 14, 5, 19]. These models are employed for real-time tracking of multiple objects, optimizing the matching cost between observations and the predicted positions of each object on a frame-by-frame basis. Nevertheless, in crowded scenes, these models often encounter challenges, particularly when faced with severe occlusions or arbitrary movements resulting from interactions among objects. In order to address some of these challenges, researchers have equipped these models with learned visual cues. This involves utilizing pre-trained CNN architectures to extract visual descriptors of the target [20], or employing Siamese networks trained to capture similarities between different poses of the same target [21] to recover the lost tracks. Notably, a recent study [22] suggests that even a simple linear motion model can perform comparably to state-of-the-art trackers, especially when equipped with robust visual cues which signifies the importance of the visual ques. On the other hand, the research also implies that using complex motion models (Kalman-filters) independently for each target in the scene does not lead to improved results. This is because such models lack the capability to capture the interactions between the targets in the scene.

Apart from those real-time *online* trackers, there is another line of research where the data association is formulated as a graph partitioning problem [23, 24, 8, 9]. In this graph formulation the detections are represented as nodes and the connections between nodes are represented as edges of the graph. The graph partitioning task can be described as classifying edges as active or inactive where former means the nodes (detections) connected by that edge belongs to the same target hence forms a trajectory, later means they belong to different targets. In previous methods, the graph partitioning problem is typically addressed using complex optimization methods that aim to minimize the global cost of matching [23] and [24]. However, in recent approaches, GNN models are employed to solve the optimization problem, introducing the use of learnable models [8].

In literature there are also some algorithms which jointly perform the detection and multi-object tracking where some of them use tracking-by-regression by altering the detection heads and regressing them in an online manner [25], some of them formulate MOT as a set prediction problem [26].

In light of this background, our study proposes a novel approach for modeling target motion. We utilize a joint transformer and Graph Neural Network (GNN) encoder to incorporate both the motion of individual targets and the interactions between targets in the scene. The model is inspired from a recent work on large language models (LLMs) fine-tuning [11], where they fine-tune the LLM model using two different data modalities such as text and knowledge graph. By allowing bidirectional information propagation between the transformer encoder and GNN they argue that the text can reason with structured information given the knowledge graph and also the knowledge graph can reason with rich context given the text data. With the adaptation of the joint encoder formulation, the dynamics of multi-object tracking are learned directly from the data using a hierarchical approach, as proposed in works such as [9] and [16].

1.2 The Outline of the Thesis

The goal of this thesis is to model the motion of the targets in the scene using a joint encoder comprising a transformer and a GNN. The transformer handles the motion of each trajectory independently, while the GNN facilitates communication among different targets to collectively understand their motion patterns.

For this purpose, we first review the preliminaries of the graph representation learning in Chapter 2 in order to familiarize the reader with the GNN model used in the proposed framework. We introduce the general GNN formulation within the Message Passing Framework to demonstrate how information propagation on graph- structured data can be formalized through neighborhood aggregation. Moreover, in Chapter 2, the formulation of the Transformer architecture is presented, drawing an analogy between fully connected graphs and Transformers. At the end of that section, examples of downstream tasks are presented to illustrate how those learned representations are utilized.

In Chapter 3, a thorough and up-to-date review of the literature on Multiple Object

Tracking (MOT) methods is conducted. The shortcomings of the existing methods are analyzed and discussed. In Chapter 4, we introduce a new framework based on joint transformer and GNN architecture. We justify the contributions of the proposed method through experimentation on real-world datasets, comparing our model with state-of-the-art multi-object trackers.

Finally, in Chapter 5, the thesis is summarized by discussing the contributions and comparisons revealed through the experimental results.

CHAPTER 2

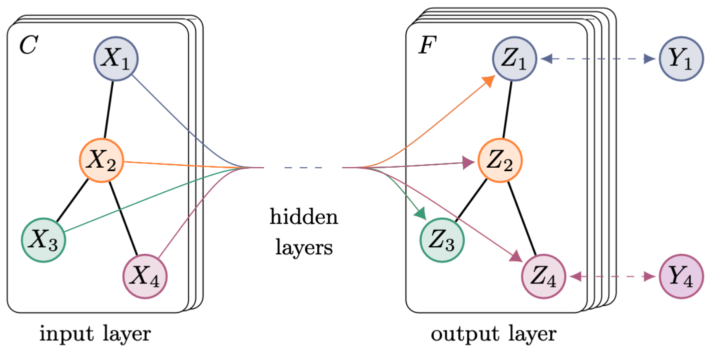
AN OVERVIEW OF GRAPH REPRESENTATION LEARNING

2.1 Introduction

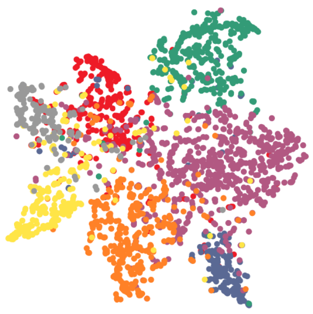
This chapter aims to provide a brief overview of recent machine learning techniques and methods developed for graph-structured data by introducing fundamental concepts and preliminaries. In recent years, a growing interest in graph representation learning has led to the emergence of numerous methods and algorithms designed to exploit the inherent relational structure within certain data modalities. With convolutional neural networks marking significant milestones in machine learning, the next significant leap forward is occurring in graph representation learning. This shift is propelled by the recognition that graphs, unlike ordered data structures such as grids or sequences, abound in many datasets we generate or consume. As a result, various types of data can be reduced to simple entities connected by relationships, allowing them to be represented as graphs. This approach has made learning distinctive features from these graph structures a key focus in machine learning research.

While certain data modalities may not naturally form a graph in their underlying structure, leveraging the dynamic and flexible nature of graph structures through the use of heuristics and distance-based methodologies allows the creation of connections among different entities within the underlying data manifold, effectively forming a graph. This approach enables the exploitation of information propagation between entities, thereby aiding in the discovery of an appropriate structure to represent the underlying data manifold and apply machine learning methods effectively.

Similar to any other deep learning architecture, the primary objective of Graph Neural Networks (GNN) is to ingest and derive useful high-level representations of the infor-



(a) Graph Convolutional Network



(b) Hidden layer activations

Figure 2.1: Graph Convolutional Network illustration [2]

mation contained in the raw features of graph-structured data given as input. These learned features are then utilized in downstream tasks. An illustration is provided in Figure 2.1.

In this chapter, we begin by introducing the general GNN formulation within the Message Passing Framework to demonstrate how information propagation on graph-structured data can be formalized through neighborhood aggregation. The theoretical foundations of such models are emphasized in Appendix A. As we approach the end of the chapter, we draw a comparison between graph representation learning and the transformer architecture. Finally, we explore the practical applications of the acquired discriminative features in tasks such as node classification and edge prediction.

2.2 Message Passing Neural Networks

Gilmer [27] introduced a general framework for designing GNNs, unifying existing GNN models into a single common framework called Message Passing Neural Networks (MPNNs). MPNNs offer a flexible formulation capable of generating various models based on two key components: (1) the message function and (2) the aggregation function (vertex update function). These MPNN models are applicable to graphs with node and even edge features, without significantly increasing computation and memory complexity.

Let $G(\mathcal{V}, \mathcal{E})$ is a undirected graph, with node features $h_i \in \mathbb{R}^d$ and edge features

$e_{ij} \in \mathbb{R}^k$, during each message passing phase the hidden states $h_i^{(t)}$ of each node are updated based on the messages $m_i^{(t)}$ of its neighboring nodes [27].

$$\begin{aligned} m_i^{(t+1)} &= \{\mathbf{M}_t(h_i^{(t)}, h_j^{(t)}, e_{ij}^{(t)}) \mid j \in \mathcal{N}(i)\} \\ h_i^{(t+1)} &= \mathbf{U}_t(h_i^{(t)}, m_i^{(t+1)}) \end{aligned} \tag{2.1}$$

The message function $\mathcal{M}_t$ is a learnable and differentiable function, while the aggregation function $\mathcal{U}_t$ operates over multi-sets since messages coming from neighboring nodes form a multi-set. Consequently, permutation-invariant functions—such as sum, mean, and max— are preferred as an aggregation function to preserve the inductive bias of the model [28], ensuring that the order of neighbors does not affect the outcome. However, learned differentiable functions can also be utilized in this context. The design of both functions can be adjusted according to the specific task requirements. Additionally, in Equation 2.1, the utilization of edge features is limited to the message computation. However, this formulation can be expanded to update the hidden states of the edge features as well.

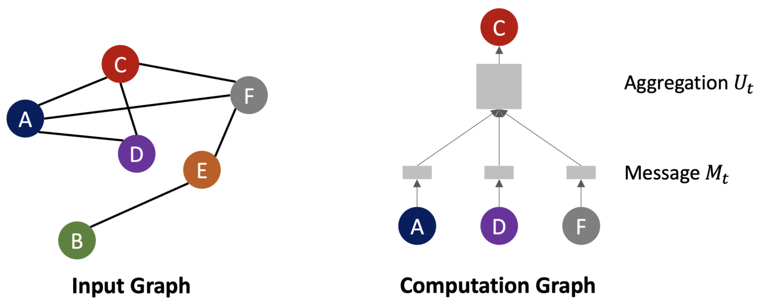


Figure 2.2: Example of Message Passing Scheme.

An illustrative example, depicted in Figure 2.2, clarifies this concept. On the left side, we observe a simple input graph, and the aim is to propagate information throughout the graph and subsequently updating the hidden states of individual nodes. This update process is demonstrated for the target node **C** in the right-hand figure. In Figure 2.2, neighboring nodes of **C** transmit messages with message function, which are then aggregated with aggregate function, resulting in the update of vertex **C**. Note that in each message passing phase, both functions, $\mathcal{M}_t$ and $\mathcal{U}_t$, are shared across the entire graph. Just like in CNNs, the filters are shared across locations but also well localized in the graph domain.

In the following subsections, we will review some of the most widely utilized models, exploring their contributions and delving into their formulations.

2.2.1 GCN

In this subsection, the formulation of the Graph Convolutional Network (GCN) [2] model in the MPNN framework is provided. In this formulation the convolutional filters are well localized across the graph, and shared among different locations.

$$\begin{aligned} m_i^{(t+1)} &= \{h_j^{(t)}\mathbf{W}^{(t)} \mid j \in \mathcal{N}(i) \cup i\} \\ h_i^{(t+1)} &= \sigma\big(\sum_{j \in \mathcal{N}(i) \cup i} \frac{1}{\sqrt{d_i}\sqrt{d_j}} m_j^{(t+1)}\big) \end{aligned} \tag{2.2}$$

where $\mathbf{W}^{(t)}$ is the learned differentiable weight matrix of t^{th} message passing phase, and d_i is the degree of node i. Equation 2.2 shows how the state of a node is updated in one GCN layer. The convolutional filters used are well localized and shared across the graph.

2.2.2 GAT

Graph Attention Network (GAT) [3] represents an extension of the attention mechanism into message-passing neural networks designed for graph-structured data. Since its initial proposal in the machine translation domain by [29], the attention mechanism has played a dominant role in various fields, yielding successful applications. The transformer architecture, as introduced by [4], exclusively employs attention in each layer, marking its significance not only in natural language processing but also in various other application domains [30, 31, 32, 33].

GAT [3] proposes to compute the hidden states of each node in the graph by attending to its 1-hop neighbors, employing a self-attention strategy. An illustration of one GAT layer is provided in Figure 2.3. During the message-passing phase, the node update

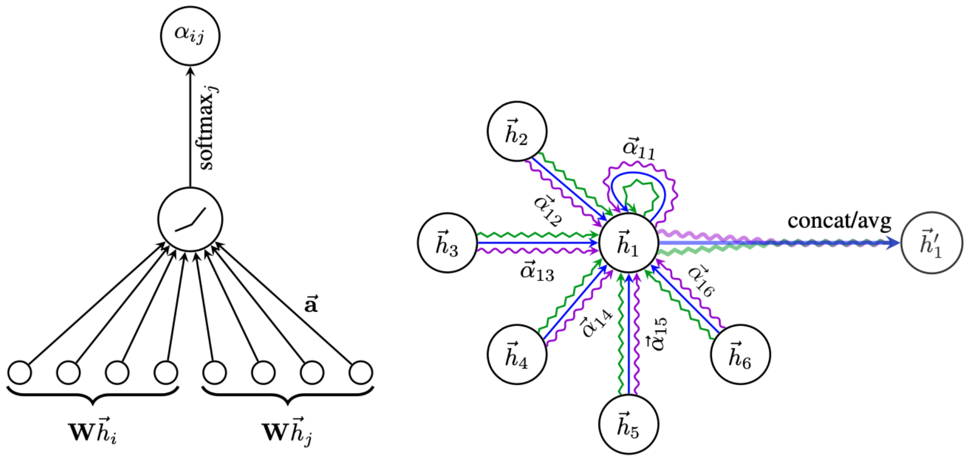


Figure 2.3: Graph Attention Network illustration [3]

is executed for each node through a weighted sum of messages from its neighbors in the graph. The message and aggregation functions are formulated as follows:

$$
\begin{aligned}
m_i^{(t+1)} &= \{h_j^{(t)}\mathbf{W}^{(t)} \mid j \in \mathcal{N}(i) \cup i\} \\
h_i^{(t+1)} &= \sigma\big(\sum_{j \in \mathcal{N}(i) \cup i} \alpha_{ij}^{(t)}\, m_j^{(t+1)}\big)
\end{aligned}
\tag{2.3}
$$

In Equation 2.3, the node aggregation function updates the state of the node through a weighted sum of the messages received from its neighbors. The attention weights for these messages are calculated in each message-passing phase (the subscript (t) is omitted for simplification) as follows:

$$
\alpha_{ij} = \frac{\exp\left(\sigma\left(\mathbf{a}^T[\mathbf{W}h_i || \mathbf{W}h_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\sigma\left(\mathbf{a}^T[\mathbf{W}h_i || \mathbf{W}h_k]\right)\right)}
\tag{2.4}
$$

where $||$ is concatenation operation, the attention mechanism is a single-layer feedforward neural network, parametrized by a weight vector $\mathbf{a} \in \mathbb{R}^{2d'}$. The attention coefficient is computed exclusively for node pairs connected within their 1-hop neighbors. These attention coefficients are normalized among the neighbors for each node using the softmax function. This formulation can also be extended to *multi-head attention*, as presented in [4], where K independent attention mechanisms execute the transformation of (2.3), an illustration of multi-head attention (with K = 3 heads) is given in the right hand side of Figure 2.3.

However, one of the shortcomings of this formulation is that for any query node, the attention function is *monotonic* with respect to the neighbor (key) scores, as discussed in [34]. The main problem that limits the expressive power of the GAT model is that the layers $\mathbf{W}$ and $\mathbf{a}$ are applied consecutively, and thus can be collapsed into a single layer.

$$\alpha_{ij} = \frac{\exp\left(\mathbf{a}^T \sigma\left(\mathbf{W}[h_i || h_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\mathbf{a}^T \sigma\left(\mathbf{W}[h_i || h_k]\right)\right)} \tag{2.5}$$

Brody et al. [34] proposed that changing the order of operations in the original GAT formulation, Equation 2.5, allows for a more expressive model representation, leveraging the universal approximation theorem of multi-layer perceptrons. This new model is known as GATv2 [34]. In our proposed method, presented in Chapter 4, we opted to use a similar formulation to GATv2 model.

2.2.3 GraphSAGE

The models discussed in the previous subsections, require all nodes of the graph to be present during the forward pass. However, these formulations are impractical for real-world large graphs as they might not fit into the GPU memory. Additionally, since they operate on complete graphs, they inherently exhibit *transductive* behavior. This results in a drop in performance when new nodes are added to the graph or when the model is applied to previously unseen graphs. Moreover, to compute the hidden states of newly added nodes, the entire graph needs to be fed to the model, leading to excessive computations each time a new node is introduced.

To overcome the aforementioned limitations Hamilton [12] proposed a general *inductive* framework that utilizes node feature information to efficiently generate node embeddings for previously unseen data by learning a function that samples and aggregates features from a node's local neighborhood. Therefore, instead of using the full graph to train the model, they perform neighborhood sampling by selecting a fixed-size neighborhood around each target node during the training process. Specifically, for each node in the graph, a fixed-size sample of its neighbors is randomly selected. The model then aggregates the features from this sampled neighborhood to generate embeddings for the target node. Since model do not dependent on the whole graph to be present at the forward pass, the embeddings of the newly added nodes can be calculated with the same sampling approach. The formulation of the GraphSAGE [12] model is given below,

$$\begin{aligned} m_i^{(t+1)} &= \{h_j^{(t)} \mid j \in \mathcal{N}(i) \cup i\} \\ h_i^{(t+1)} &= \sigma\big(\sum_{j \in \mathcal{N}(i) \cup i} \mathbf{W}^{(t)} \, [h_i^{(t)} || m_j^{(t+1)}]\big) \end{aligned} \tag{2.6}$$

where the symbol || is the concatenation operation.

2.3 Transformers

Since the introduction of the widely recognized Transformer architecture by Vaswani in their seminal paper [4], nearly every subfield of deep learning has attempted to incorporate this architecture into their designs. Subsequently, not only natural language processing but also other domains, such as computer vision [30, 33, 15], multi-model processing [35], etc., have witnessed successful applications of the Transformer architecture. In this section, our aim is to introduce the Transformer architecture and explain its fundamental working principles.

Transformers are neural network models initially designed to process sequential data in machine translation, such as sentences. With the self-attention mechanism, model can effectively capture long-range dependencies among its inputs by assigning relative importance weights to different inputs. Beyond this significant capability, the Transformer architecture is highly parallelizable, enabling the model to be scaled to extensive sizes.

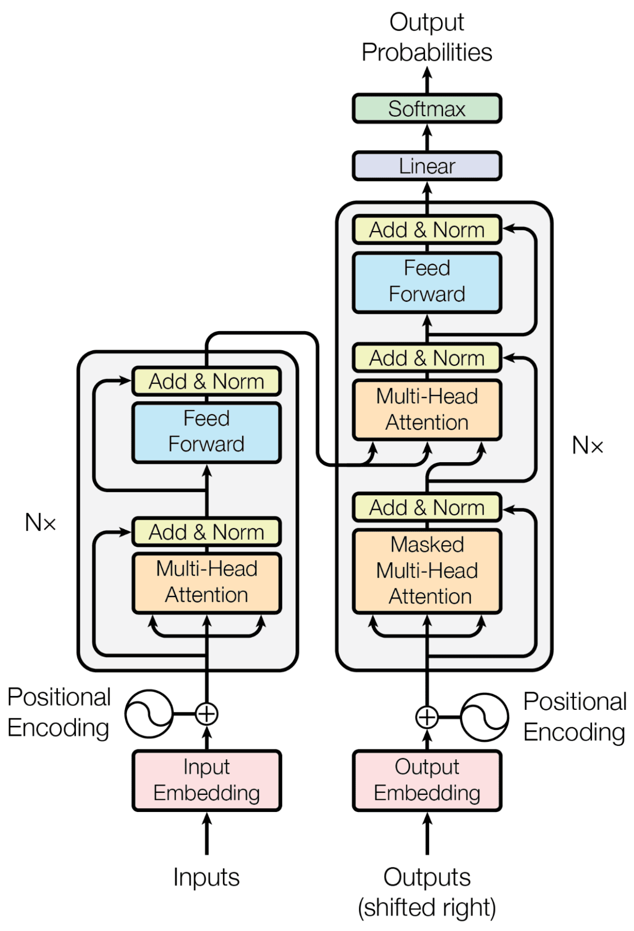


Figure 2.4: The vanilla transformer architecture [4].

In this section, we will introduce the original Transformer architecture as shown in Figure 2.4 proposed by Vaswani [4] and then we will make an analogy about the formulation of Transformers and GNNs.

2.3.1 Review of Vanilla Transformer Architecture

The model proposed by Vaswani [4] was specifically designed for machine translation tasks, featuring both encoder and decoder blocks as shown in Figure 2.4. For instance, the encoder receives inputs in English, while the decoder generates translations into French, producing one word at a time as an autoregressive model.

The architecture comprises three essential components: (1) dot-product attention, (2) feed-forward layers and (3) positional embeddings. Additionally, normalization layers follow both the attention and feed-forward layers. In the encoder, the inputs are the sequences, with each element of the sequence at each timestamp referred to as a *token*. These tokens can be conceptualized as feature vectors.

Dot-Product Attention: Various methods for calculating attention scores exist in the literature, including multiplicative or additive approaches. However, the authors opted for dot-product attention calculation as it can be calculated with only matrix multiplications, making it highly parallelizable and scalable.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.7}$$

In Equation 2.7, Q, K, and V represent matrices formed by linear transformations of the input sequence, where d_k is the dimension of the features. The term $softmax(\frac{QK^T}{\sqrt{d_k}})$ computes the relative importance between input tokens, referred to as attention weights. Subsequently, information propagation occurs between tokens through a weighted sum of their neighboring tokens. Normalizing the attention weights with $\sqrt{d_k}$ stabilizes the training process by smoothing the output of the softmax function.

The attention calculation provided in Equation 2.7 is based on a single head, but it can be extended to multiple heads, enabling each token to attend to different tokens across various heads. This approach enhances the flexibility of the model to assign different attention weights to multiple tokens simultaneously within the same layer. Additionally, since attention is computed using dot-product attention, the initial iteration of training may result in meaningless attention calculations due to random initialization, which can lead to unstable and challenging training. Utilizing multi-head attention addresses this issue of poor initialization, contributing to more stable training. After multi-head attention, values are normalized with layer normalization [36] to ensure consistent scaling of the feature representations across tokens.

Feed-Forward Layer: At each position, the feature vectors are independently processed through a fully connected feed-forward network, which consists of two linear transformations with a ReLU activation function in between. This layer serves two main purposes: firstly, it increases the model capacity by further propagating features through the network; secondly, it addresses the scaling issue of the features. The dimensions of the two-layer multi-layer perceptrons (MLPs) are initially increased to four times the dimension of each token's features and then reduced back to the original token dimension. Throughout various implementations of Transformer architectures across different domains, this choice of hyper-parameter often proves effective without requiring additional tuning. Although not explicitly mentioned in the paper, it can be argued that the scaling problem of the features persists even after the normalization layers. Therefore, the feed-forward layer serves to further mitigate the scaling problem and enable stable training.

Positional Encoding: Given that the model processes all tokens simultaneously without inherent positional understanding during information propagation and attention calculation, tokens lack awareness of their order or semantic positioning relative to others. Thus, additional information regarding absolute or relative positions is necessary to be injected into the tokens in the sequence. To address this, sin and cosine positional encodings are created, each with the same length as the token feature dimension, and subsequently summed to incorporate positional information into the tokens.

$$\begin{aligned} PE_{pos,2i} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{pos,2i+1} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.8}$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The authors [4] hypothesized that this selection of positional encodings would enable the model to effectively learn to attend by relative positions. This is because, for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

2.3.2 Analogy between Transformer and Graph Neural Networks

The positional encoding is essential for injecting positional information into the input sequence. Without these positional encodings, there is no notion of order or position among the tokens within the sequence. Therefore, it can be inferred that the input tokens form a fully-connected graph structure, where each word is connected to every other word, as illustrated in Figure 2.5.

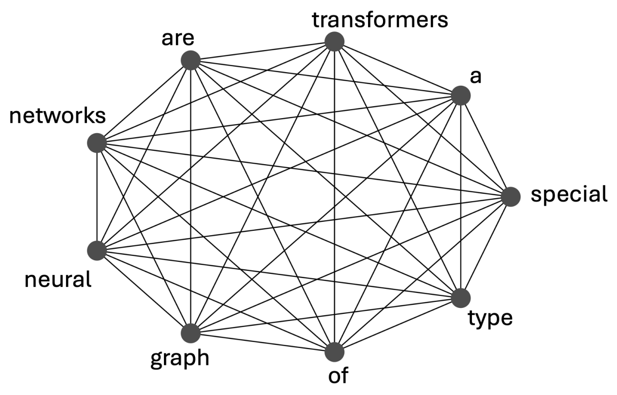


Figure 2.5: Fully-connected graph example.

We have previously discussed the formulation of Graph Attention Networks (GAT) in Section 2.2.2. Now, we can apply a similar framework to describe the architecture of the Transformer. In this message passing framework, the message function and update functions of the Transformer can be expressed as follows:

$$\begin{aligned} m_i^{(t+1)} &= \{h_j^{(t)}\mathbf{V}^{(t)} \mid j \in \mathcal{S}\} \\ h_i^{(t+1)} &= \sigma\big(\sum_{j \in \mathcal{S}} \alpha_{ij}^{(t)}\ m_j^{(t+1)}\big) \end{aligned} \tag{2.9}$$

where $\mathcal{S}$ represents the entire sequence, given that each node is connected to every other node in the sequence. The attention calculation is also formalized in a similar manner as follows:

$$\alpha_{ij} = \frac{exp\Big(\mathbf{Q}h_i \cdot \mathbf{K}h_j/\sqrt{d_k}\Big)}{\sum_{k\in\mathcal{S}} exp\Big(\mathbf{Q}h_i \cdot \mathbf{K}h_k/\sqrt{d_k}\Big)} \tag{2.10}$$

In Equation 2.10, the superscript (t) is omitted for simplicity. Equations 2.9 and 2.10 depict the attention-based message passing process in the Transformer architecture. The attention calculation can also be transformed to multi-head.

In graph terminology, Transformers aggregate messages from all nodes (words) in the graph (sequence) to propagate information in each layer. In addition to this, Transformers are equipped with layer normalizations, feed-forward layers, and most importantly, positional encodings. Moreover, training Transformer architecture is harder in general, since in real-world graphs are mostly sparse. However, it can be observed that Transformers, in essence, are a special type of graph neural networks where the neighborhood definition includes the entirety of the graph.

2.4 Graph Level Tasks

Up to this point, the review of graph neural networks has been provided. Additionally, an analogy drawn between the most successful architecture of the recent decade, Transformers, and the newly emerging models of graph neural networks. The main objective of these architectures is to extract better features from raw data and utilize them in task-specific downstream tasks to achieve high performance. Therefore, in this section, two types of downstream tasks in the graph structure data, utilizing learned representations from graph neural networks will be discussed.

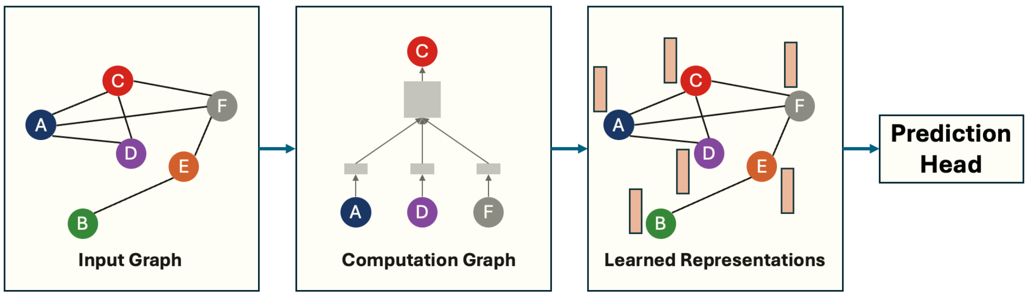


Figure 2.6: Training pipeline for Graph Neural Network models.

The training pipeline of graph neural network models is illustrated in Figure 2.6. Initially, the input graph ($G(\mathcal{V}, \mathcal{E})$) is provided to the model, assuming that only node features are present for simplicity. Subsequently, a series of message passing operations are conducted. Following the message passing process, the features of the nodes are updated to incorporate higher-level representations. These learned representations are then utilized in downstream tasks, such as (1) node/edge classification, and (2) link prediction.

2.4.1 Node/Edge Classification

Learned node features can be directly utilized for making predictions, which may involve classification or regression depending on the task at hand. For example, each node can independently undergo processing through a linear layer to predict k labels. Similarly, predictions regarding edges can also be made by utilizing node features. In Equation 2.11, we illustrate this process, where the features of the nodes connected by a particular edge are leveraged for edge classification or regression.

$$\begin{gathered}\{h_i^{(L)} \in \mathbb{R}^d, \forall v \in G\} \\ \{[h_i^{(L)}, h_j^{(L)}], \forall (i,j) \in \mathcal{E}\}\end{gathered} \tag{2.11}$$

In our research, we will utilize the edge prediction objective. The edges between nodes indicate potential connections, and our aim is to classify them as active or inactive. Active edges connect nodes belonging to the same target.

2.4.2 Link Prediction

Unlike edge classification, where existing edges in the input graph are predicted to belong to k classes, link prediction typically involves predicting non-existing edges. For instance, in social networks, individual nodes represent accounts and edges represent friendship connections between accounts. Link prediction is commonly used in friend recommendation systems. While this aspect is specific to the design choice, assuming we have node features as shown in Equation 2.11, we can proceed with the link prediction task.

$$H^L = \{[h_1 \, h_2 \, \ldots \, h_n]\}, \quad H^L \in \mathbb{R}^{d \times n} \tag{2.12}$$

We can perform a matrix multiplication ($H^T H$) of the node features, which yields the probability of having an edge between unconnected nodes.

CHAPTER 3

VISUAL MULTI-OBJECT TRACKING

In this chapter, we present a background theory of the components that are frequently used in visual MOT research and literature review of recent works relevant to the thesis. Since the groundbreaking paper by Krizhevsky et al. [37] and the subsequent advancements in deep neural networks, particularly in computer vision, progress in virtually every subfield, including multi-object tracking, has been rapid. Our aim is to comprehensively review studies relevant to our research.

In the first section, we provide an introduction to multi-object tracking, offering background information and highlighting its significance. Next, in the section of Preliminaries, we examine different components frequently used in MOT algorithms. Following this, we introduce deep learning-based approaches, where we further segment this section into subsections focusing on re-identification networks, deep Siamese networks, and graph-based MOT. We then revisit the baseline methods relevant to our research and provide a comprehensive review of the papers upon which our research is built. Finally, we conclude with a discussion, synthesizing key findings from the literature review and providing insights into potential avenues for future research.

3.1 Introduction

Multi-object tracking is a subset of object tracking that involves tracking multiple objects simultaneously within a given scene. It stands as one of the most extensively researched areas in computer vision due to its crucial role in understanding the motion patterns of dynamic objects. This field finds application in diverse areas such as surveillance, scene understanding, and autonomous driving. For instance, in autonomous driving, an autonomous driving agent must be capable of detecting and following other nearby objects to safely navigate on the road.

In recent years, with the advent of powerful object detectors [17, 33], the tracking-by-detection paradigm has become the common approach for designing multi-object tracking models. In this paradigm, tracking occurs in two stages: first, every object in each frame is detected, and then the tracking process involves assigning the same IDs to detections across different temporal axes that belong to the same object. While there are also works [25, 31] that perform joint detection and tracking, they face challenges in training due to simultaneously handling two tasks. Additionally, their larger model sizes require substantial training data. In this thesis, our research also follows the tracking-by-detection paradigm. With the availability of powerful detectors, errors due to missed detections or background detections are rare. Therefore, tracking primarily involves associating objects across frames.

In the MOT literature, trackers are often classified as *online* or *offline*. The former refers to models that are causal, meaning that the tracker can only utilize the information from current and past frames to make associations; future information from subsequent frames is not considered. Conversely, in the latter approach, a batch of frames can be used as input, allowing the utilization of future information from subsequent frames to predict the result of the current frame. However, offline tracking may not be practical in some applications, particularly those requiring real-time processing.

In this thesis, we avoid creating sections specifically dedicated to *online tracking* or *offline tracking* while reviewing the MOT literature. This decision is based on the understanding that the suitability of tracking methods often depends on the specific application and design considerations. Instead, our sections focus on general approaches commonly utilized in MOT research, such as Kalman Filter-based Approaches, Deep Learning-based Approaches, etc. While discussing these methods, we might address their applicability to online or offline tracking as relevant.

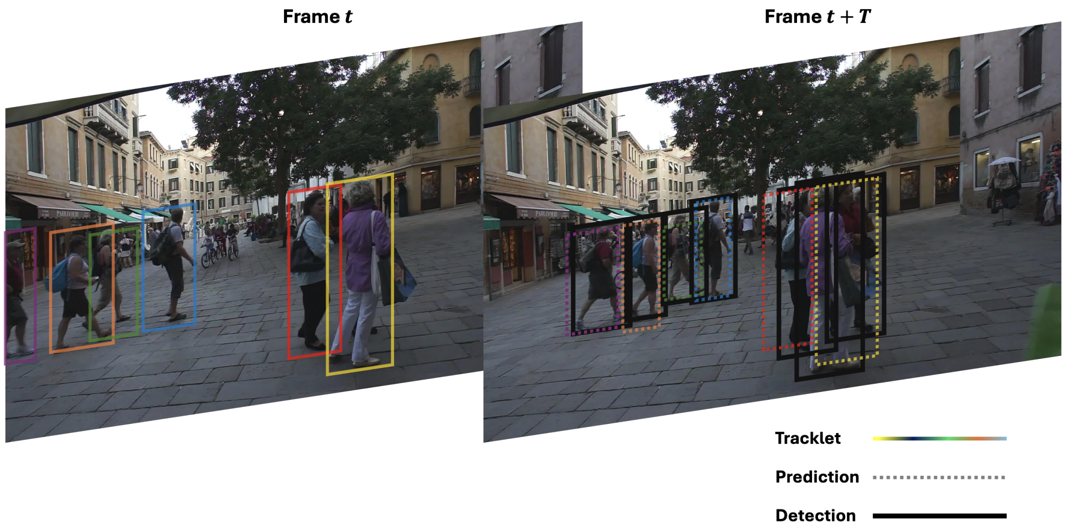


Figure 3.1: Kalman Filter based online tracking example. Colored bounding-boxes depict active tracks at frame t, dashed bounding-boxes represent the predicted positions of the tracklet at frame $t+T$, and black bounding boxes indicate the detections at frame $t+T$.

3.2 Preliminaries

3.2.1 Kalman Filter-based Approaches

Kalman filter-based tracking methods are widely used in real-time tracking scenarios due to their simplicity and effectiveness in modeling object motion. While many approaches leverage Kalman filters for object tracking, we will focus on more recent methods that employ the tracking-by-detection paradigm. An exemplary work demonstrating high performance with fast inference using Kalman filters is the SORT (Simple Online and Real-time Tracking) model proposed by Bewley et al. [18]. In this method, when assigning detections to existing targets, each target's state(position and size) in the current frame are estimated by looking at their previous states. The assignment cost matrix is then computed based on the intersection-over-union (IOU) distance, or different distance metric of choice, between each detection in the current frame and all predicted bounding boxes, in the current frame, from the existing targets as shown in Figure 3.1. Finally, this assignment problem is optimally solved using the Hungarian algorithm.

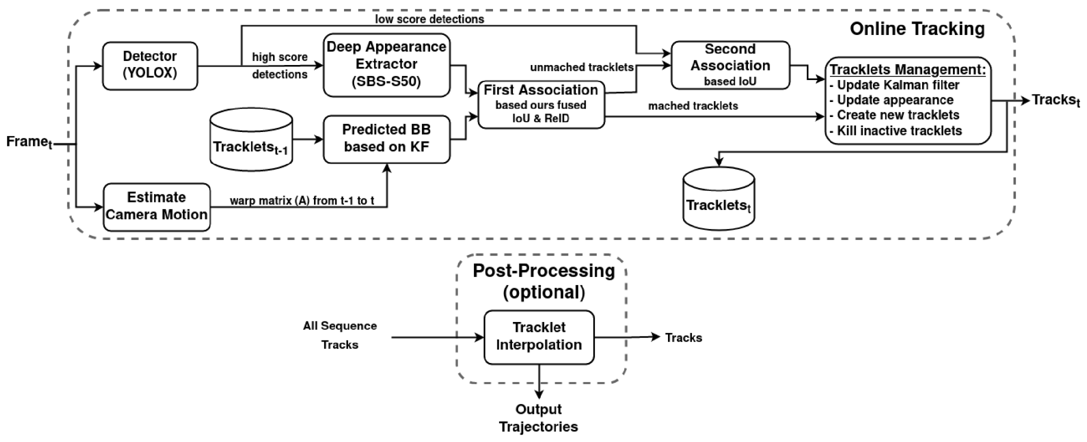


Figure 3.2: Online tracking pipeline of BoT-SORT [5].

The state of each target is represented by $[x, y, s, r, \dot{x}, \dot{y}, \dot{s}]$, where x and y denote the center position, s represents the scale (area), and r represents the aspect ratio of the bounding box. To simplify forward propagation, a linear constant velocity model is utilized.

Despite achieving state-of-the-art results at the time of its publication, the method has several limitations. Most notably, the association is solely based on the IoU distance as the cost, resulting in frequent ID switches between targets that closely interact with each other. Additionally, the authors do not incorporate mechanisms to recover occluded objects upon reappearing in the scene, nor do they address the non-rigid motions of the target. Furthermore, the method does not handle interactions and associations between numerous objects in crowded scenes.

Wojke [20] proposed a solution to address the issue of occlusion handling by equipping the model with an appearance model, which is a convolutional neural network trained to discriminate pedestrians on a large-scale person re-identification dataset. Instead of performing matching only on consecutive frames, they retain the history of the tracked object. The trajectory of the object is preserved and propagated to the current frame, allowing matching to be performed with distant frames. However, relying solely on the IoU distance becomes problematic as the occlusion time increases, since the linear motion assumption does not hold. Therefore, they also utilize the cosine distance of the visual features extracted with a pre-trained CNN, and the final distance is the weighted sum of the two.

Zhang [14] introduced ByteTrack, a Kalman-filter based tracker designed to address identity switches and occlusion issues using a novel approach. While their method shares similarities with Wojke's approach, in the tracking-by-detection paradigm, they enhance performance by employing a superior object detector. Additionally, they broaden the scope of association by considering both high and low score detections. ByteTrack implements a two-stage matching process: initially, high score detections are matched with trajectories, followed by matching low score detections, which often correspond to partly occluded objects.

In addition to the approaches aimed at improving data association, numerous methods [5, 19, 38, 39] attempt to mitigate the effects of camera motion, error accumulation, or poor visual features caused by occlusions. The authors of BoT-SORT model [5] equipped their model with camera motion compensation algorithms by image registration between two adjacent frames. Their online tracking pipeline, illustrated in Figure 3.2, closely resembles other Kalman-filter based online tracking approaches, with the addition of camera motion compensation. On the other hand in [39] the authors view the camera motion on the scene as a source of noise, and model that as a system motion model and inject this noise as process noise in to the Kalman-filter to increase the tracking performance. Although these methods propose elegant solutions to existing challenges, their designs are mostly handcrafted and rely on heuristic choices.

In the studies presented in this section, deep learning methods are typically employed solely for feature extraction from the detected bounding boxes, rather than in the data association step. However, in the era of deep learning, designing methods equipped with appropriate inductive biases for a given problem often yields more effective solutions than heuristic approaches. In the coming sections, we will delve into the approaches which rely on deep learning methods.

3.2.2 Deep Learning-based MOT Approaches

In multi-object tracking, deep learning finds application across various stages of the tracking process. As noted in the preceding section, deep learning models are predominantly employed in the initial stage for detection and extracting high-level features from the detected bounding boxes. However, deep learning methods can also be utilized to determine globally or locally optimal associations, model social interactions, capture target motion, perform similarity calculations, and more. In this section, we will examine studies in the MOT literature by categorizing them into different components as shown in Figure 3.3. It is important to note that the intention is not to classify models, but rather to familiarize the reader with the various components of MOT research that utilize deep learning architectures and discuss their utility and limitations.

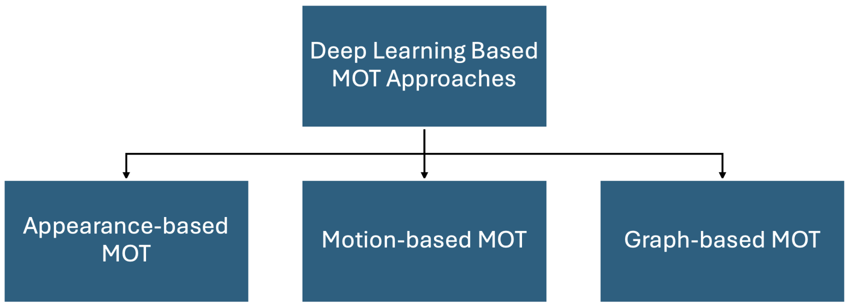


Figure 3.3: Taxonomy of Deep Learning-based MOT approaches.

3.2.2.1 Appearance-based MOT

One of the primary challenges in visual tracking lies in accurately representing the appearance features of objects, enabling their distinction from one another. Unlike single-object tracking, where appearance models are pivotal for discriminating the object of interest from the background, recent methods utilize discriminative correlation filtering models [40] to locate the target object in subsequent frames. However, they differ from multiple object tracking (MOT) in that the semantic class of the tracked object is not known a priori; the target is given at inference in the first frame. Consequently, these models require strong motion models. In contrast, in MOT, the semantic class of the target objects is known a priori, and in each frame, they can be detected by pre-trained object detectors, making object localization relatively straightforward. Therefore, appearance features serve as additional information representing the detections, and tracking can be performed even without them, albeit sub-optimally. Since depending solely on motion models frequently results in tracking failures, especially in crowded scenes. Additionally, in the case of long-term occlusion, objects must be re-tracked with the same ID to prevent tracking loss. This necessitates robust appearance models capable of withstanding changes in illumination or pose, as targets within a scene may undergo such changes during extended periods of absence before reappearing.

ReID Networks: To overcome the previously discussed challenges, pre-trained person re-identification (ReID) networks are utilized to extract appearance features from bounding boxes. The primary reason for using pre-trained ReID models is that objects in the scene are subject to illumination changes, background variations, and pose alterations throughout their trajectories. As a result, features extracted using a simple CNN might yield different embeddings for the same object in different locations within the scene. To accurately identify and recover the tracks of the same target, appearance features are obtained from person re-identification networks trained to produce consistent embeddings for the same objects under different poses or lighting conditions.

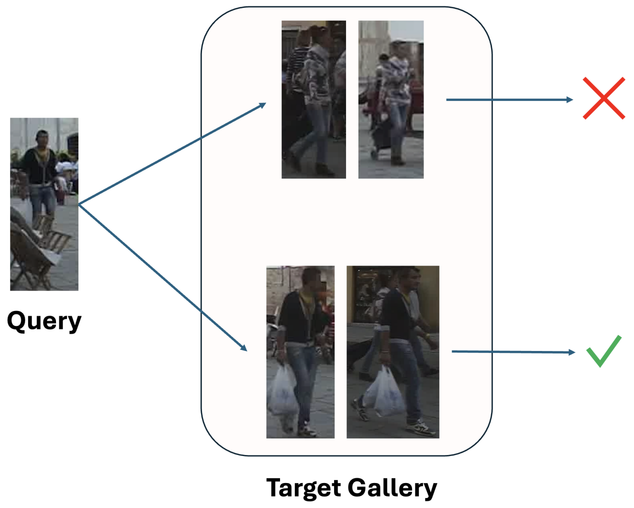


Figure 3.4: Person re-identification example.

Person re-identification involves identifying the same individual across non-overlapping cameras or across different time instances from the same camera as shown in Figure 3.4. This task is primarily utilized for identity retrieval, where the goal is to find instances of a queried person appearing in different frames. The pipeline of person re-identification includes image pre-processing, feature extraction backbone, prediction head and loss function.

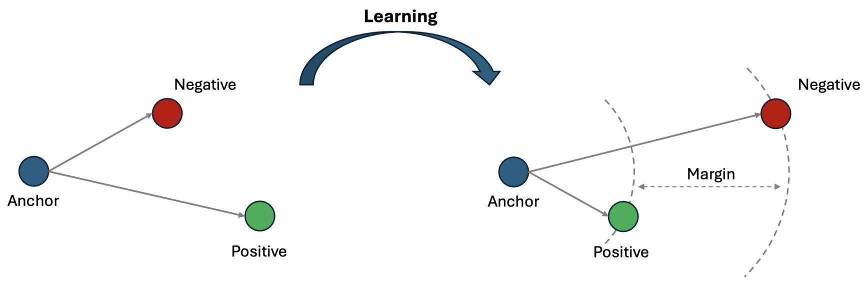


Figure 3.5: Triplet-loss objective.

Although there are different approaches solving this problem, in MOT research most of the models follows the same pipeline with different backbone architectures or loss functions. One of the most popular backbone used in MOT research that utilizes ReID networks [14, 8, 9, 41, 42] is ResNet50 [43] and ResNet-IBN [44]. For the loss functions either cross-entropy loss or triplet-loss in 3.5 is utilized; however, in some studies [45, 46, 47] contrastive loss is also used which also allows models to train in an unsupervised way as in [47].

Person re-identification networks are typically pre-trained on large-scale datasets such as Market-1501 [48] or CUHK03 [49], specifically designed for person re-identification tasks. Some models directly utilize pre-trained ReID networks in multi-object tracking, while others may fine-tune the pre-trained ReID model on MOT-specific datasets to reduce the domain gap between the two tasks. In a recent paper by Seidenschwarz et al. [22], they demonstrated that pre-trained ReID networks often exhibit domain shift when applied to MOT. As a solution, they proposed an on-the-fly domain adaptation scheme to mitigate the domain gap issue.

After obtaining the ReID features of the bounding boxes, they can be considered as visual descriptors for the detected bounding boxes. In downstream tasks, the utilization of these features is design-specific. Some works opt to directly employ metrics such as Euclidean distance or cosine similarity of these features, while others utilize more advanced algorithms, such as deep affinity networks [21, 13] to measure the similarity between the appearances of the targets.

Deep Siamese Networks: Siamese networks are neural network models designed to measure the similarity between two different objects, outputting a probability score of their similarity. Leal-Taixé et al. [13] utilize Siamese networks to measure the visual similarities of two detections, determining whether they belong to the same trajectory. The Siamese network comprises a CNN-based backbone that takes pairs of images as stacked tensors and outputs their similarity score. To obtain the pairwise data association score, a graph is constructed linking all available detections, and tracking is performed by solving the graph optimally with Linear Programming. In [21], the authors utilize an online tracking algorithm where the appearance similarity between tracks and detections is measured using a Siamese network, and optimal associations are determined using the Hungarian algorithm.

ReID networks are a special subclass of Siamese networks. The main distinction between the two lies in how they handle input and output. In ReID networks, appearance information is embedded into a feature vector, whereas in Siamese networks, the model takes pairs of detections as input and directly outputs their similarity scores.

3.2.2.2 Motion-based MOT

As discussed in Section 3.2.1, motion can be modeled using traditional approaches, assuming that the target motion between consecutive frames is linear, especially in high frame rate videos. However, this assumption does not always hold true, particularly in scenarios involving camera motion, the rapid and arbitrary movement of objects in the scene. Modeling motion with the constant velocity assumption can hinder the performance of trackers.

In the literature, there are methods that explicitly or implicitly model motion. Some utilize RNNs [50, 51, 52, 16] to capture the non-linear motion of trajectories, while others implicitly model motion through tracklet embedding [7] or by considering interactions among targets [7, 53] in the scene.

Recurrent Neural Networks: RNNs are neural network models capable of processing sequential data and making predictions at each time step by considering previous states. They utilize a hidden state, which is a vector that encapsulates previous information in the sequence. As RNNs process sequences one step at a time through their gating mechanisms, they continuously update the hidden state based on the in-

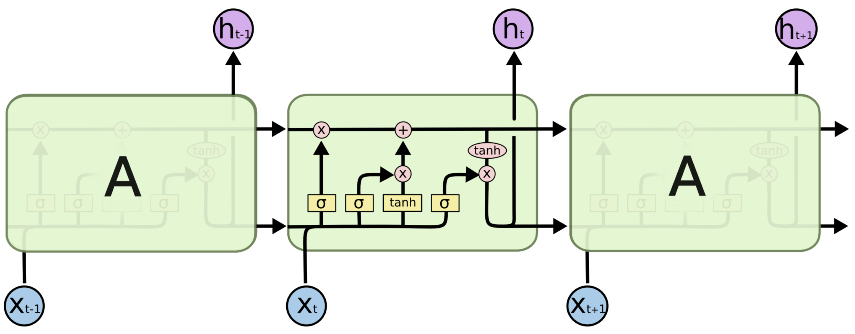


Figure 3.6: LSTM module [6].

put at each time stamp. Upon completing the sequence processing, the hidden state effectively summarizes the information contained within the sequence, representing a higher-level representation of the input sequence. In Figure 3.6 a special kind of RNN, Long Short Term Memory Network (LSTM) is shown, which is a more complex model that better captures the long-term dependencies in the input data.

Milan et al. [50] introduced an online multi-object tracking model employing LSTM networks. Unlike traditional approaches that use Kalman filters to model target motion, they utilize LSTM to learn the nonlinear motions of targets directly from data. The prediction of the state of the tracked object for the next frame depends solely on the current state of the object and the network's hidden state. The model is trained to minimize the mean square error between the predicted state and the ground truth. Apart from motion modeling, the method is also equipped with different RNN blocks to perform data association and track management, making it end-to-end trainable.

Sadeghian et al. [51] also employed the LSTM network to model target motion. However, unlike previous approaches, their model does not utilize the LSTM network as a state estimator. Instead, it takes the velocities of each trajectory independently and produces a fixed-size vector at the output. These features are then used in similarity calculation in the data association step. Additionally, apart from motion modeling, the authors include another LSTM block to model the appearances of the targets, as well as another one to model the interactions among the targets in the scene. However, in recent works, there are more appropriate model choices than LSTM networks for interaction modeling or appearance embeddings, such as GNNs or transformers.

There is another line of research called *Trajectory Forecasting* [16, 52], which aims to forecast the short-term spatial coordinates of objects in a scene based on their previous spatial coordinates. Auto-regressive generative models are commonly used for this purpose. Some MOT methods incorporate advancements in *Trajectory Forecasting* into their frameworks. For instance, Girbau [16] proposed a trajectory estimator based on recurrent mixture density networks, which learns the underlying distribution of an object trajectory. By sampling from this distribution, multiple hypotheses for the object's most likely position can be generated, providing the tracker with a prior on the object's future position. Learning the underlying distribution of object trajectories offers several advantages over state estimators. Unlike state estimators, this approach allows for the sampling of multiple trajectories, which aligns well with the arbitrary and stochastic nature of pedestrian movement. Moreover, in scenarios involving occlusions, the state estimators might accumulate errors due to a lack of observations. In contrast, learning the distribution of trajectories helps mitigate challenges such as occlusion handling, re-identification, and ID switches.

Dendorfer [52] also proposed a method to leverage trajectory forecasting in multi-object tracking. They employ a recurrent neural network-based autoregressive generative model to predict the future coordinates of targets. However, instead of operating in the image coordinate frame, they utilize homography transformation to transform still images into a bird's-eye view representation. This transformation enables better localization of the targets in 3D world coordinates. However, the idea requires heavy image pre-processing. Even though the authors only calculate the transformation parameters from the initial frame, it creates sub-optimal results if there is camera motion or if there are illumination changes in the scene.

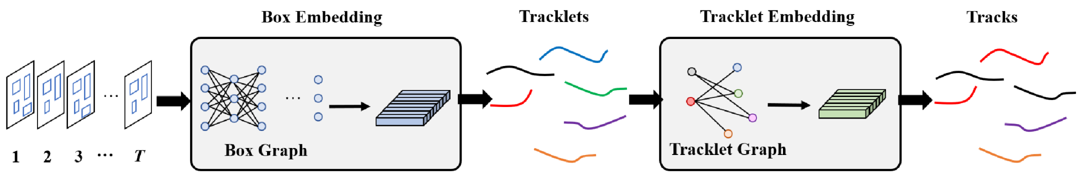


Figure 3.7: The flowchart of the local-global motion tracker. [7]

In addition to the previously mentioned methods that explicitly model the dynamic motions of targets in the scene, there are also approaches that implicitly model target

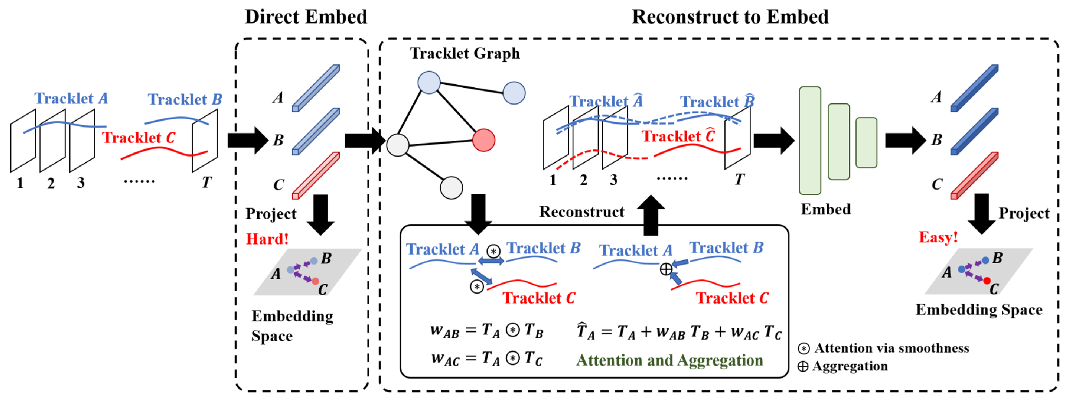


Figure 3.8: Reconstruct-to-embed strategy in the tracklet embedding. [7]

motions by creating embeddings of tracklets, which are temporal sequences of detections or observations corresponding to the same object or target. Wang [7] proposed a local-global motion tracker , employing a two-stage offline tracking approach. The pipeline of the proposed method is given in 3.7. In the first stage, individual detections are associated, forming short tracklets, while in the second stage, tracklets are merged to obtain longer trajectories of the targets. The details of the embedding generation model based on Graph Neural Networks (GNNs) will be further elaborated in Section 3.2.2.3. We mention this study under motion modeling because by creating embeddings of either individual detections or collections of detections (tracklets), the motion dynamics can be learned and summarized in the created embeddings. Then in the inference time, depending on the distance metrics and loss functions, greedy methods can be employed to perform data association between the embeddings of the detections/tracklets.

In Wang et al.'s work [7], the tracklets contain only information about the position and size of the bounding boxes within them. Given that tracklets belonging to the same target exist in different frames and often have varying temporal lengths, it is nontrivial to find a common latent space where their embeddings are close to each other. Therefore, the authors employed a reconstruct-the-embed strategy, as depicted in Figure 3.8, to ensure that tracklets belonging to the same target share similar feature embeddings. The idea of embedding motion of the tracklets into fixed sized feature vectors enables further fusing those features with additional information's such as appearance, or utilizing those embeddings in more complex neural network models.

3.2.2.3 Graph-based MOT

Graph representation learning is a recently emerging field that attracts increasing attention. The background theory and effectiveness of graph representation learning are reviewed in Chapter 2. Along with the theoretical foundation, the most famous GNN architectures are examined in detail. In this subsection, we will begin with methods in the MOT literature that utilize graph-structured data and optimally solve the tracking problem with global optimizations. Then, we will move on to more recent end-to-end trainable networks that include graph neural networks in multi-object tracking.

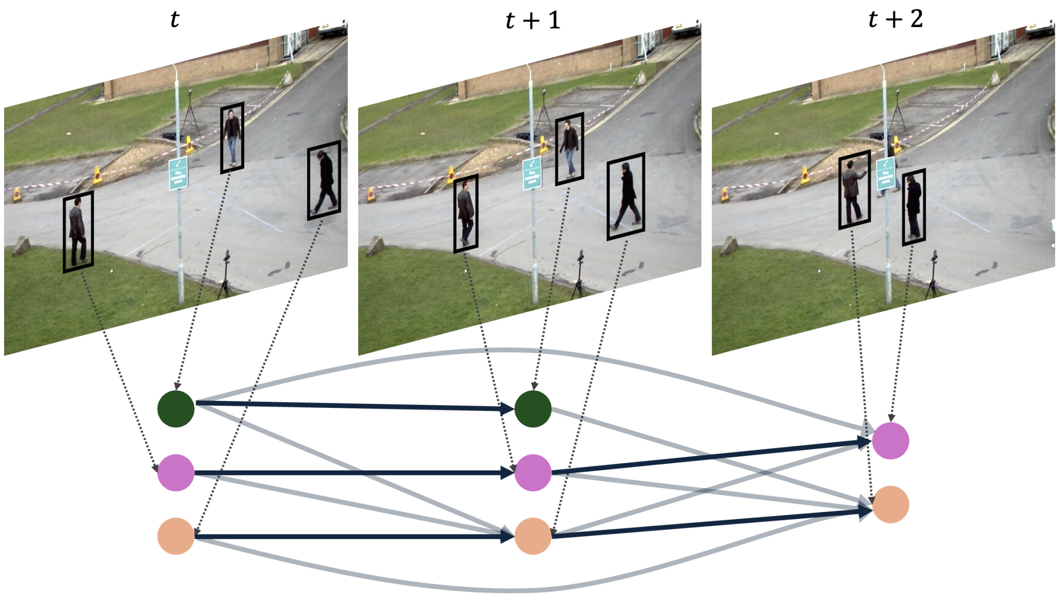


Figure 3.9: Graph representation of tracking sequences; nodes represent detections, edges represent trajectory hypotheses.

Graphs are a commonly used framework for data association in multi-object tracking, where nodes represent detections (or tracklets) and edges represent the possible matching hypotheses, as shown in Figure 3.9. Bold edges represent active connections, indicating that the nodes are connected, while transparent edges represent initially hypothesized matches. The colors represent the unique IDs of the trajectories. Graph structures are predominantly used in offline tracking scenarios, processing batches of frames instead of consecutive ones, allowing them to search for globally optimal solutions to data association. As it can be observed in Figure 3.9, edges are not constrained to formed between consecutive detections; in this way, missing detections due to occlusions can be recovered by finding the correct edges for distant frames.

Numerous studies have utilized various optimization strategies, including multi-cuts [54] and network flow [23, 24, 55], to address the challenge of object association in multi-object tracking. These studies treat this problem as a global combinatorial optimization task using generic graph-based approaches. However, these models lack end-to-end trainability as they rely on solving the optimization problem through linear programming, which is not differentiable. Recognizing this limitation, Brasó and Leal-Taixé (2020) [8] introduced an end-to-end trainable graph neural network model. This model adapts a simplified minimum cost network flow formulation [23]. However, this section will not delve into the specifics of generic graph-based approaches; instead, it will concentrate on reviewing learning-based graph approaches.

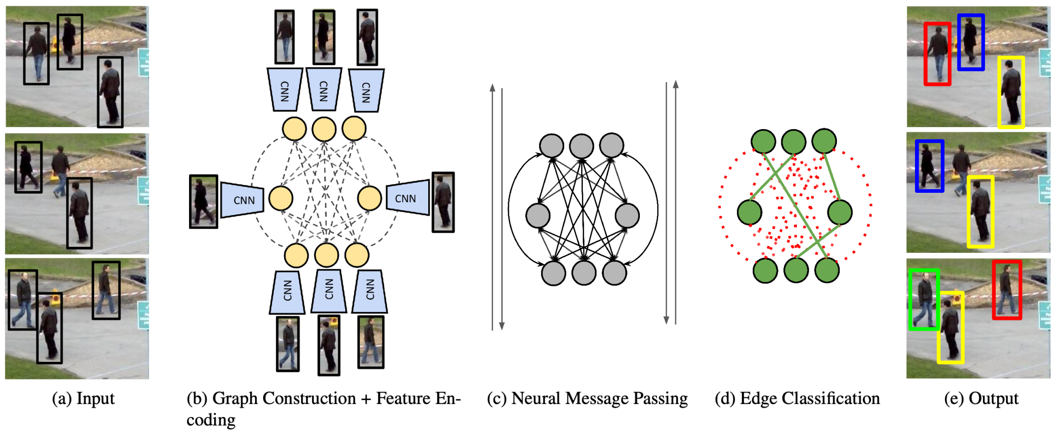

CNN
(a) Input
(b) Graph Construction + Feature Encoding
(c) Neural Message Passing
(d) Edge Classification
(e) Output

Figure 3.10: Overview of MPNTrack from Brasó and Leal-Taixé [8].

The simplified minimum cost network flow formulation proposed by Brasó and Leal-Taixé (2020) [8] involves a straightforward edge classification task on the graph. The pipeline of the approach is illustrated in Figure 3.10. Steps (a) and (b) are common in generic graph-based approaches. However, in step (c), they conduct a series of message-passing operations on the graph to enhance representations on the nodes (detections). Subsequently, in step (d), a simple two-layer MLP is applied to perform binary classification, classifying edges as active (1) or inactive (0) to indicate that the connected detections belong to the same object, hence forming the trajectory. After post-processing, the final results are obtained. This formulation offers the utilization of the advancements in the GNN literature, since the framework is end-to-end trainable. However, the method has shortcomings; as the number of objects increases in the scene, the model size dramatically increases because the formed edges between detections increase exponentially with the number of nodes. Therefore, sparser graphs must be formed in the second step (b). Even with sparse graphs, the model does not fit into a single GPU, hence the temporal axis is processed with a sliding window fashion with overlap between those windows. Post-processing is then required to extract the trajectories

Apart from performing edge classification to find the trajectories Dai [41] proposed iterative graph clustering which is similar to two-stage object detector Faster RCNN, i.e., proposal generation, proposal scoring and proposal pruning. In their model, they employ a GCN model to calculate the purity scores of each cluster. This GCN model takes visual and spatio-temporal cues as input and produces the purity probability, indicating whether there are any distinct identities within the detections inside the proposed cluster. This formulation reduces computation costs, as they iteratively cluster nodes, with the number of vertices in the graph decreasing at each iterations.

In addition to the previously mentioned models that utilize GNN models in offline tracking scenarios, there are also models in the literature designed for online tracking on a frame-by-frame basis [56, 57, 53]. Papakis introduced GCNMatch [57], operating on bipartite graphs, aiming to incorporate both spatial and visual cues from tracks in past frames and the current frame. This incorporation is achieved through message passing on a bipartite graph where current detections and previously created tracks interact, making them aware of each other's visual and geometric cues. Optimal association is performed with Hungarian algorithm. Due to its online nature, GCNMatch cannot effectively model the long-term interactions of objects in scenarios involving severe occlusions without increasing memory usage.

In both [56, 53], GNN models are employed to capture interactions between objects within the scene. This allows the models to enhance awareness of surrounding objects and create more discriminative features by effectively modeling these interactions.

Rangesh et al. proposed TrackMPNN [58], which is a framework based on dynamic undirected graphs representing the data association problem over multiple timesteps.

They utilize a message passing graph neural network (MPNN) that operates on these graphs to generate the desired likelihood for each association. Their work provides insights into designing an MPNN model tailored for multi-object tracking research.

3.3 Revisiting the Baseline Methods

In this section, we revisit the baseline methods that form the foundation of our research. Our work builds upon the insights and methodologies presented in prior studies, namely SUSHI [9] from Cetintas and GREASELM [11] [11] from Zhang. By revisiting these approaches, we aim to provide a comprehensive understanding of the underlying principles and techniques that have shaped our research direction. This review will serve as a crucial framework for clarifying the advancements and innovations introduced in our work

3.3.1 SUSHI

The SUSHI model, proposed by Cetintas et al. [9], serves as a graph-based hierarchical multi-object tracker. In contrast to previous MOT techniques, which often treat short-term and long-term associations differently, this method employs a GNN architecture to address both types of associations in a unified way. It achieves data association across various time frames through a hierarchy of smaller graphs. At the lowest hierarchy level, nodes represent object detections in nearby frames. Leveraging the GNN, these nodes are processed into short tracklets, and subsequent graphs are formed to extend trajectories at each level of the hierarchy. This unified approach is facilitated by the consistent use of the same GNN architecture with identical parameters at each hierarchical level, eliminating the need to assume optimal cues for individual time frames.

The approach follows the tracking-by-detection paradigm and is an offline tracking method. In the first step, each object in every frame is detected, and then their ReID features are extracted [59]. The video clip is then recursively partitioned into smaller sub-clips, forming a binary-tree-like structure, as illustrated in Figure 3.11. In each sub-clip, graph-structured data is formed by representing detections/tracklets in the

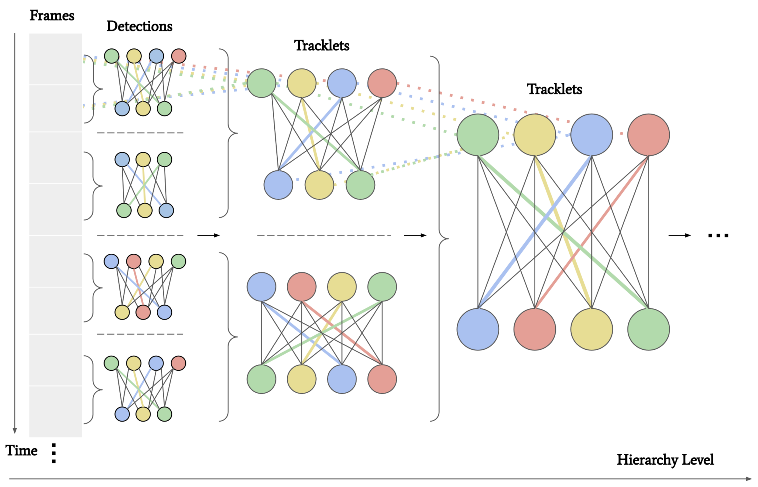


Figure 3.11: Recursive partitioning of a video clip into hierarchy of graphs [9].

nodes and creating edges between nodes that are in non-overlapping frames, following certain distance heuristics, such as Euclidean distance on the image plane or l_2 norm distances of the extracted ReID features, to obtain a sparser graph. Subsequently, starting from the lowest level of the hierarchy, each graph is solved independently, meaning that data association in each sub-clip is performed. Associated nodes are then merged, and a new graph is created at the next higher level of the hierarchy. This process continues until the highest hierarchy level spans the entire clip duration, at which point the trajectories formed in this last hierarchy are returned.

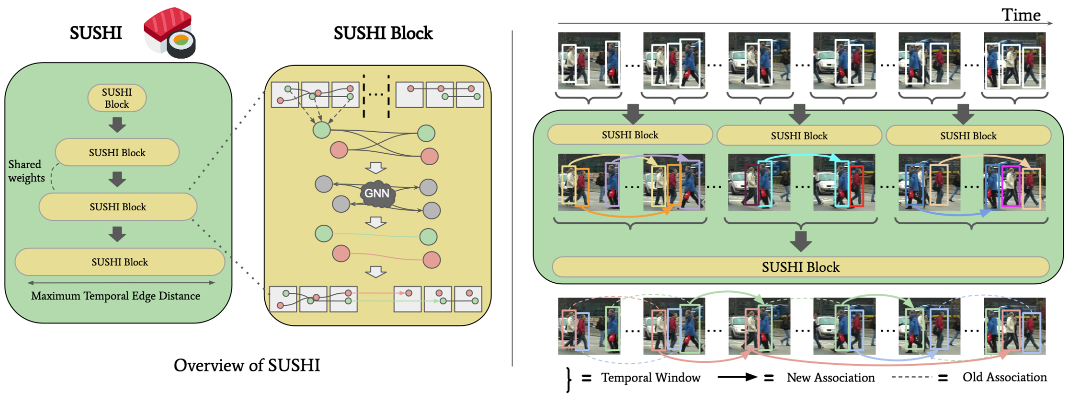


Figure 3.12: SUSHI block overview. Each SUSHI block processes a graph composed of tracklets from a sub-clip, utilizing neural message passing to merge nodes into longer tracks. [9]

Each SUSHI block, shown in Figure 3.12, processes a graph representing a sub-clip, where nodes are equipped with ReID features and edges carry geometric cues such as relative distance and size. These features are transformed into more representative features through a series of neural message passing operations using a shared GNN model. The weights of this GNN model are shared across all sub-clips and hierarchy levels. After obtaining more representative features, edges are classified as active or inactive to indicate which nodes are to be merged into the subsequent hierarchy. For the merged nodes, ReID features are averaged, and positional features are represented only by the first and last detection nodes in the time frame, which discards the intermediate positional information of the tracklets. This approach is not ideal for modeling the motion of the tracklets, as the movement of objects in the scene is more complex and cannot be inferred from only the start and end positions. For implementation details please refer to the original work [9].

3.3.2 GreaseLM

With the advancements in deep learning, researchers often draw inspiration from various application fields for their studies. This interdisciplinary approach is valuable because many fundamental concepts and algorithms across different fields share underlying principles or can be adapted to suit different needs. Therefore, we are investigating the potential application of the GreaseLM [11] method, despite it not being directly related to multi-object tracking.

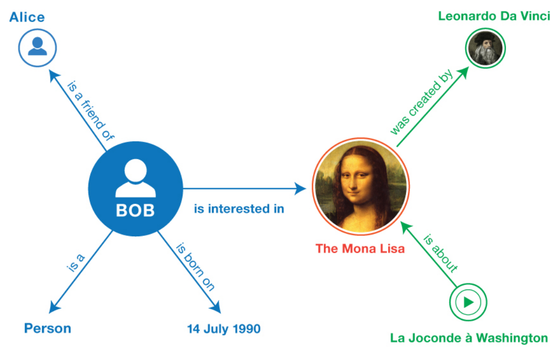


Figure 3.13: An example of Knowledge Graph [10].

GreaseLM is proposed to enhance the reasoning capabilities of pre-trained Large Language Models (LLMs) by incorporating structured knowledge graphs (KG) alongside

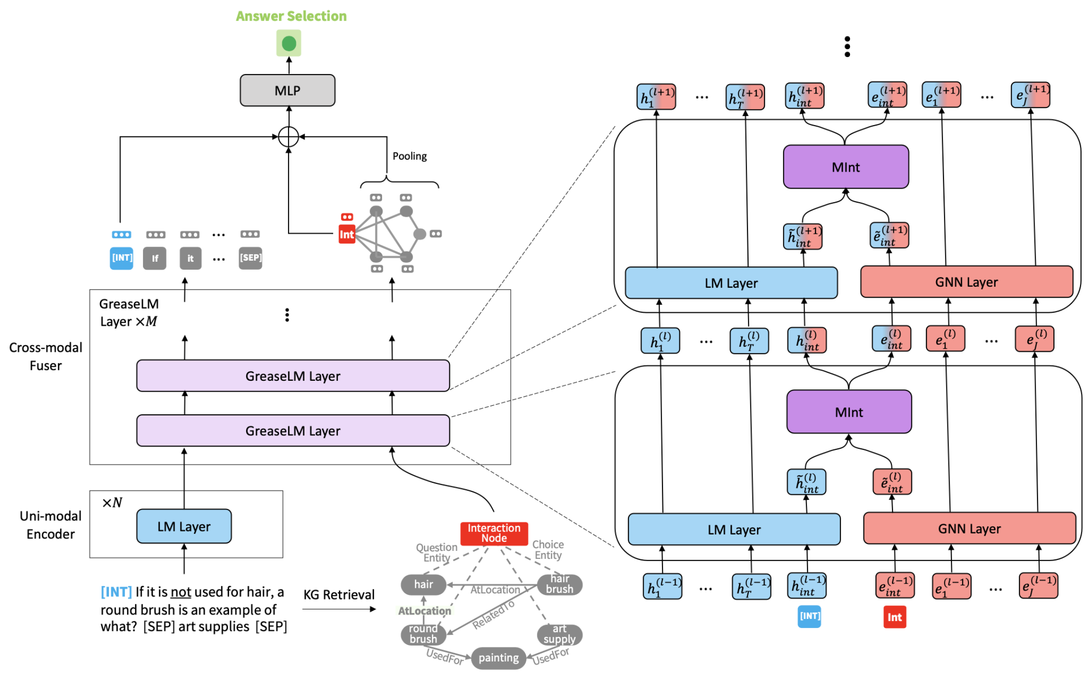


Figure 3.14: GreaseLM architecture [11].

textual context. The knowledge graph represents a network of real-world entities, such as objects, events, situations, or concepts and illustrates the relationships between them, an example is shown in Figure 3.13. The model takes two inputs: textual content, similar to any other LLMs, and a knowledge graph. The textual content is processed using Transformer architecture, while the knowledge graph is processed with a GNN. After each layer of Transformer and GNN, a modality interaction module bi-directionally transfers information from each modality to the other. Consequently, all tokens in the language context receive information from the KG entities, and the KG entities receive information from the text tokens.

The bi-directional information propagation occurs between the interaction token $\hat{h}^l_{int}$ and an interaction node $\hat{e}^l_{int}$. The interaction token is a learnable embedding token that is pre-pended to the input sequence, whereas the interaction node is an artificial node (learnable embedding) placed in the knowledge graph and only connected to a subset of the nodes in the knowledge graph. These interaction tokens and nodes first interact with text and the knowledge graph respectively, then are fed into the interaction module, which performs the following:

$$[h^l_{int}; e^l_{int}] = MLP([\hat{h}^l_{int}; \hat{e}^l_{int}]) \tag{3.1}$$

In Equation 3.1, $[\cdot;\cdot]$ represents concatenation, and MLP denotes a basic two-layer MLP. The interaction token and node are then extracted and fed into the subsequent layers, as illustrated in Figure 3.14.

For further details on the model, please refer to [11]. In this thesis, our work is inspired by the bi-directional interaction between two different modalities. Section 4 will discuss how this proposed architecture could be leveraged in MOT.

3.4 Discussion

As discussed in previous sections of this chapter, learning-based methods for data association exhibit superior tracking performance compared to heuristic approaches. Learning optimal cues for data association from data proves more powerful than relying on hand-crafted heuristics. Additionally, offline models outperform their online counterparts. Offline models have access to both previous and future frames, allowing for optimal batch-wise association by globally minimizing association costs. In contrast, online methods operate on a frame-by-frame basis and are prone to losing track identity in ambiguous scenarios, particularly when faced with severe occlusion or missed detections.

In offline tracking, graph-based approaches show promise due to the ease with which tracking data can be represented using graph structures in tracking-by-detection paradigm. This parardigm allows for the utilization of recent powerful methods reviewed in Section 2. For this reason, we have chosen to adopt an offline tracking approach that utilizes graph-structured data. In the next section, we will propose a bi-directional motion encoder in a hierarchical setting for multi-object tracking.

CHAPTER 4

PROPOSED BI-DIRECTIONAL MOTION ENCODER FOR MOT

4.1 Introduction

Multi-Object Tracking (MOT) involves the simultaneous detection and tracking of each object within a scene, assigning a unique ID to each object throughout the entire video duration. MOT has extensive applications in diverse fields, such as autonomous driving, robotics and video analytics. In recent years, MOT is widely studied and significant progress has been made; however, MOT still remains as a challenging problem due to severe occlusions, similar appearances and interactions and complex motions between objects.

By the advent of powerful object detectors [17, 33], the tracking-by-detection paradigm has become a common approach. To establish associations between detections in different frames, spatial and temporal priors, such as motion and appearance are utilized. In recent studies, the association is either performed online with motion models and local appearance cues [18, 19, 14, 5, 20, 13, 21], or offline, mostly utilizing graph-structured data with global matching costs [9, 8, 41]. In both cases, object motion is modeled with either Kalman filters or simple linear-motion assumptions. These methods often fall short, when there are interactions among targets. Even though graph-based methods implicitly model interactions, they often lack a strong motion model.

In this chapter, we propose a novel solution for tracklet merging using some recent promising learning based architectures. We propose a novel method to explicitly model both the motions and interactions among targets. Our approach involves a simple model that does not require any handcrafted short-term or long-term matching processes. We achieve this goal by leveraging a joint Transformer and GNN encoder. In the former, the individual detection of each tracklet can attend to each other via self-attention; thus, modeling motion of individual targets. While the latter captures the interactions among them via Message Passing. A fundamental challenge remains in combining GNN with Transformer while leveraging the advantages of both architectures. Although there are numerous works on how to combine GNN and Transformer architectures mostly in NLP domain such as [60, 61, 62], they are mostly utilizing the pre-trained transformer architecture (BERT) to produce the embeddings for the nodes of the subsequent GNN layer, and they do not allow for bi-directional information propagation between these two modalities. Recently, a new approach, namely GreaseLM [11], has been proposed to efficiently combine all the information input to both Transformer and GNN. This solution enables bi-directional information propagation between these two modalities. Moreover, adopting this approach in our study allows motion modeling to depend on interactions, and conversely, interaction modeling to depend on the motions of each target.

4.2 Motivation

We propose a novel architecture, namely *Tr-GNN*, which models object motion while still being aware of social interactions. *Tr-GNN* utilizes a joint Transformer and GNN encoder in a hierarchical setting (Figure 3.11) and follows the tracking-by-detection paradigm; hence, moving object bounding boxes are assumed to be available by any pre-trained object detector [17]. The ultimate goal is to find optimal associations across detections (or tracklets in higher hierarchies).

A tracklet refers to a grouping of N detections $\{d_1^i, d_2^i, ..., d_N^i\}$ associated with the same object (target) i. Each detection included in a tracklet belongs exclusively to that tracklet, ensuring that there are no overlapping detections between different tracklets representing distinct targets. These detections contain information about the detected object, which includes the bounding-box position and size normalized relative to the image dimensions. Additionally, visual features, ReID, are extracted from the corresponding portion of the image frame and presented alongside with the positional information of the each detections.

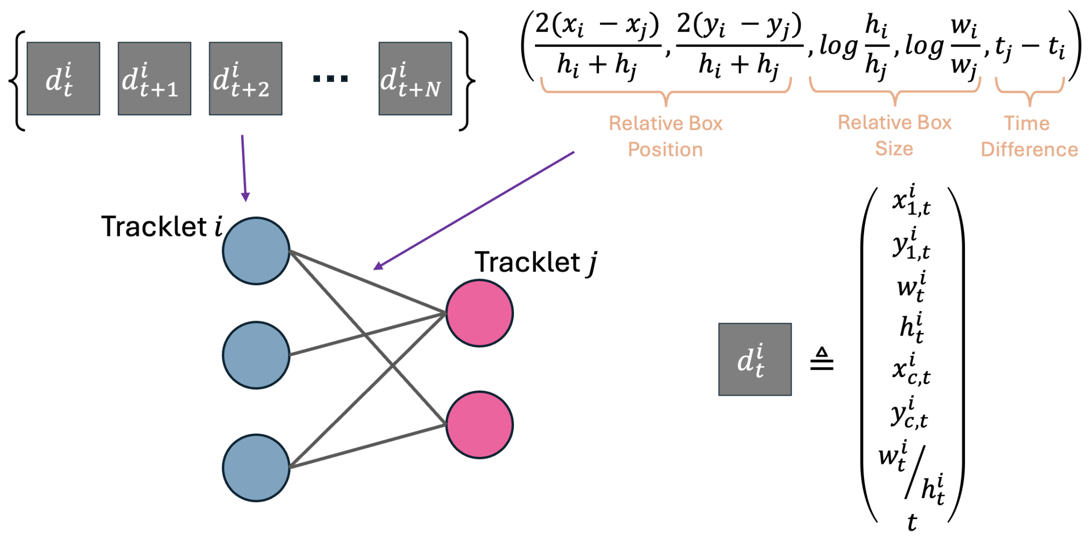
We drew inspiration from the hierarchical solution approach of SUSHI method [9]. SUSHI is an offline MOT model that proposes a hierarchical solution to multiple object tracking by partitioning the input video with a binary-tree-like hierarchical structure, where each leaf corresponds to individual frames and the root nodes correspond to tracklets covering the entire temporal dimension, as shown in Figure 3.11. Tracking is performed by recursively merging detections (tracklets in the upper hierarchies) in the leaf nodes via GNNs, thereby forming longer tracklets in the subsequent hierarchies until the tracklets in the root nodes cover the entire temporal dimension of the input video. However, in SUSHI model the tracklets formed in the upper hierarchies only consider the start and end detections of the object, (d_1^i, d_N^i), instead of N detections, which might discard valuable information that could be utilized to model the motion characteristics of each tracklet, especially when the object has a complex motion. Our aim is to utilize all detections of the individual tracklets to better model their motion characteristics with a joint Transformer and GNN encoder.

Another inspiration for our proposed method comes from another line of research in Natural Language Processing. GreaseLM [11] model introduces bi-directional information propagation between two different modalities: GNN and Transformer to improve performance. Details of this work are provided in Section 3.3.2.

Therefore, in our work, we aim to utilize this bi-directional information propagation in such a way that motion modeling depends on interactions, and conversely, interaction modeling depends on the motions of each target.

4.3 Proposed Bi-Directional Motion Encoder

Ours proposed model follows the commonly used graph formulation in MOT [8]. Although the graph structures model more irregular relations, in this particular problem consecutive time instances or tracklets that follow each other, provides a more regular interconnection. Hence, for such regular relations, *bipartite graphs* can be used to determine different connection hypotheses at different time instances. In each partition of each hierarchy, an undirected *bipartite* graph $G(\mathcal{V}, \mathcal{E})$ is formed. The edges $\mathcal{E} \subset \{(v_i, v_j) \in \mathcal{V} \times \mathcal{V} | t_i \neq t_j\}$ are formed between nodes which have a possibil-



$\left\{ d_t^i \; d_{t+1}^i \; d_{t+2}^i \cdots d_{t+N}^i \right\}$
$\left(\frac{2(x_i - x_j)}{h_i + h_j}, \frac{2(y_i - y_j)}{h_i + h_j}, \log\frac{h_i}{h_j}, \log\frac{w_i}{w_j}, t_j - t_i \right)$
Relative Box Position
Relative Box Size
Time Difference
Tracklet i
Tracklet j
$d_t^i \triangleq \left(x_{1,t}^i, y_{1,t}^i, w_t^i, h_t^i, x_{c,t}^i, y_{c,t}^i, w_t^i / h_t^i, t \right)$

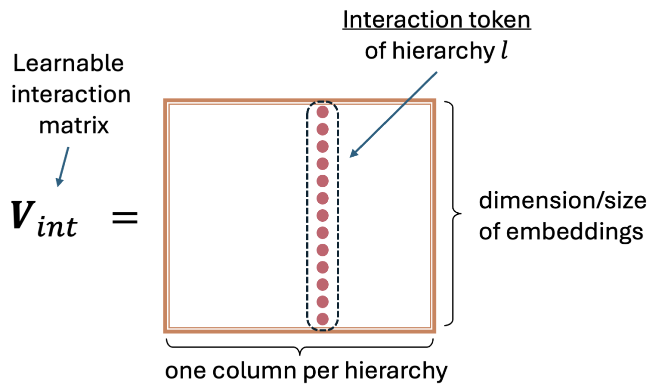
Figure 4.1: Feature encoding of input graph.

ity of matching, determined by simple distance heuristics, such as IoU or GIoU, as proposed in [9]. Subsequently, our proposed model independently identifies the optimal associations within each bipartite graph, by performing binary classifications on the edges, in which the predicted probability indicates the likelihood that the nodes connected by that edge belong to the same object.

Our key distinctions from SUSHI model [9] include the use of a bipartite graph formulation and the incorporation of a joint Transformer and GNN encoder for optimal association. We utilize the bipartite graph structure to limit the number of detections in each tracklet at different hierarchy levels. Specifically, in the first hierarchy, nodes have a single object detection (i.e. bounding box); in the second, they have two detections, and this pattern continues with powers of 2.

Each resulting bipartite graph is illustrated in Figure 4.1, which serves as the input to the joint Transformer and GNN encoder. The provided bipartite graph example in Figure 4.1 demonstrates the features utilized by our model. We only utilize the positional information of the detections, such as bounding box coordinates (top-left corner, width, and height), center location, aspect ratio, and the frame number. These features are stored in the nodes of the graphs, representing individual tracklets.

In addition to the formed bipartite graph, an interaction token is also included as input to our model. This token is appended to all tracklet sequences and is aimed to model the motion characteristics of the tracklets, which they belong, by relating all individual detections. As depicted in Figure 4.2, the interaction token is a learnable embedding that remains the same within the same hierarchy level for all tracklets but differs for different hierarchy levels.



Interaction token of hierarchy l
Learnable interaction matrix
V_{int} =
dimension/size of embeddings
one column per hierarchy

Figure 4.2: Interaction token representation

4.3.1 Joint Transformer and GNN Encoder

One can conceptualize each detection in a tracklet as a token in a text. Instead of utilizing a KG as in GreaseLM [11], our approach leverages a dynamically formed bipartite graph at each hierarchy level. This structure enables different tracklets to exchange information, allowing them to take into account the social interactions among them. In the Transformer encoder illustrated in Figure 4.3, we prepend the same interaction token - identical only within the same hierarchy level - to all tracklets. Through masked self-attention, each interaction token learns the motion pattern of its corresponding tracklet by attending to all its positions. Consequently, each interaction token within each tracklet learns the motion embedding of the tracklet it belongs to. These interaction tokens are then extracted and stored in the nodes of the bipartite graph. Message passing is performed, enabling each interaction token belonging to distinct tracklets to exchange information.

More specifically, for each graph $G^l = (\mathcal{V}^l, \mathcal{E}^l)$ at hierarchy level l, each node represents a tracklet, which is a set of detections $T_i^l = \{v_{int}^l, d_{i1}, d_{i2}, ..., d_{in}\}$ (as shown in Figure 4.1) merged from previous hierarchies. Here, $d_k \in \mathbb{R}^{d_v}$ denotes an embedding for each detection and v_{int}^l denotes the prepended interaction token (as shown in Fig-

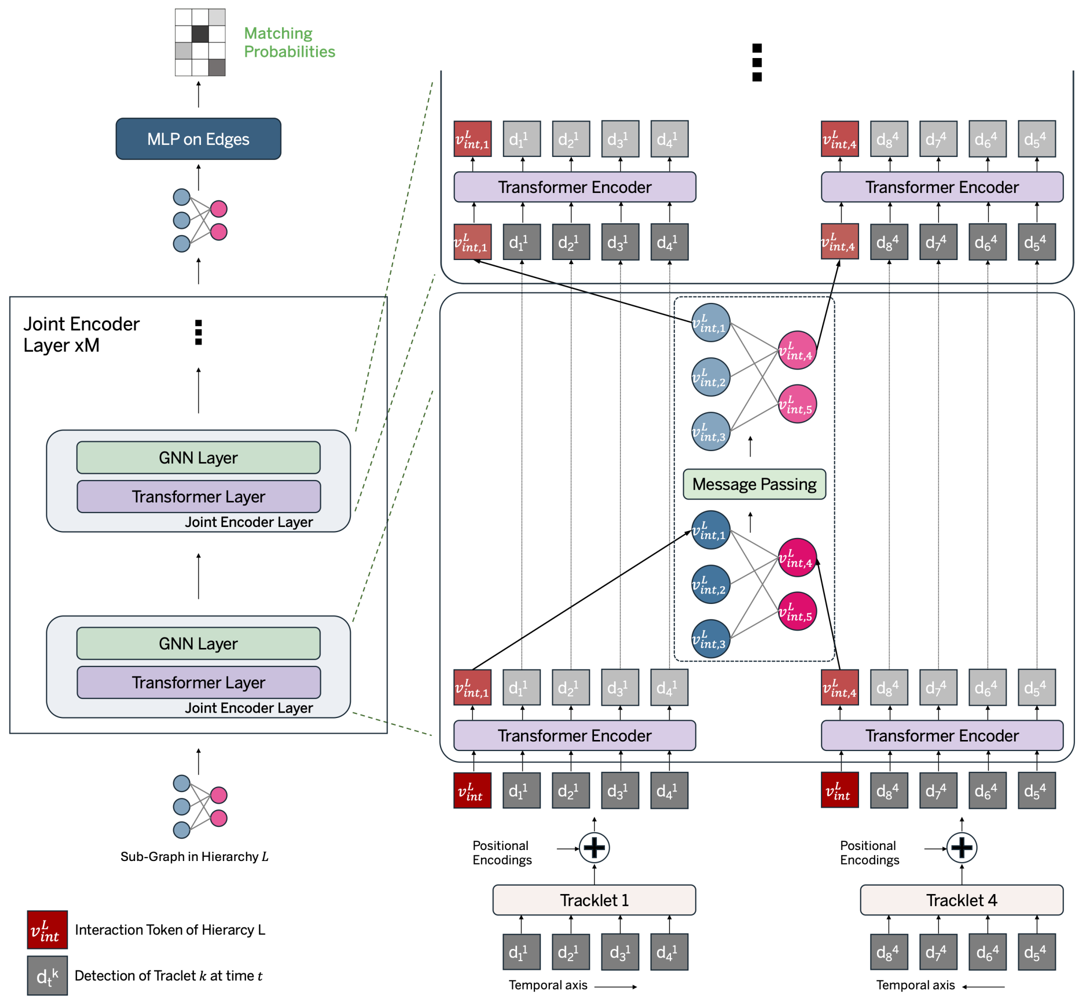


Figure 4.3: Tr-GNN architecture overview. Joint Transformer-GNN encoder processes a bipartite graph, and all the weights of Transformer and GNN layer are shared across all hierarchy and sub-graphs.

ure 4.2) to the tracklet sequence which is same for every tracklet in the same hierarchy level l initially; moreover, we consider embeddings $e_{i,j}$ for each edge $(i, j) \in \mathcal{E}^l$. Detection d_k contains positional information, such as bounding-box position and size, and the frame number to which the detection belongs, as shown in Figure 4.1. In the edges formed between two tracklets, relative position and size information are utilized based on the closest detections of the both following [8]. The formed tracklets, which are sets of detections, are input into a Transformer encoder, as illustrated in Figure 4.3. Masked self-attention is performed, and embeddings for each token are propagated and interaction tokens attend to all detections of the tracklets that they are belong to. Then, to enhance model efficiency, only the interaction tokens of each tracklet are extracted and stored as the node features of the graph $G^l = (\mathcal{V}^l, \mathcal{E}^l)$, denoted as v_i. Message passing occurs solely among these interaction tokens, thereby learning interactions among tracklets within the bipartite graph. The propagated interaction tokens are then prepended to the tracklet sequences and once again input into the Transformer encoder. This procedure is repeated M times, allowing each interaction token to interact with both individual detections within each tracklet and other interaction tokens from different tracklets. This design creates a bottleneck for information propagation, ensuring a focused mechanism for capturing relations between tracklets. An overview of the proposed method is presented in Figure 4.3.

Reversed Tracklets: Since our goal is to match tracklets with similar motion patterns, obtaining similar embeddings for tracklets belonging to the same target is not straightforward. This is due to the tracklets' nature of existing in different temporal positions and the varying size of their bounding boxes in the image plane as they move closer to or farther away from the camera. In our bipartite formulation, to simplify the model's task, we flip the positions of the tracklets on the right side of the bipartite graph. This intuitively allows the model to perceive those tracklets as moving towards each other, rather than moving further apart.

Masked Self-Attention: In the tracking-by-detection paradigm, due to detection errors or severe occlusions, some detections may be missing, resulting in empty positions within formed tracklets. It is important to note that the length of each tracklet remains consistent within the same hierarchy level. Starting from a length one tracklet, subsequent hierarchies see an increase in tracklet length in powers of two. Consequently, certain positions within formed tracklets may be empty, which we fill with zeros to maintain consistent sequence length. However, including these empty positions in self-attention can negatively impact motion modeling. To address this, we employ a masked self-attention strategy to prevent attending towards the empty tokens.

Weight Sharing: In our architecture, all weights of the Transformer and GNN encoder are shared across all hierarchy levels to reduce the number of learnable parameters of the model. At the lower levels of the hierarchy, detections that are closer in time are merged for short-term tracking, while at higher levels, tracklets spanning longer temporal ranges are merged for long-term tracking. To enable the model to distinguish between short and long-term tracking scenarios, learnable interaction tokens are different for each hierarchy l, as shown in Figure 4.2. This differentiation injects the information about the hierarchy level at which the model operates.

4.3.2 GNN Model

In a GNN model, a message passing framework is employed, wherein the features of nodes, denoted as v, and edges, denoted as e, are updated in each message passing phase based on the graph connectivity. The formulation for these updates can be expressed as follows

$$\begin{aligned} \hat{v}_i &= f_n\Bigg(\big[\sum_{j \in N_i} \alpha_{ij} e_{ij} || v_i \big] \Bigg) \\ \hat{e}_{ij} &= g_n\Bigg(\big[e_{ij} || v_j - v_i \big] \Bigg) \end{aligned} \tag{4.1}$$

where g_n and f_n update functions parametrized by one-layer MLPs, N_i is the neighbors of node i. This GNN formulation is a variant of Graph Attention Networks [3]. The attention coefficients are calculated as follows.

$$\alpha_{ij} = \frac{exp\Bigg(\mathbf{a}^T \Bigg(\sigma(\mathbf{W}v_i - \mathbf{W}v_j) \Bigg) \Bigg)}{\sum_{k \in \mathcal{N}(i)} exp\Bigg(\mathbf{a}^T \Bigg(\sigma(\mathbf{W}v_i - \mathbf{W}v_k) \Bigg) \Bigg)} \tag{4.2}$$

The rationale behind the utilization of the differences between pair of node features in both attention calculation and message calculation, presented in Equations 4.1 and 4.2, is based on findings from previous works, and theoretical foundation is indicated in Appendix B.1. In [63], it is demonstrated that employing edge update functions, as described in Equation 4.1, provides the model with a partial translation-invariance property. This property serves as an effective geometric prior, enabling the model to consider local patch geometry while retaining global shape information. Additionally,

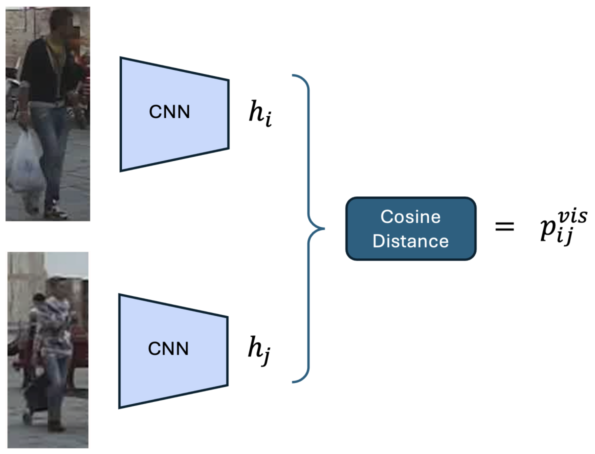


Figure 4.4: Appearance-based similarity calculation.

in the MOT literature, Rangesh et al. [58] showed that incorporating the difference of node features in the message calculation, as illustrated in Equation 4.1, leads to a substantial increase in tracker performance in their design. Hence, we opted with a similar message computation formulation.

4.3.3 Tracklet Association

After propagating information between each position and different tracklets in the scene, the embeddings of each tracklet are formed in the interaction tokens. These embeddings effectively capture high-level features, facilitating both motion information and awareness of the social interactions. To compute the positional similarities of distinct tracklets along each edge of the bipartite graph, the nodes connected by that edge subtracted and fed into a Multi-Layer Perceptron (MLP) with a sigmoid activation.

$$p_{ij}^{pos} = MLP([e_{ij}||v_i - v_j])$$

This process produces the likelihood of connected tracklets belonging to the same target. However, when severe occlusion occurs in the scene, relying solely on motion information may not be sufficient; hence, visual cues must also be utilized. Currently, our model does not incorporate visual cues directly. Instead, we obtain them from a pre-trained Re-Identification (Re-ID) network, similar to the approach described in [14]. Without further training, we calculate the cosine distance between the visual embeddings as shown in Figure 4.4, $h_i \in \mathbb{R}^{d_h}$, of pairs of tracklets,

$$p_{ij}^{vis} = \frac{< h_i \cdot h_j >}{||h_i|| \, ||h_j||}$$

To account for both motion and visual cues, we compute the weighted sum of these two similarities, as below:

$$p_{ij} = \lambda \cdot p_{ij}^{vis} + (1 - \lambda) \cdot p_{ij}^{pos} \tag{4.3}$$

After calculating the similarity measures between every pair of nodes in a bipartite graph, a greedy matching strategy is utilized, prioritizing the highest probabilities of matches. The entire model can be trained using the Binary Cross Entropy loss, which aims to maximize the similarity between pairs of nodes belonging to the same target

4.4 Experiments

Dataset: The proposed method is tested on MOT17 [64] public benchmark. MOT17 contains 7 training sequences and 7 test sequences. The videos were captured by stationary cameras mounted in high-density scenes with heavy occlusion. Only pedestrians are annotated and evaluated. The video frame rate is 25-30 FPS. The MOT dataset does not provide an official validation split. For ablation experiments, we reserve 5 training sequence for training and 2 of them for validation. Our main results are reported on the test set.

It is important to understand the performance metrics of the MOT research, as it is not straight forward to measure the quality of a tracker with a single score. Our results are given in MOTA, IDF1 and HOTA metrics, and they are defined in the Appendix-C.

Implementation Details: Our implementation is based on SUSHI [9] model. For our ReID feature extractor we use a pretrained ResNet50-IBN following [41], and this ReID model is not further trained. Following [9], all hierarchy levels are trained jointly with a learning rate $3 \cdot 10^{-4}$, ADAM optimizer [65] is used.

We construct hierarchies spanning a maximum temporal edge distance of 128 frames. Our hierarchy consists of seven levels, each processing sub-clips of 2, 4, 8, 16, 32, 64 and 128 frames, respectively. For each graph, following [9] we connect each node to its top 10 nearest neighbors based on geometry, appearance, and motion similarity. We process entire videos by feeding overlapping clips of 128 frames to our method in a sliding window fashion. We then merge per-clip tracks into trajectories of arbitrary length with a simple stitching scheme, similarly to [8]. During inference, we fill trajectory gaps by linear interpolation similar to [9].

4.4.1 Ablation Studies

We first ablate the two main aspects of our design: one is the bidirectional information propagation and the heuristic choices that we follow in our design. The second aspect is the effect of the number of joint Transformer GNN encoder layers. In the first ablation study, we compare our full model with five baselines.

Appearance Only performs matching in a hierarchical manner by utilizing only the cosine distance between pre-trained Re-ID visual features, without any training or fine-tuning.
Without Transformer does not utilize bi-directional tracklet embedding; it exclusively employs the GNN model for making optimal associations. While it shares similarities with the SUSHI model, no fine-tuning of hyper-parameters is conducted. Moreover, our GNN model differs from the SUSHI model.
Without Masked-attention does not employ any masking on self-attention calculation in the Transformer encoder. At certain timestamps, due to missing detections or occlusions, some positions might be absent. We zero-fill those positions, however, we do not prevent the model to attend those zero-filled positions in self-attention layer in Tr-GNN.
Without Reversed-tracklet, in the bipartite graph, tracklets located on the right side are not reversed along their temporal axis.
Without Appearance, does not utilize appearance based similarity scores in tracklet association. The parameter λ in Equation 4.3 is set to zero. So the associations are performed with only positional ques processed by our model Tr-GNN.

Table 4.1: Ablation of the single parts of our method in MOT17 validation set.

	HOTA↑	IDF1↑	MOTA↑
Appearance Only	67.63	68.84	87.27
w/o Transformer	86.3	92.4	96.7
w/o Masked-attention	87.93	94.93	97.51
w/o Reversed-tracklet	89.07	96.72	97.68
w/o Appearance	89.32	96.85	**98.08**
Tr-GNN (ours)	**89.61**	**97.25**	98.00

Table 4.1 presents the results of our ablation experiments. Based on the results tabulated in this table, the proposed full Tr-GNN model significantly outperforms over the baselines across all metrics. The importance of bidirectional tracklet embedding in data association is clearly highlighted by the results, showcasing an approximate 3% enhancement in both HOTA and IDF1 scores. Additionally, attending to empty position tokens hinders the model's performance by potentially introducing noisy embeddings. Furthermore, although the impact of reversed tracklets may be subtle, it still contributes to improved results without any additional computational overhead. Our model effectively captures target motion characteristics, performing just 0.3% worse when relying solely on positional features, showcasing its robustness.

Table 4.2: Ablation of the number of Transformer-GNN layers

	Number of Layers					
	1	2	3	4	5	6
HOTA↑	88.68	89.11	**89.61**	89.35	89.36	89.37
IDF1↑	96.01	96.64	**97.25**	97.11	97.08	97.10
MOTA↑	97.78	97.87	98.00	97.96	97.84	**98.02**

In the second ablation study, we analyze the effect of the number of joint Transformer-GNN encoder layers. By iteratively applying the joint encoder, our model facilitates more bi-directional information propagation between two modalities. Increasing the bi-directional information propagation improves results up to a certain point and the best results are achieved when the number of layers is equal to three as shown in Table 4.2. However, increasing number of layers also drastically increases the number of learnable parameters which leads to degradation in performance. This result is due to insufficient training data to effectively train large Transformer-GNN encoders.

4.4.2 Benchmark Results

We compare Tr-GNN to the current state-of-the-art results in MOT17 under the private detection setting, where, instead of using publicly available detections provided by the challenge, we opted to use a more advanced object detector [17] to obtain the detections, following the approach outlined in [14].

Table 4.3: Test set results on MOT17 benchmark on private detections. The SUSHI model published their results with varying maximum temporal edge distances. In SUSHI† and SUSHI, the maximum temporal edge distances are 150 and 512, respectively.

Method	IDF1↑	HOTA↑	MOTA↑	IDSW↓
QDTrack [46]	66.3	53.9	68.7	3378
MeMOT[66]	69.0	56.9	72.3	2724
GTR [15]	71.5	59.1	75.3	2859
FairMOT [42]	72.3	59.3	73.7	3303
GHOST [22]	77.1	62.8	78.7	2325
ByteTrack [14]	77.3	63.1	80.3	2363
FCG [16]	77.7	62.6	76.7	1737
MotionTrack[53]	80.1	65.1	81.1	1140
Deep OCSORT [67]	80.6	64.9	79.4	**1023**
UMCTrack [39]	81.0	65.7	80.6	1689
SUSHI† [9]	80.5	65.2	80.7	1335
SUSHI [9]	**83.1**	**66.5**	**81.1**	1149
Tr-GNN (ours)	80.8	65.4	80.5	1347

Table 4.3 presents the benchmark results of Tr-GNN, as well as other state-of-the-art approaches. Since the aim of our method is to establish correct data associations, the primary metric for us is the IDF1. Our work produces results on par with the state-of-the-art methods. When comparing our results to those of our base method, SUSHI† , we observe improvements in IDF1 and HOTA metrics, particularly when the maximum temporal edge distances are comparable. Specifically, with a maximum temporal edge distance of 150 for the SUSHI model, we outperform their results with our own maximum temporal edge distance set to 128. However, it's worth noting that SUSHI performs better when their maximum temporal distance is set to 512. Unfortunately, we were unable to train our model with such a high temporal distance due to computational constraints.

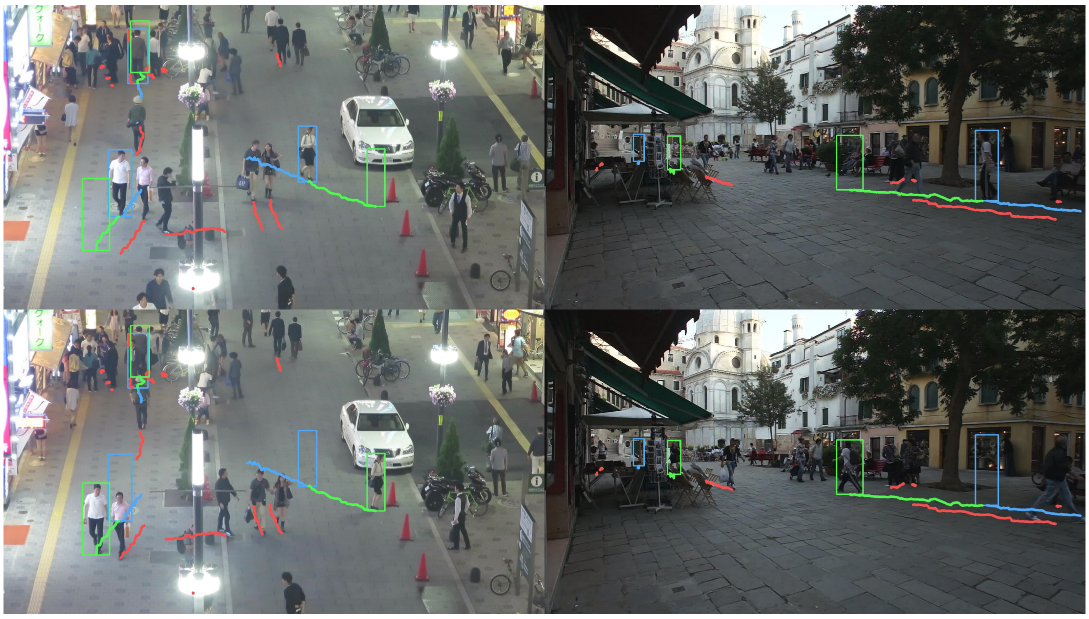


Figure 4.5: Results of visualizations of selected tracking examples. Blue lines indicates the query tracklet, red lines indicate the rejected matches, green line indicates the matched tracklet by the model.

In Figure 4.5, we present a visualization of our model's operation. Specifically, we illustrate the scenario where a query tracklet (depicted in blue) exists in time frames t to $t + \tau$, and we aim to predict its corresponding tracklet in time frames $t + \tau$ to $t + 2\tau$. Matched tracklets in future frames are represented in green, while tracklets connected to the blue tracklet in the bipartite graph but not matched by the model are shown in red.

4.5 Discussion

In this chapter, we introduced Tr-GNN, a joint Transformer-GNN encoder architecture designed for multi-object tracking. Inspired by recent works such as SUSHI, our model adopts a hierarchical approach and incorporates a novel joint Transformer and GNN encoder to effectively capture both motion and social interactions among tracklets. Through comprehensive ablation studies, we demonstrated the importance of bidirectional motion modeling, masked self-attention, and the reversal of tracklets in enhancing overall tracking performance. A key strength of our approach lies in its capability to model intricate interactions between objects, enabling more accurate data association even in challenging scenarios characterized by heavy occlusions or crowded environments. By simultaneously learning motion patterns and social interactions, Tr-GNN achieves performance on par with state-of-the-art methods on the MOT17 benchmark, demonstrating competitive efficacy.

However, there are notable limitations and potential areas for future improvement that warrant consideration. For instance, our reliance on pre-trained Re-Identification (Re-ID) features for visual information may limit the model's ability to fully exploit available visual cues in the data. Incorporating a dedicated visual feature extractor trained jointly with the tracking model could potentially enhance performance in this regard. Additionally, the computational complexity associated with the joint Transformer and GNN encoder may pose challenges in scaling the model to larger datasets or longer temporal sequences.

In conclusion, Tr-GNN represents a significant advancement in multi-object tracking, leveraging hierarchical modeling and joint Transformer-GNN encoders to effectively capture both motion and social interactions among objects. While the model achieves competitive performance on benchmark datasets, incorporating richer visual information and developing efficient pre-training strategies for transformer layers hold promise for further improving its efficacy and applicability in real-world scenarios.

CHAPTER 5

CONCLUSION

5.1 Summary

In this thesis, we have addressed the problem of multi-object tracking using an offline method based on the tracking-by-detection paradigm. To represent the regular interconnections among detections in each frame, we have leveraged graph representation, as graphs offer high flexibility in representing entities and their relations. Specifically, we have utilized recent advancements in graph representation learning to propagate information between different entities and extract high-level, more complex features.

We first explored the fundamental concepts and recent advancements in graph representation learning. We began by reviewing GNN architectures in MPNN framework which offer a flexible formulation for incorporating both node and edge features. We reviewed prominent models such as GCN, GAT, and GraphSAGE, each offering unique approaches to information propagation and feature aggregation. In the same chapter, we drew an analogy between the renowned Transformer architecture and GNNs, demonstrating that Transformers are a special type of GNNs where the underlying graph is fully connected. To this end, we reviewed Graph Representation Learning from various point of views.

Next, a comprehensive overview of multi-object tracking is provided in the thesis. MOT is a crucial field in computer vision with applications ranging from autonomous driving to surveillance systems. The chapter reviews various approaches used in MOT research, starting from general methods and progressing to the utilization of deep learning architectures. It discusses how motion and appearance ques are learned and utilized in MOT. Particularly in offline tracking, we focus on graph-based methods. These graph-based approaches offer enhanced performance compared to online frame-by-frame methods by minimizing association costs globally. By leveraging advancements in graph representation learning, these methods utilize deep networks not only for feature extraction but also for data association and modeling relationships between targets. Additionally, these models are predominantly end-to-end trainable, eliminating the need for assumptions or heuristics regarding short and long-term associations. Graph-based state-of-the-art tracking methods are thoroughly reviewed, their advantages and shortcomings are stated.

We utilize a joint Transformer-GNN architecture to generate embeddings of the tracklets, which are subsequently matched hierarchically. Our observations reveal that enabling bi-directional information propagation between two modalities significantly enhances tracking performance. Both motion and social interactions are implicitly modeled through tracklet embeddings. Experimental results are presented for the MOT17 dataset, which is specifically curated for the MOT challenge. Notably, participants are unable to access annotations for the test sequences, ensuring a fair evaluation environment for testing the developed methods. The dataset encompasses challenging scenarios including occlusions, camera motion, and similar appearances of targets. In the challenge, our model ranks 11th among all participants (approximately 200), showcasing its competitive performance in a diverse and complex tracking environment.

As a concluding remark for this summary, it can be stated that the thesis provides an extensive review of state-of-the-art techniques addressing the MOT problem. Furthermore, a novel approach is proposed and explored to tackle this challenge. The experimental results underscore the promising future of graph representation learning, not only in MOT but also in other data fields characterized by high-dimensional geometry and irregular structures.

5.2 Conclusions

In the MOT literature, we observed that graph-based methods often overlook the modeling of target motions, which is crucial for long-term tracking. Previous methods either disregard this aspect [8, 57, 9] or devise overly complex architectures [56, 7] and training schemes that deviate from the primary objective. In this thesis, our aim is to address this limitation by incorporating motion modeling through tracklet embedding, inspired by the recent successes of language models in handling sequential data. Additionally, to leverage the inherent symmetries in tracking data, we opted to utilize a graph structure and exploit advancements in Graph Representation Learning. The novel contribution of our method lies in effectively combining these two architectures to tackle the challenges of multi-object tracking. Specifically, we propose a bi-directional information propagation scheme between two modalities, enabling motion modeling through a Transformer encoder to be informed by social interactions among targets, and vice versa.

Our experimental results demonstrate advancements in identity preservation of tracked targets through our proposed Tr-GNN method. Through comprehensive ablation studies, we elucidate the usefulness of critical components of our approach, highlighting the pivotal role of bidirectional tracklet embedding. This is evidenced by an approximate 3% increase in both the HOTA and IDF1 scores, validating the effectiveness of utilizing bi-directional information propagation between motion and social interaction modeling. Furthermore, we identify the optimal number of Transformer-GNN layers crucial for achieving superior tracking accuracy.

Moreover, our method demonstrates competitive performance when compared to state-of-the-art approaches on the MOT17 benchmark, particularly excelling in the IDF1 metric. In IDF1 metric, our method achieves a score of 80.8%, surpassing the score of 80.5% achieved by the SUSHI† method, which our implementation is based on. This improvement is notable given comparable maximum temporal edge distances (ours: 128, theirs: 150), underscoring the efficacy of our approach in handling complex tracking scenarios characterized by heavy occlusion and crowded scenes. However, it is worth noting that the full model of SUSHI, operating with a maximum temporal length of 512, currently stands as the state-of-the-art method among all MOT trackers. Despite our attempts to increase our maximum temporal length, the use of a transformer encoder for every tracklet presented computational constraints, preventing us from training the model as desired.

Nevertheless, our experimental findings underscore the significance of our proposed Tr-GNN method, performing on par with state-of-the-art methods. In future research, we believe our approach could be further enhanced, especially by pre-training the transformer encoder with fragmented trajectories from various MOT datasets, akin to recent advancements in language models. For instance, in [68], authors demonstrate that the bidirectional language-knowledge model proposed in [11] significantly improves over baselines in the NLP domain when used in pre-training. A similar approach could be adopted in multi-object tracking to achieve a better-performing tracker.

As a final conclusion, this thesis provides experimental validation of the effectiveness of recent learnable deep neural network models and their interactions in addressing the challenges of multi-object tracking

REFERENCES

[1] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[6] C. Olah, "Understanding lstm networks," 2015. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[7] G. Wang, R. Gu, Z. Liu, W. Hu, M. Song, and J.-N. Hwang, "Track without appearance: Learn box and tracklet embedding with local and global motion patterns for vehicle tracking," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9856–9866, 2021.

[8] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[9] O. Cetintas, G. Braso, and L. Leal-Taixe, "Unifying short and long-term tracking

with graph hierarchies," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, March 2023.

[10] A. Wood, "W3c working group note 24 june 2014," 2014. `https://www.w3.org/TR/rdf11-primer/`.

[11] X. Zhang, A. Bosselut, M. Yasunaga, H. Ren, P. Liang, C. D. Manning, and J. Leskovec, "GreaseLM: Graph REASoning enhanced language models," in *International Conference on Learning Representations*, 2022.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[13] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler, "Learning by tracking: Siamese cnn for robust target association," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 418–425, 2016.

[14] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, (Berlin, Heidelberg), p. 1–21, Springer-Verlag, 2022.

[15] X. Zhou, T. Yin, V. Koltun, and P. Krähenbühl, "Global tracking transformers," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8761–8770, 2022.

[16] A. Girbau, F. Marqués, and S. Satoh, "Multiple object tracking from appearance by hierarchically clustering tracklets," in *In 33rd British Machine Vision Conference*, 2022.

[17] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[18] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, Sept. 2016.

[19] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, “Observation-centric sort: Rethinking sort for robust multi-object tracking,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9686–9696, 2023.

[20] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, 2017.

[21] Y.-H. Wang, J.-W. Hsieh, P.-Y. Chen, M.-C. Chang, H.-H. So, and X. Li, “Smiletrack: Similarity learning for occlusion-aware multiple object tracking,” in *AAAI Conference on Artificial Intelligence*, pp. 5740–5748, Mar. 2024.

[22] J. Seidenschwarz, G. Brasó, V. C. Serrano, I. Elezi, and L. Leal-Taixé, “Simple cues lead to a strong multi-object tracker,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13813–13823, 2023.

[23] L. Zhang, Y. Li, and R. Nevatia, “Global data association for multi-object tracking using network flows,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[24] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn, “Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 120–127, 2011.

[25] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, “Tracking without bells and whistles,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019.

[26] X. Zhou, V. Koltun, and P. Krähenbühl, “Tracking objects as points,” in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV*, p. 474–490, 2020.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML’17, p. 1263–1272, JMLR.org, 2017.

[28] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," 2018.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, 2015.

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[31] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8834–8844, 2022.

[32] Y. Jiang, S. Chang, and Z. Wang, "Transgan: Two pure transformers can make one strong gan, and that can scale up," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 14745–14758, Curran Associates, Inc., 2021.

[33] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, (Berlin, Heidelberg), p. 213–229, Springer-Verlag, 2020.

[34] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," in *International Conference on Learning Representations*, 2022.

[35] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, vol. 139, pp. 8748–8763, July 2021.

[36] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[38] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, “Strongsort: Make deepsort great again,” *IEEE Transactions on Multimedia*, vol. 25, pp. 8725–8737, 2023.

[39] K. Yi, K. Luo, X. Luo, J. Huang, H. Wu, R. Hu, and W. Hao, “Ucmctrack: Multi-object tracking with uniform camera motion compensation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 6702–6710, Mar. 2024.

[40] G. Bhat, M. Danelljan, L. Van Gool, and R. Timofte, “Learning discriminative model prediction for tracking,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6181–6190, 2019.

[41] P. Dai, R. Weng, W. Choi, C. Zhang, Z. He, and W. Ding, “Learning a proposal classifier for multiple object tracking,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2443–2452, 2021.

[42] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, “Fairmot: On the fairness of detection and re-identification in multiple object tracking,” *International Journal of Computer Vision*, vol. 129, pp. 3069–3087, 2021.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[44] X. Pan, P. Luo, J. Shi, and X. Tang, “Two at once: Enhancing learning and generalization capacities via ibn-net,” in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), pp. 484–500, Springer International Publishing, 2018.

[45] J. Pang, L. Qiu, X. Li, H. Chen, Q. Li, T. Darrell, and F. Yu, "Quasi-dense similarity learning for multiple object tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 164–173, 2021.

[46] T. Fischer, T. E. Huang, J. Pang, L. Qiu, H. Chen, T. Darrell, and F. Yu, "Qdtrack: Quasi-dense similarity learning for appearance-only multiple object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 15380–15393, 2023.

[47] W. Li, Y. Xiong, S. Yang, M. Xu, Y. Wang, and W. Xia, "Semi-tcl: Semi-supervised track contrastive representation learning," *arXiv preprint arXiv:2107.02396*, 2021.

[48] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1116–1124, 2015.

[49] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 152–159, 2014.

[50] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, p. 4225–4232, 2017.

[51] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 300–311, 2017.

[52] P. Dendorfer, V. Yugay, A. Ošep, and L. Leal-Taixé, "Quo vadis: Is trajectory forecasting the key towards long-term multi-object tracking?," in *Conference on Neural Information Processing Systems*, 2022.

[53] Z. Qin, S. Zhou, L. Wang, J. Duan, G. Hua, and W. Tang, "Motiontrack: Learning robust short-term and long-term motions for multi-object tracking," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17939–17948, 2023.

[54] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3701–3710, 2017.

[55] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, "Deep network flow for multi-object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2730–2739, 2017.

[56] P. Chu, J. Wang, Q. You, H. Ling, and Z. Liu, "Transmot: Spatial-temporal graph transformer for multiple object tracking," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4859–4869, 2023.

[57] I. Papakis, A. Sarkar, and A. Karpatne, "Gcnnmatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization," *arXiv preprint arXiv:2010.00067*, 2020.

[58] A. Rangesh, P. Maheshwari, M. Gebre, S. Mhatre, V. Ramezani, and M. M. Trivedi, "Trackmpnn: A message passing graph neural architecture for multi-object tracking," *arXiv preprint arXiv:2010.00067*, 2021.

[59] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, "Fastreid: A pytorch toolbox for general instance re-identification," in *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, (New York, NY, USA), p. 9664–9667, Association for Computing Machinery, 2023.

[60] A. C. Aras, T. Alikaşifoğlu, and A. Koç, "Graph receptive transformer encoder for text classification," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 10, pp. 347–359, 2024.

[61] Y. Lin, Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li, and F. Wu, "BertGCN: Transductive text classification by combining GNN and BERT," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, (Online), pp. 1456–1462, Association for Computational Linguistics, Aug. 2021.

[62] X. She, J. Chen, and G. Chen, "Joint learning with bert-gcn and multi-attention for event text classification and event assignment," *IEEE Access*, vol. 10, pp. 27031–27040, 2022.

[63] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *ACM Transactions on Graphics (TOG)*, 2019.

[64] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “Mot16: A benchmark for multi-object tracking,” *arXiv preprint arXiv:1603.00831*, 2016.

[65] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, (San Diega, CA, USA), 2015.

[66] J. Cai, M. Xu, W. Li, Y. Xiong, W. Xia, Z. Tu, and S. Soatto, “Memot: Multi-object tracking with memory,” 2022.

[67] G. Maggiolino, A. Ahmad, J. Cao, and K. Kitani, “Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification,” in *2023 IEEE International Conference on Image Processing (ICIP)*, pp. 3025–3029, 2023.

[68] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. Liang, and J. Leskovec, “Deep bidirectional language-knowledge graph pretraining,” in *Neural Information Processing Systems (NeurIPS)*, 2022.

[69] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Signal Processing Magazine*, vol. 30, p. 83–98, May 2013.

[70] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, “Geometric deep learning on graphs and manifolds using mixture model cnns,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5425–5434, 2017.

[71] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” in *Applied and Computational Harmonic Analysis*, pp. 129–150, 2011.

[72] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in *2nd International Conference on Learning*

Representations, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (Y. Bengio and Y. LeCun, eds.), 2014.

[73] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (Red Hook, NY, USA), p. 3844–3852, Curran Associates Inc., 2016.

[74] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[75] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, p. 246309, 2008.

[76] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixe, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," *International Journal of Computer Vision*, vol. 129, p. 548–578, Oct. 2020.

[77] J. Luiten, "How to evaluate tracking with the hota metrics," 2021. https://jonathonluiten.medium.com/how-to-evaluate-tracking-with-the-hota-metrics-754036d183e1.

APPENDIX A

REVIEW OF SPECTRAL AND SPATIAL GRAPH THEORY

A.1 Spectral Graph Theory

In this appendix, we will review the graph spectral domain by drawing an analogy between the classical frequency domain in signal processing. Classical signal processing techniques cannot be directly applied here due to the irregular structure of the graph domain [69], so we will define the Fourier transform and frequency notion in the graph spectral domain.

Consider an undirected, weighted graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ which consists of a finite set of vertices $\mathcal{V} := \{1, ..., n\}$, a set of edges $(i, j) \in \mathcal{E}(G)$ and represented by *adjacency matrix* $\mathbf{W} = (w_{ij})$ where $w_{ij} = 0$ if $(i, j) \notin \mathcal{E}(G)$ and $w_{ij} > 0$ if $(i, j) \in \mathcal{E}(G)$. The (unnormalized) graph Laplacian is defined as

$$L := D - \mathbf{W} \tag{A.1}$$

D is the *degree matrix*; it is a diagonal matrix where the diagonal element $D_{ii} = \sum_{j \in N(i)} w_{ij}$ which is the sum of the weights of all the edges incident to vertex i. The graph Laplacian is a symmetric, positive semi-definite matrix; thus it is always diagonalizable,

$$L = \Phi^T \Lambda \Phi \tag{A.2}$$

In the eigendecomposition of the Laplacian, $\Phi = \{\phi_1, ..., \phi_n\}$ are the eigenvectors and they form an orthonormal set, $\Lambda = diag(\lambda_1, ...\lambda_n)$ is the diagonal matrix of corresponding eigenvalues. The eigenvectors play the role of Fourier basis in classical signal processing and the eigenvalues can be interpreted as frequencies [70]. Let a signal $\mathbf{f} = \{f_1, ..., f_n | f_i \in \mathbb{R}^d\}$ is defined on the vertices of the graph.

$$\begin{aligned} f &= \sum_{k=1}^{n} < \phi_k, f > \phi_k \\ &= \sum_{k=1}^{n} \hat{f}_k \phi_k \\ &= \underbrace{\Phi \hat{f}}_{\mathcal{F}^{-1}(\hat{f})} \end{aligned} \tag{A.3}$$

Equation (A.3) shows the decomposition of function f with Fourier functions [71]. Thus the *Fourier transform* is defined on the graph as $\Phi^T \mathbf{f}$. Given two signals $\mathbf{f}, \mathbf{g}$ on graph, their convolution is equivalent to the element-wise multiplication of their Fourier transforms.

$$\mathbf{f} \circledast \mathbf{g} = \Phi\big((\Phi^T \mathbf{f}) \odot (\Phi^T \mathbf{g})\big) \tag{A.4}$$

Hence Equation (A.4) defines the convolution theorem on the graph domain so that operators like filtering on the graph can be defined.

A.2 Spectral Graph Neural Networks

Spectral CNN [72]

The spectral filters introduced in Appendix A.1 can be learned through backpropagation. The pioneering work by Bruna et al. [72] introduced the concept of learning spectral filters on graphs, termed as *Spectral Graph Neural Networks*. This work is the generalization of CNNs to signals defined on more general domain such as graphs. Spectral convolution layer is defined as,

$$\mathbf{f}_j^{out} = \sum_{i=1}^{d} \Phi_k \mathbf{G}_{j,i} \Phi_k^T \mathbf{f}_i^{in} \tag{A.5}$$

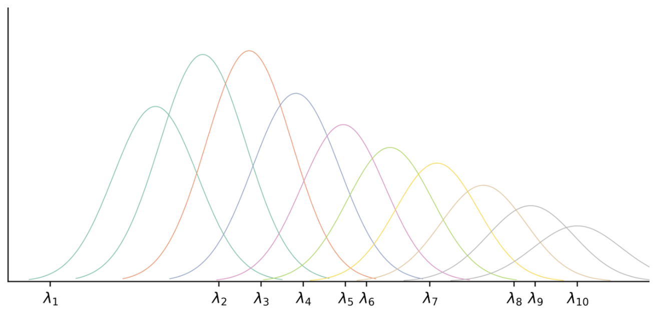


Figure A.1: Cubic splines.

Equation A.5, $\mathbf{F}^{in} \in \mathbb{R}^{n \times d} = (\mathbf{f}_1^{in}|...|\mathbf{f}^{in}d)$ represents a d-dimensional input signal, and $\mathbf{F}^{out} \in \mathbb{R}^{n \times d'} = (\mathbf{f}^{out}1|...|\mathbf{f}^{out}d')$ represents a d'-dimensional output signal defined on the graph. Φ_k represents the first k eigenvectors of the graph Laplacian, and $\mathbf{G}j, i$ is a $k \times k$ diagonal matrix of spectral learnable filters. With this formulation, the convolution is performed in the frequency domain, enabling filters similar to those in CNNs to be established in the graph domain. However, this formulation has some drawbacks: (i) the filters are dependent on the Fourier basis of the graph Laplacian, and (ii) there is no guarantee that those filters defined in the frequency domain are localized in the spatial domain, since they are learnable and arbitrary.

To address the latter issue, the authors [72] enforced smooth spectral filters such that, according to Parseval's Identity, smoothness in the frequency domain corresponds to localization in the spatial domain. This approach allows for the learning of filters that are shared across locations but also well localized in the original graph domain. They used parametric filters of the form,

$$g_i = \sum_{l=1}^{r} \alpha_l \beta_l(\lambda_i) \tag{A.6}$$

In Equation A.6, $\beta_l(\lambda)$ represents the fixed cubic spline kernels as shown in Figure A.1, and α_l's are the interpolation coefficients. Therefore, in each layer of the spectral convolution, it is sufficient to learn r interpolation coefficients for each filter. Enforcing smoothness in the frequency domain and reducing the number of learnable parameters to r, independent of the input size $|\mathcal{V}|$, the number of vertices, is achieved through this approach.

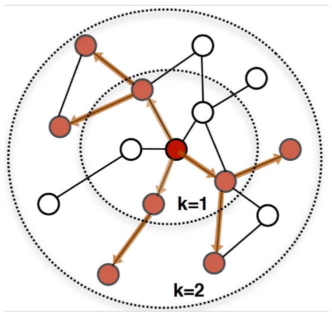


Figure A.2: Example of 1-hop and 2-hop neighborhoods [12].

ChebNet [73]

The spectral graph convolution operation tends to be computationally expensive because it necessitates eigendecomposition, resulting in a complexity that scales quadratically in the number of vertices $|\mathcal{V}|$. Defferrard et. al. [73] proposed a method that constructs filters directly on the Laplacian matrix which does not require the explicit computation of the Laplacian eigenvectors by using recurrent Chebyshev polynomials. This approach not only accelerates the convolution operation but also ensures that the filters are precisely localized in the k-hop neighborhood. This is due to the fact that the Laplacian is a local operator that only affects the 1-hop neighborhood of the graph. Leveraging the advantageous property of the graph Laplacian, the application of n powers of the graph Laplacian extends the influence to n-hop neighborhoods.

A.3 Spatial Graph Neural Networks

Spectral GNN methods have a significant drawback: they are inherently *transductive*; learned spectral features on one graph cannot be trivially applied to another graph, since they directly apply filters to the graph Laplacian or utilize the graph dependent Fourier basis. Given that the same graph can be represented through permutations of its nodes, numerous isomorphic graph representations exist even for a single graph.

To address these limitations hindering the generalizability of the model, researchers have pursued the implementation of spatial filters instead of localized spectral filters. This adaptation aims to enhance GNN models, enabling them to better generalize to unseen graphs. The seminal paper by Kipf and Welling [2] employs the first-order approximation of localized spectral filters on graphs [73]. This approximation confines the convolutional filters to be localized within 1-hop neighborhoods. The authors [2] construct a multi-layer Graph Convolutional Network (GCN) with the following layer-wise propagation rule:

$$\mathbf{f}^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{\mathbf{W}}\tilde{D}^{-\frac{1}{2}}\mathbf{f}^{(l)}\mathbf{g}^{(l)}) \tag{A.7}$$

where $\tilde{\mathbf{W}}$ represents the adjacency matrix of the undirected graph with self-loops, defined as $\tilde{\mathbf{W}} = \mathbf{W} + I$. In Equation A.7, D_{ii} signifies $\sum_{j \in N(i)} \tilde{w}_{ij}$, and $\mathbf{g}^{(l)} \in \mathbb{R}^{d \times k}$ stands as our spatial filter for layer (l), constituting a learnable weight matrix. Meanwhile, $\mathbf{f}^{(l)} \in \mathbb{R}^{N \times d}$ represents the signal or features defined on the graph at layer (l). The function $\sigma(\cdot)$ denotes any activation function, such as ReLU or Sigmoid.

In this formulation, the filter parameters are shared across graphs. Successive application of this layer-wise propagation k times enables reaching k-hop neighbors, similar to the receptive field concept in CNNs. An anchor node can gather information from all of its k-hop neighbors. Thus, by stacking multiple layers, deep GNN models can be constructed. GCN partially mitigates the inherent transductivity of spectral GNN methods by not relying directly on the graph Laplacian and instead using localized filters that operate within the 1-hop neighborhood of each node. This approach lessens the dependence on eigenbasis and enables the model to be more flexible and adaptable to varying graph structures.

APPENDIX B

MOTIVATION OF GEOMETRIC DEEP LEARNING AND RELATION TO GRAPHS

Geometric Deep Learning (GDL) [74, 1] is a recently emerged field whose main purpose is to generalize deep learning models to operate on both Euclidean and non-Euclidean domains. It provides a common mathematical framework to study neural network models such as CNNs, RNNs, GNNs, and Transformers, while exploiting the inductive biases inherent in the symmetries of the data.

The inherent symmetries present in deep learning models have been pivotal since the introduction of convolutional neural networks. These symmetries leverage the translation equivariance property of the convolution operation, which operates on a grid. This property not only exploits local correlations within natural image domains but also introduces a localized weight sharing schema to effectively construct filters. Moreover, the stacking of layers in CNNs establishes hierarchical feature spaces, transitioning naturally from fine-grained to coarse representations of the domain. Additionally, pooling layers in these models introduce translation invariance within patches, greatly benefiting downstream tasks such as image classification, as the outcome is independent of the object's specific position. On the other hand, multi-layer perceptrons, which can approximate any smooth function according to the universal approximation theorem, cannot achieve the same performance levels as CNNs due to the lack of shift-invariance property. Thus, GDL offers a constructive approach to integrating symmetry priors into neural network architectures, resembling the approach observed in CNNs.

At this point, it is essential to provide simple definitions to lay the groundwork for an introductory understanding of GDL. Informally, as per [1], a symmetry of an object or system refers to a transformation that preserves a specific property of that object or system, keeping it unchanged or invariant. Symmetries are defined within symmetry groups $\mathcal{O}$, characterized by a binary operation $\circ : \mathcal{O} \times \mathcal{O} \to \mathcal{O}$, satisfying properties such as Associativity, Identity, Inverse, and Closure for all elements h, l of the group. For example, in computer vision, object categories exhibit invariance to shifts, while in social networks, community structures maintain invariance under permutations of node ordering. Selecting network models that leverage these symmetry priors is crucial; in CNNs, for instance, the convolution operation is shift-equivariant, thus incorporating an inductive bias aligned with inherent symmetries.

Hence according to Bronstein [1], considering a data domain Ω, some metric $\mathcal{C}$ in the domain and signal defined on that domain $\mathcal{X}_j(\Omega_j, \mathcal{C}_j) := \{x_j : \Omega_j \to \mathcal{C}_j\}$ the neural network architectures can be constructed with simple building blocks defined as,

- Linear *O-equivariant* Layer $B : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$ satisfying $B(\mathbf{g}.x) = \mathbf{g}.B(x)$ for all $\mathbf{g} \in \mathcal{O}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$
- Non-linearity $\sigma : \mathcal{C} \to \mathcal{C}'$ applied element-wise as $(\sigma(x))(u) = \sigma(x(u))$
- Local Pooling $P : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$
- *$\mathcal{O}$-invariant* Layer (global pooling) $A : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$ satisfying $A(\mathbf{g}.\mathbf{x}) = A(x)$ for all $\mathbf{g} \in \mathcal{O}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$

Using these blocks allows constructing $\mathcal{O} - \textit{invariant}$ functions $f : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$ of the form,

$$A \circ \sigma_j \circ B_j \circ P_{j-1} \circ P_1 \circ \sigma_1 \circ B_1$$

By selecting a suitable symmetry group O, one can effectively employ these foundational elements to design neural architectures that closely align with the inherent inductive biases of the domain Ω.

In our research, we will leverage permutation invariance and other geometric priors apparent in the underlying data domain, which in this case involves tracking sequences. As we represent these sequences using graph structures, considerations extend to both the ordering of nodes and the relative positioning of bounding boxes across different temporal axes in the image plane.

Table B.1: Different choices of architecture, domain and symmetry groups for GDL [1]

Architecture	**Domain Ω**	**Symmetry Group $\mathcal{O}$**
CNN	Grid	Translation
Spherical CNN	Sphere / SO(3)	Rotation SO(3)
GNN	Graph	Permutation
Transformer	Complete Graph	Permutation
LSTM	1D Grid	Time Warping

Different GDL methods differ in their choice of domain, symmetric group and specific architectural designs. For widely used and famous neural network architectures, the design choices can be summarized as in the Table B.1.

In the subsequent section, we will provide an example architecture that is designed with geometric considerations for point cloud classification and segmentation.

B.1 EdgeConv

The EdgeConv neural network architecture, proposed by Wang et al. [63], is designed for point cloud classification and segmentation. The primary goal of the EdgeConv architecture is to construct a model capable of effectively leveraging both local and global geometric priors inherent in point clouds. The model must maintain permutation invariance, which is essential for handling unordered point clouds. However, achieving permutation invariance solely by operating on individual points would disregard the local geometric relationships among them. EdgeConv addresses this issue by capturing local geometric structures while preserving permutation invariance. Furthermore, an important property of the proposed method [63] is translation invariance, which will be elaborated on later.

To implement EdgeConv, the authors construct a undirected graph $G(\mathcal{V}, \mathcal{E})$ with node features $X = \{x_1, ..., x_n\}, x_i \in \mathbb{R}^F$ of the point cloud, where each point is connected to its k-nearest neighbors. Importantly, the set of k-nearest neighbors varies from layer to layer, corresponding to dynamic graph construction in each layer. This adaptive approach allows the model to capture varying degrees of local geometric information across different layers, enhancing its ability to discern complex patterns within the point cloud. The message passing equation of the proposed method is given as,

$$x'_i = \max\{h_\theta(x_i||x_j - x_i)\} \quad for\ j \in N_{x_i}$$

where max is an element-wise maximum in the neighborhood dimension, and || indicates the concatenation, N_{x_i} is the neighbors of the node x_i and h_θ is a learnable function. The author argues that this asymmetric message function explicitly captures global shape structure with local neighborhood information. This is achieved by leveraging not only the central node x_i but also the relative distances of the neighbors with respect to the central node. Furthermore, the inclusion of relative distances on the edges $(x_j - x_i)$ offers a degree of partial translation invariance. Through the incorporation of such right inductive biases tailored to the underlying data domain, the model achieves state-of-the-art results in the classification task on point clouds.

The effectiveness of the partial translation-invariance property of message passing networks in multi-object tracking has also been investigated in a recent paper by Rangesh et al. [58]. In the ablation studies examining the choice of message function, it was found that the asymmetric message function significantly outperformed other alternative. Hence, even though the research is relatively new on constructing deep learning models considering geometric priors in the underlying data domain, it is possible to explain such heuristic choices in other studies with GDL. The reason of making the introduction to GDL in this appendix is because we used a a similar massage passing operation in our work, as described in Chapter 4.3.2.

APPENDIX C

MULTI-OBJECT TRACKING PERFORMANCE METRICS

In this appendix, we provide a detailed explanation of performance metrics commonly used in multi-object tracking research. Understanding these metrics is essential for interpreting and assessing the results presented in Chapter 4.

Table C.1: Metric definitions for assessing multi-object tracking performance.

True Positive	:	Predicted detection matched with ground-truth detection
False Negative	:	Ground truth detection exists but predicted detection was missed
False Positive	:	Predicted detection exists for no ground-truth detection
ID Switch	:	Tracker wrongfully swaps object identities or when a track was lost and reinitialised with a different identity

C.1 MOTA

In Multi-Object Tracking Accuracy (MOTA) [75], matching conducted at the detection level. A bijective (one-to-one) mapping is established between predicted detections and ground-truth detections in each frame. True Positives (TP), False Negatives (FN), False Positives (FP), and ID switc-hes (IDSW) are defined in Table C.1.

$$\mathrm{MOTA} = 1 - \frac{|\mathrm{FN}| + |\mathrm{FP}| + |\mathrm{IDSW}|}{|\mathrm{gtDets}|} \tag{C.1}$$

MOTA does not include a measure of localization error, and detection performance significantly outweighs association performance, as discussed in [76]

C.2 IDF1

In [76], the importance of IDF1 is emphasized, which prioritizes the accuracy of associations over detections. IDF1 establishes a one-to-one mapping between ground truth trajectories and predicted trajectories to determine their presence, unlike MOTA, which associates them based on object detection at each time step.

IDF1 defines Identity True Positives (IDTPs) as occurrences when a predicted identity matches a ground truth identity when a similarity threshold (S) is met for a certain percentage (α) of trajectories. The metric calculates the ratio of correctly identified detections to the average number of ground truth and predicted detections. Furthermore, the Hungarian algorithm is employed in IDF1 to select trajectories to be matched, aiming to minimize the sum of the number of Identity False Positives (IDFP) and Identity False Negatives (IDFN).

The ID-recall, ID-precision and IDF1 are calculated as follows:

$$\begin{aligned}\mathrm{ID-recall} &= \frac{|\mathrm{IDTP}|}{|\mathrm{IDTP}| + |\mathrm{IDFN}|}\\ \mathrm{ID-precision} &= \frac{|\mathrm{IDTP}|}{|\mathrm{IDTP}| + |\mathrm{IDFP}|}\end{aligned} \tag{C.2}$$

$$\mathrm{IDF1} = \frac{|\mathrm{IDTP}|}{|\mathrm{IDTP}| + 0.5|\mathrm{IDFN}| + 0.5|\mathrm{IDFP}|} \tag{C.3}$$

C.3 HOTA

Higher Order Tracking Accuracy (HOTA) [76] is proposed to mitigate the deficiencies of mostly used MOTA [75] metric. HOTA provides a single score for tracker evaluation which fairly combines all different aspects of tracking evaluation, it is the combination of three IoU scores (detection, association and localization) [77].

Localization IoU: measures the spatial alignment (IoU) between *a single predicted detection* and *a corresponding ground-truth detection.* Therefore, the overall localization accuracy is calculated as the average of the Localization IoU scores across all pairs of matches between predicted and ground-truth detections in the entire dataset.

$$\mathrm{LocA} = \frac{1}{|\mathrm{TP}|} \sum_{c \in \mathrm{TP}} \text{Loc-IoU}(c) \tag{C.4}$$

Detection IoU: evaluates how well the *set of all predicted detections* align with the *set of all ground-truth detection.* The intersection of predicted and ground-truth detection should exceed a certain threshold, e.g. $\text{Loc-IoU} > 0.5$. Since it is possible for a single predicted detection to overlap with multiple ground-truth detections, the Hungarian algorithm is employed to establish a one-to-one correspondence between predicted and ground-truth detections. These paired detections, termed True Positives (TP), represent instances of intersection between the two detection sets. Predicted detections lacking a corresponding match are categorized as False Positives (FP), while ground-truth detections without a match are classified as False Negatives (FN). Consequently, the detection IoU is computed as follows,

$$\text{Det-IoU} = \frac{|\mathrm{TP}|}{|\mathrm{TP}| + |\mathrm{FN}| + |\mathrm{FP}|} \tag{C.5}$$

The overall detection accuracy DetA can be defined as Det-IoU of the whole dataset.

Association-IoU: measures the effectiveness of a tracker in connecting detections across consecutive frames to form consistent identities, with respect to the known identity links provided by ground-truth tracks. The overlap between two tracks is quantified by counting the number of correct matches, termed True Positive Associations (TPA). Detections in the predicted track that aren't matched to any ground-truth tracks, or are matched incorrectly, are categorized as False Positive Associations (FPA). Similarly, detections in the ground-truth track that aren't matched to any predicted detections are labeled False Negative Associations (FNA). The Association IoU can be computed as,

$$\text{Ass-IoU} = \frac{|\text{TPA}|}{|\text{TPA}| + |\text{FNA}| + |\text{FPA}|} \tag{C.6}$$

The overall Association Accuracy in the entire dataset is calculated as,

$$\text{AssA} = \frac{1}{|\text{TP}|} \sum_{c \in \text{TP}} \text{Ass-IoU}(c) \tag{C.7}$$

Given the definitions of the three IoU metrics, the HOTA score combines these three metrics. It is calculated as:

$$\begin{aligned} \text{HOTA}_\alpha &= \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha} \\ \text{HOTA} &= \frac{1}{19} \sum_{\alpha=0.05}^{0.95} \text{HOTA}_\alpha \end{aligned} \tag{C.8}$$

Previously, both DetA and AssA were defined based on a Hungarian matching procedure using a specific Loc-IoU threshold (α). As DetA and AssA scores are influenced by Loc-IoU values, these scores are computed across various α thresholds. For each threshold value, the final score is calculated as the geometric mean of the detection score and the association score. By integrating over different α thresholds, localization accuracy is incorporated into the final score. For in-depth explanation and analysis of HOTA metric, please refer to [76, 77].