TIMEGRAPHS: GRAPH-BASED TEMPORAL REASONING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Paridhi Maheshwari
May 2023

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Jure Leskovec) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Stefano Ermon)

Approved for the Stanford University Committee on Graduate Studies

Abstract

Interactions over time result in complex dynamic behaviors. Models that can reason about such interactions are essential for better understanding of individual as well as group dynamics in many real-world scenarios. Most approaches for temporal reasoning represent time dynamics as a linear sequence of events. However, simple linear representations are insufficient to fully express rich and complex behaviors. Here we propose TimeGraphs, a novel approach that represents dynamic interactions as a temporal knowledge graph of events rather than as a linear sequence. Temporal knowledge graphs capture dynamics in a compact representation and thus allow us to perform reasoning at different time scales. We develop a self-supervised method that can construct event hierarchy from a sequence of temporal observations. The construction approach is scalable and works incrementally in a streaming fashion. We illustrate the capabilities of TimeGraphs on several curated datasets for complex inference tasks, including NBA basketball games, a football simulator, The Resistance games and the MOMA human activity dataset. TimeGraphs achieves up to 7% improvement over state-of-the-art on the event prediction and recognition tasks on these diverse datasets. We also demonstrate powerful capabilities of our approach, including zero-shot generalization, robustness to data sparsity, and streaming data flow. With its ability to capture complex dynamics, TimeGraphs is applicable to a wide range of domains as well as downstream tasks.

Acknowledgments

I would like to express my sincere gratitude to Professor Jure Leskovec for giving me an opportunity to work with him as part of the SNAP Lab at Stanford University. His invaluable guidance and constructive feedback have been instrumental in shaping my research journey, without which this thesis would not have been possible.

I am also thankful to Professor Stefano Ermon for his role as my secondary advisor and providing feedback and suggestions on my research thesis.

I would also like to thank my collaborators and mentors Hongyu Ren, Rok Sosič and Yanan Wang for their insightful discussions, valuable suggestions, and technical assistance throughout the project. Their expertise and knowledge have greatly enriched my learning experience.

I am immensely grateful to my parents and brother, without whose constant love and support, this journey would not have been possible. Their unwavering belief in me and their encouragement at every step of the way have been instrumental in helping me achieve this milestone. I also extend my heartfelt thanks to Ritwick, who has been my pillar of strength throughout this process. His insight and support have helped me navigate the highs and lows of graduate school. I am truly blessed to have such amazing people in my life, and I will always be grateful for their love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Interactions often result in complex behaviors. These are observed in many real-world situations. For example, human interactions can be either antagonistic, such as two teams competing in a sports game, or cooperative, such as interactions between multiple people with a common purpose, for example, having dinner in a restaurant. An important characteristic of such complex behaviors is that they are temporal as they involve interactions occurring over a period of time. The information about the behaviors is usually recorded as a sequence of observations, each observation capturing the state of the people or objects at a certain time interval.

Currently such complex behaviors are modeled using sequence models, where the observations are recorded at regular time intervals, which results in long sequences of individual observations. Take for example an NBA basketball game, which can be represented as a sequence of positions of all the players and the ball, taken at a rate of 25 frames per second. However, dynamics of an NBA game is complex and spread over time. There exist complex long-range dependencies in the way the game evolves and develops. For instance, in a basketball game, there may be a prolonged period of inactivity where a player holds the ball for a few seconds and no significant changes take place on the court. Conversely, there may also be a fast-paced and complex multi-person attack on the basket, involving multiple simultaneous events in a short span of time.

Using a simple sequence-based model to capture and model the dynamics of such complex behaviors is infeasible in practice as it would require large modeling capacity and ability to reason over long-range events. At the same time it is hard for sequence models to learn how to be selective about the attention each individual frame gets. So, there is a real need for a general approach that takes a sequence of observations and is able to reason about long-range complex temporal behaviors, which includes tasks such as event prediction and recognition.

Such an approach would be extremely useful in many practical applications, including predicting when some event will happen (a goal in football or guests leaving a restaurant) or which event happened at a certain time (ball passing in a game or paying a bill in a restaurant). In video

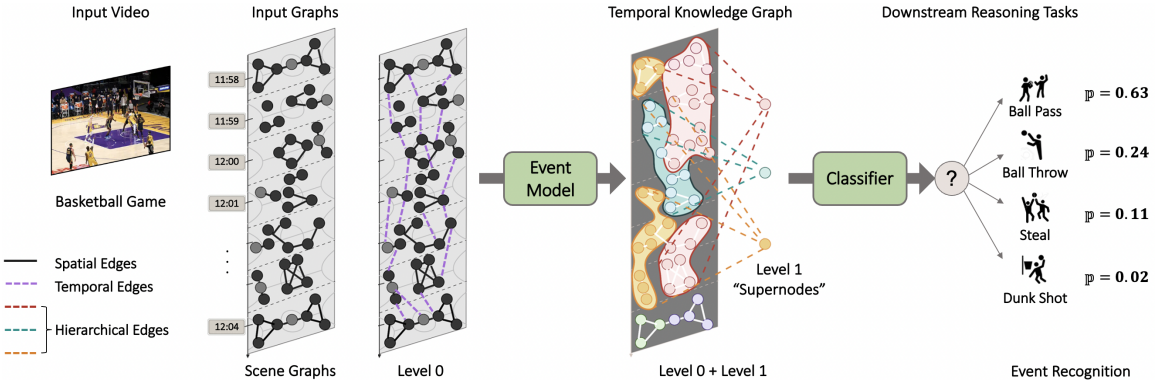

Figure 1.1: *TimeGraphs Architecture*: We take as input frame-wise scene graphs (obtained from video or sensors). We add spatial as well as temporal connections to obtain level 0 of our temporal knowledge graph. Next, we use our event model to create level 1 hierarchical events that can uncover latent dependencies at multiple scales. Finally, we leverage the rich temporal knowledge for various downstream reasoning tasks, such as event recognition, using a separate classifier module.

processing, for example, a recognition model could answer queries, such as "find me all cases where the guests and the waiter were arguing", which would result in significant time savings by reducing the need for human effort.

1.1 Proposed Approach

Here we propose *TimeGraphs*, a novel graph-based approach for temporal reasoning over complex events. The key insight of our approach is that time is not linear but instead should have a richer relational structure. We consider time and its potential future realization not as a linear sequence, but as a graph representation, called *temporal knowledge graph of events*, where nodes are connected by spatial as well as temporal edges. The graph captures important spatial-temporal patterns in data, and our hierarchical approach allows for expressing events at higher abstraction levels. Such a hierarchical model allows us to detect both short-term and long-term events in a scalable way by identifying temporal dependencies at multiple scales.

Our work draws inspiration from cognitive science and neuroscience literature [26, 3] which postulates that humans segment and categorize events into consistent groups. Furthermore, individuals consciously encode those ongoing activities as a hierarchy—a process known as the event segmentation theory [21] or the hierarchical bias hypothesis [43].

We build our graph using a self-supervised graph pooling method in order to build a hierarchy of events (see Figure 1.1). This graph pooling module, termed *event model*, is trained using a contrastive objective based on mutual information maximization. Once the event model is trained, it can be used to build the temporal knowledge graph with hierarchical events. The graph is then

used to train a graph neural network for downstream applications. An important characteristic of our approach is that it operates in a streaming fashion, reading one input observation at a time.

Figure 1.2: This example depicts a complex event modeling scenario from a football game with fine-grained event categories. Predicting whether this sequence will lead to a `'scoring'` or `'shooting'` or `'ball passing'` event is a difficult task and requires an understanding of the current game dynamics and strategies as well as players' capabilities and intentions.

We illustrate the capabilities of TimeGraphs on several curated datasets for complex inference tasks. The downstream task involves reasoning about videos and classifying them into a pre-defined set of event categories. We consider multi-player games that feature dynamical aspects of adversarial groups such as NBA basketball games, football games from Google's simulator [20], and Resistance games which comprise of evolving face-to-face interactions in a multi-player social game [1, 19]. Lastly, we apply TimeGraphs to the video-based human activity classification dataset Multi-Object Multi-Actor Activity Parsing (MOMA) [24]. Our results show that TimeGraphs outperforms traditional sequence-based methods and achieves up to 7% improvement on these diverse datasets.

Furthermore, we demonstrate additional powerful capabilities of TimeGraphs using extensive ablation experiments: (i) generalization to unseen games/teams and good zero-shot performance; (ii) robustness against frame rate and hence, data sparsity; (iii) ability to handle streaming data and improve accuracy as data flows in; (iv) accurate predictions even for longer time horizons, i.e., when we need to forecast an event far in the future.

1.2 Contributions

In summary, our key contributions are:

- A novel approach that represents time as a temporal knowledge graph of entities and events rather than as a linear sequence.
- An approach to pre-train the event model in a self-supervised manner based on mutual information optimization.
- A deep learning-based TimeGraphs pipeline that constructs a temporal knowledge graph of hierarchical events from a sequence of temporal observations.
- Extensive experiments on multiple datasets, demonstrating state-of-the-art performance of our graph-based approach.

- Analysis to illustrate powerful capabilities of TimeGraphs.

The thesis is organized as follows. Chapter 2 discusses the prior work in the field of temporal reasoning from data. Chapter 3 presents our proposed method TimeGraphs for temporal reasoning with graphs and explains the underlying principles. In Chapter 4, we describe the datasets and baselines for our task as well as the experiments performed to evaluate the performance of TimeGraphs. In Chapter 5, we provide extensive ablation experiments to demonstrate powerful capabilities of our approach. Finally, we conclude the thesis in Chapter 6 and provide directions for future work.

Chapter 2

Related Work

In video understanding, event prediction and recognition has received considerable attention from the research community due to its numerous potential uses such as sports analysis, video surveillance, and so on. Deep features trained on large-scale supervised datasets have enabled dramatic improvements in various computer vision tasks. However, most of these advancements focus on analyzing images, while performance on video understanding is far from satisfactory [17].

2.1 Action recognition in Videos

The closest related task in existing literature is *action recognition in videos* which is a popular task in the computer vision community. Prior works commonly adopt the following types of frameworks.

Space-time networks learn context from the temporal axis, I3D builds an inflated 3D ConvNet by inflating 2D filters and pooling kernels into 3D [5]. Qiu et al. [29] propose bottleneck building blocks to recycle off-the-shelf 2D residual networks for a 3D CNN.

Multi-stream networks model heterogeneous inputs using multiple convolutional networks to boost performance for action recognition [35, 38, 12, 14]. The spatial CNN learns from individual frames, while the temporal CNN recognizes objects using the optical flow field. Wang et al. [39] develop Temporal Segment Networks (TSN) which combines sparse temporal sampling strategy and video-level supervision to capture long-range temporal structure.

Hybrid networks combine the advantages from the architectures of language domain by adding a recurrent layer on top of the 2D CNNs. Recent works have explored LSTMs [6] and transformers [27] within this context. Yue et al. [42] study several pooling strategies and recurrent models for aggregating features learnt from CNNs. Action Transformer Network [10] is an extension of I3D to incorporate region proposal networks.

These approaches, however, rely on the appearance of specific entities and learn correlations between the scene context and object occurrence [44, 11]. Strikingly, even video benchmarks for this

task suffer from implicit biases in scene context and object appearance that can outweigh changes in time. For instance, recognizing the action of *'playing football'* given a field and ball in the background is a trivial task. They do not perform well in complex scenarios, as depicted in Figure 1.2, where the underlying structure is governed by transformations and temporal relations. To facilitate true understanding of videos, it is important to reason about their long-term behavior and also model latent concepts such as intentions and causal relationships [34, 4], which is beyond the capabilities of the current approaches. The inherent limitation of these methods is that they discretize spatial and temporal axes by having separate streams. In this work, we develop TimeGraphs where we reason about the two axes together in a unified framework and learn the rich spatio-temporal structure in a hierarchical manner.

2.2 Skeleton-based Human Action Recognition

Another common task is *skeleton-based human action recognition* which involves predicting activities from skeleton representations (coordinates of key joints) of human bodies as opposed to raw videos. These works [40, 33] construct spatio-temporal graphs from the frame-level skeleton annotations and employ Graph Neural Networks (GNNs) for classification. Liu et al. [23] propose a graph convolution operation (G3D) to facilitate information flow across space and time.

Our work significantly differs from the above approaches as we are not restricted to skeleton data, but use structured graph representations that describe the underlying scene as objects and their relations. Scene graphs have demonstrated improvements in a variety of vision tasks including image captioning [28, 41], image retrieval [32, 25] and visual relationship detection [18]. Although these structured representations have been increasingly used for frame-by-frame action recognition [15], their potential has not been fully realized for studying temporal events. Additionally, we do not limit to human activity recognition, and classify much more fine-grained and complex scenarios involving multi-player dynamics.

Chapter 3

Method

3.1 Temporal Reasoning

Without any loss in its generality, we describe our method in the context of temporal reasoning in videos. This domain is highly demanding, offering a wide range of datasets and benchmarks for the development of new methods that deal with temporal data. Consider a video consisting of a sequence of frames. Using standard vision methods [36], the frames can be transformed to structured scene graphs which contain information about the objects and their relations. At time t, we thus obtain the graph $G_t = (\mathcal{V}_t, \mathcal{E}_t)$, where $\mathcal{V}_t$ is the set of nodes and $\mathcal{E}_t$ is the set of edges between the nodes. Nodes and edges can also have associated features.

Given a sequence of graphs $\{G_{t_1}, \cdots, G_{t_2}\}$, our aim is to develop a knowledge format that captures important temporal patterns and behaviors in the video and that can be used for temporal reasoning tasks, such as event prediction (binary classification for whether or not an event has occurred) or event recognition (multi-class multi-label classification into different categories of events). Our approach for temporal reasoning works as follows. First, we use self-supervised pre-training to construct a temporal knowledge graph with hierarchical events. This pre-training step is task agnostic. In the next phase, we train a graph neural network for specific downstream tasks using the hierarchy-augmented graph (see Figure 1.1).

3.2 Temporal Knowledge Graph

Temporal knowledge graph, our central representation format, is composed of many connected subgraphs, each subgraph representing an event, an important spatial-temporal pattern in input data. The nodes in events have associated time stamps and represent entities, which can be humans or objects, and the edges encode relationships between the entities. The edges can connect nodes within one event or between different events, indicating temporal relationships. Nodes and edges

can have additional associated features. These relations between events and their nodes allow our approach to identify relevant past events, including those that might possibly occur in the distant past. Furthermore, our approach allows for hierarchical events, where events at a higher level are connected to events at a low level. The aim is that events at higher levels of hierarchy capture increasingly more complex patterns and behaviors, spanning a longer time interval and involving more entities, allowing for representation and reasoning at a higher level of abstraction. A graph-based representation provides many benefits: (i) Complex dependencies in time can naturally be modeled via temporal graphical relations; (ii) The possible branching realizations of the future can then be captured as a distribution over such trees/graphs. Our hierarchical representation allows us to handle both short-term and long-term events in a scalable way by learning from repeating patterns in data and identifying temporal dependencies at multiple scales. Given a set of input graphs $\{G_{t_1}, \cdots, G_{t_2}\}$ corresponding to a sequence of video frames, we thus aim to construct a temporal knowledge graph, capturing important events.

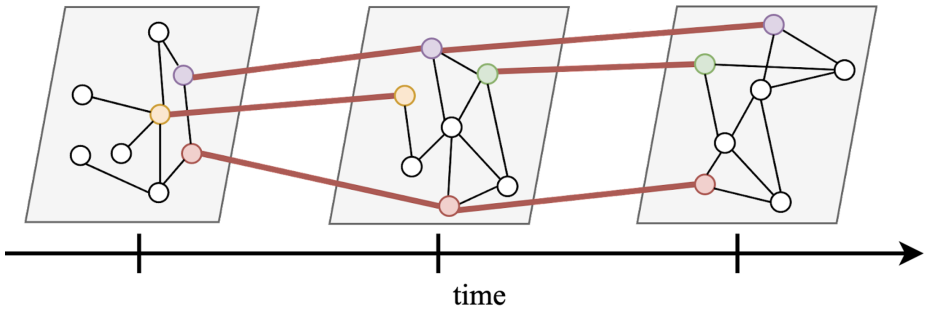

Figure 3.1: Modeling time as a spatio-temporal graph.

First, we use the input graphs to construct the base level of the temporal knowledge graph. The process of constructing this level is relatively straightforward. Each input graph forms a level 0 event in the temporal knowledge graph, preserving the nodes and edges of the input graph. We further augment the graph by adding temporal edges that connect the nodes of similar entities in consecutive input graphs (see Figure 3.1). This base level can be constructed in a streaming fashion where the output graph is updated incrementally with each new time-stamped graph G_{t_i}.

3.3 Pre-training Hierarchy-aware Event Model

A major challenge in constructing higher level events in the temporal knowledge graph is that the ground-truth does not exist for those events and therefore direct supervision is not feasible for this learning task. We address this challenge with a self-supervised approach that exploits the graph structure by using similarities and dissimilarities within the data itself. The key goal of this pre-training phase is to discover and represent the latent structure of the graph at the lower level.

In order to produce multiscale hierarchical events, we leverage a graph-pooling method based on mutual information, called vertex infomax pooling (VIPool) [22]. The general idea of graph pooling is to pick a certain proportion of nodes and connect them as in the original graph. Although downscaling is bound to lose some information, the goal is to minimize information loss such that the pooled graph can maximally represent the original graph. The VIPool method uses mutual information estimation and maximization to preserve nodes which have high mutual information with their respective neighborhoods. This technique of selecting nodes that accurately represent their local subgraphs results in the most informative subset of nodes.

Mathematically, the task is to identify a subset of nodes Ω that maximizes the information content – as defined by the criterion $C(\cdot)$ to measure the node's ability to express neighborhoods:

$$\max_{\Omega \subset \mathcal{V}} C(\Omega), \text{ such that } |\Omega| = K$$

The criterion $C(\cdot)$ has a contrastive formulation where the affinity between node and its neighborhood is minimized and that between node and an arbitrary neighborhood is maximized. We have

$$\begin{aligned} C(\Omega) \;&= \frac{1}{|\Omega|} \sum_{v \in \Omega} \log \sigma\big(T_w(\mathbf{x}_v, \mathbf{y}_{\mathcal{N}_v})\big) \\ &+ \frac{1}{|\Omega|^2} \sum_{(v,u) \in \Omega} \log \Big(1 - \sigma\big(T_w(\mathbf{x}_v, \mathbf{y}_{\mathcal{N}_u})\big)\Big) \end{aligned}$$

where $\mathcal{N}_v$ represent the neighborhood of node v – all nodes whose geodesic distance to v is below a certain threshold, w denotes the trainable parameters and T_w is a neural network that estimates the mutual information between a given node and a set of nodes. Specifically, $T_w(\mathbf{x}_v, \mathbf{y}_{\mathcal{N}_u}) = \mathcal{S}_w\left(\mathcal{E}_w(\mathbf{x}_v), \mathcal{P}_w(\mathbf{y}_{\mathcal{N}_u})\right)$ where networks $\mathcal{E}_w(\cdot)$ and $\mathcal{P}_w(\cdot)$ embed nodes and neighborhoods respectively, and $\mathcal{S}_w(\cdot, \cdot)$ is a similarity measure. These components are implemented as follows – $\mathcal{E}_w$ and $\mathcal{S}_w$ are MLPs and $\mathcal{P}_w$ aggregates nodes features via connected edges.

$C(\cdot)$ thus captures the intuition that selected nodes must have a maximal coverage of their neighborhoods alone. The optimization problem is unfolded using a greedy approach – select a vertex with the highest $C(\Omega)$ for $|\Omega| = 1$ and repeat it sequentially til the desired number of unique nodes is reached. For computational efficiency, the second term in $C(\Omega)$ is approximated using negative sampling of neighborhoods $\mathcal{N}_u$. Finally, the training loss is $\mathcal{L}_{\text{hierarchy}} = -C(\Omega)$ for learning parameters w via gradient descent.

We also achieve multi-scale feature learning using Graph Cross Networks (GXN) [22]. It comprises a pyramid structure where pooling and unpooling operations are repeated sequentially to get multiple levels of the hierarchy and loss aggregated across levels. Further, feature-crossing layers between levels enables information flow between coarser and finer representations of the graphs. Finally, deep features from all scales are combined in a multi-scale readout, providing hierarchy-aware

reasoning on downstream tasks.

3.4 Constructing Higher Order Events

Given the pre-trained event model, we construct higher level events as follows. We pass level 0 of temporal knowledge graph G through the event model to obtain a subset of nodes Ω. These nodes serve as seeds for the nodes at the higher level of G. For every node in Ω, we then add a new, higher level node, called a *supernode*, to G. These supernodes form the next hierarchical level of G. The features of the supernodes are the aggregate of their local neighborhoods at the lower level. We connect the supernodes with two types of edges: (i) hierarchy edges that connect supernodes at level l to its neighbors at level $l-1$, denoting information relations across levels, and (ii) connections between supernodes themselves denoting information relations at the higher level. This process can be iteratively repeated to construct additional hierarchical levels on top of new supernodes. The outcome is the fusion of graphs from *all* scales/levels of the hierarchy in a single representation, which prevents us from losing information by utilizing only a particular level and allows us to create a richer temporal knowledge graph structure.

3.5 Fine-tuning for Downstream Tasks

The temporal knowledge graph constructed in a hierarchical fashion can now be fine-tuned for downstream applications. We consider the task of event prediction and classification which requires temporal reasoning across various time horizons. In particular, the graph-level prediction task takes a sequence of time frames as input and classifies it into a set of event categories.

To achieve this, we train a graph neural network on the temporal knowledge graph which contains higher-order events from the time window. Note that our approach is general and can be coupled with any graph network. In our experiments, we use relational-GCN [31] as our data consists of relational edge types (such as spatial, temporal and hierarchical). Additionally, we incorporate extra node-level features from the trained hierarchy constructor, such as attention scores and one-hot vectors for identifying the level of each node. This is followed by a classifier head to predict event labels. The downstream model can then be finetuned using the binary cross-entropy loss across labels, as the standard choice for the task. Mathematically,

$$\mathcal{L}_{\text{classification}} = -\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} p_c \mathbf{y}_c \log \hat{\mathbf{y}}_c + (1-\mathbf{y}_c)\log(1-\hat{\mathbf{y}}_c)$$

where $\mathcal{C}$ denotes the set of event categories and p_c is the weight of a positive sample determined from label distribution for that event. While this is one application that can be advanced using our TimeGraphs method, we believe our approach to build a richer temporal knowledge graph is general

and other applications can be benefit as well.

3.6 Multi-task End-to-end Learning

We have introduced a two-phase approach where we first employ self-supervised pre-training of the event model and then fine-tune on the downstream classification task. Alternatively, we can also set up a multi-task problem where the event model and classifier head are trained together. For this end-to-end (E2E) approach, we adaptively weigh the two losses as

$$\mathcal{L}_{\text{e2e}} = \mathcal{L}_{\text{classification}} + \alpha \mathcal{L}_{\text{hierarchy}} \quad \text{where} \quad \alpha = 2 - \frac{\text{epoch}}{\text{total \# epochs}}$$

where hyperparamter α linearly decays with epoch number from 2 to 1 during training. This setup gives higher importance to graph-pooling loss in initial iterations and gradually increases the weight of classification loss with epochs, thereby balancing the two tasks.

Chapter 4

Experiments and Results

In this chapter, we evaluate the performance of TimeGraphs on the following temporal reasoning tasks: event prediction and action recognition. For event prediction, the task is to predict the next event in the immediate future given a sequence of frames. In action recognition, the task is to classify a sequence of frames into a pre-defined set of categories. The categories could be complete events (e.g., 'goal kick') or simpler actions (e.g., 'walking'). We start by describing our dataset curation and then proceed to experimental setup and results.

4.1 Datasets

We apply our proposed TimeGraphs approach to multiple novel datasets that feature dynamical aspects of adversarial groups. We describe these datasets along with required pre-processing steps to obtain frame-level graphs from raw videos:

4.1.1 Basketball

We use the NBA Movement dataset [1] which contains information about 632 basketball games from the $2015-2016$ NBA season. The dataset also contains tracking information extracted from videos at a rate of 25 frames per second. Specifically, it includes positions (x-y coordinates) of the ball and all 10 players on the court, height of the ball, player identities, team identities, period, game clock and shot clock (time left for a shot attempt). We utilize the play-by-play text information to infer event labels that took place during a game, such as ball passes, free throws, dunk shots and rebounds. To create a scene graph G_t at time t, we create nodes for all players and the ball and connect them via spatial edges using a proximity threshold of 10 feet.

[1] https://github.com/sealneaward/nba-movement-dataNBA Movement Data: https://github.com/sealneaward/nba-movement-data

Dataset	Nodes	Node Features	Edges	Identified Event Categories	Window Size
Basketball	{players}, ball	positions, height, team identity, game clock, shot clock	spatial; ball	scoring; shooting; substitution	4 sec (100 frames)
Football	{players}, ball	positions, velocity, team identity, player roles, tired factor	spatial; ball	scoring; shooting; ball passing; ball possession; goal kick; free kick; corner; throw in	9 sec (5 frames)
Resistance	{players}	facial action unit, speaking probability, emotions	interaction prob.; global	outcome (success/failure)	33 sec (100 frames)
MOMA	{actors}, {objects}	groundtruth actor or object class	spatial-temporal	17 activity classes; 67 subactivity classes; 52 atomic action classes	up to 175 sec (175 frames)

Table 4.1: Graph formulation for all datasets. Note that we formulate certain features (e.g. team identity and player role which are non-binary) as one-hot vectors. If a feature is not applicable to a certain node type (e.g., tired factor for ball), we set it to zero.

4.1.2 Football

We use the Google Research Football simulator [20] to generate full-game football matches. Leveraging the authors' pre-trained models, we create a reinforcement learning environment in which agents are trained to play football in an advanced, physics-based 3D simulation. Each agent or player in the simulation can be assigned different roles: a goalkeeper (who has unique rules) and other strategies such as playing defense, offense, middle field, or covering the left/right side of the field. By following common practices in football games, we choose a different mix of player roles in the simulator and create 6 teams with varying strategies – such as focusing heavily on defense or offense and in-between modes. After defining these teams and setting them to the highest proficiency level, we simulate 10 matches for each pair of teams, resulting in a total of 150 games. Each game's state, which includes player and ball positions as well as major events, is provided at a rate of 0.55 frames per second for the entire duration of the game (90 minutes). Finally, we create input graphs in a similar fashion as basketball games: nodes for player and ball, and edges for spatial closeness as well as from players' to ball node.

4.1.3 Resistance

We also evaluate on a multi-user social game called The Resistance (similar to the Mafia game). Each game involves $5-8$ players divided into deceivers and non-deceivers and lasts between $45-60$ minutes. A game consists of several rounds. By analyzing the outcome (success or failure) of every round, the goal of the game is for non-deceivers to identify the deceivers as early as possible. We use the dynamic face-to-face interaction networks dataset [1, 19] which represents the evolving interactions and discussions between participants playing the game. Networks are extracted at a rate of 3 frames per second, covering 62 games and 313 rounds in total. The network captures details about individual players (such as facial action unit [2], speaking prediction and emotions)

and pairwise features (probability that a player looks at another player). We divide every round into time windows and try to make predictions about the outcome as the round progresses. We build the input graphs using the player's attributes as node features and connect the edges by thresholding on the node pair probabilities.

4.1.4 MOMA

Lastly, we evaluate on a public dataset for video-based human activity classification, namely Multi-Object Multi-Actor Activity Parsing (MOMA) [24]. Aiming to parse human activities in hierarchy, MOMA provides 373 raw videos at activity level, annotated in 17 activity classes (e.g., dinning service), and each raw video is additional trimmed to several videos at sub-activity level (e.g., take the order, serve food, clean up the table) to make up part of a larger activity (e.g., dinning service), annotated in 67 sub-activity classes. At the most primitive level, MOMA also parses atomic action which represents an actor's movement and interaction with other entities in a trimmed video. The atomic action (e.g., A actor is speaking to B actor) is annotated in 52 classes in a multi-label setting, as an actor can perform multiple action simultaneously. Given scene graphs of video frames, we build a spatial-temporal graph by introducing temporal context edge to link the same actor/object by tracking them across all frames in the video.

4.1.5 Discussion

We believe these datasets are a good choice for evaluation because their constituent events are complex in nature, range over different durations and contain dependencies across entities at varying time scales. For every dataset, we identify relevant event categories for downstream prediction subject to data availability. Additionally, we pick a suitable window size Δt for history frames depending on the average duration of events in the dataset or manual observation of games (see Table 4.1 for further details). For all datasets, we divide full-length games into training / validation / test sets using a $7:2:1$ data split.

4.2 Baselines

For the datasets curated in this paper – namely basketball, football and Resistance games, we implement the following baselines and compare their performance with our TimeGraphs approach:

- **Counts**: This is a simple count-based statistical classifier which uses event-category distributions to predict the result based on a random sample from a uniform distribution.
- **LSTM**: Standard deep learning approaches use recurrent architectures for modeling sequence data. Here, we use LSTMs [13] that can capture long-term dependencies better. We flatten the features of all nodes in the graph at every timestep, resulting in a fixed-length feature at every

time. A sequence of these vectors is input to the recurrent layer, followed by a classification head for predicting the category of event.

- **Transformer**: Given the recent success of modeling attention in a transformer [37] model, we also use a transformer encoder on the sequence data. We add a position encoding to the flattened input (from recurrent models) to embed time. In the end, we average the embeddings of all token embeddings (here, every token corresponds to a timestep) and pass through a fully-connected classifier head.

- **Vanilla GNN**: To evaluate the benefits of a purely graph-based approach, we also consider vanilla GNNs which use only the level 0 of the temporal knowledge graphs as opposed to the hierarchical temporal knowledge graph used by TimeGraphs. We use R-GCN [31], the same GNN layer as used in our TimeGraphs method.

The sequence baselines operate on the input scene graphs from the video, while the Vanilla GNN method uses level 0 of the temporal knowledge graph. This setting allows us to address key questions of whether the temporal knowledge graphs are useful and whether hierarchical events are useful.

For the public dataset MOMA, we compare with the following state-of-the-art methods from the original paper [24]:

- **HGAP**: An action parsing network built on the hypergraph of video data. It consists of two sub-networks that unify hypergraph neural networks [8] for modeling frame-level action representation, and an action recognition model X3D [7] for video-level RGB features. Here, the hypergraph is a well-structured frame-level spatial-temporal graph for multi-object and multi-actor scenarios, and nodes extracted from each frame are embedded with image features.

- **HGAP Oracle**: With the same architecture as HGAP, 'Oracle' indicates that the ground-truth hypergraph is used as input.

- **GCN**: We use the Graph Convolutional Network (GCN) [16] which is built on a modified hypergraph by decomposing hyper edges into pair-wise edges.

4.3 Results

We evaluate different methods using standard metrics for multi-label classification tasks. Particularly, we use exact match score (proportion of samples where all labels are identified correctly), macro-averaged accuracy (average over per-label accuracies) and lastly, macro averages for F1 score, precision, and recall. Higher numbers are better for all the metrics. Our central evaluation metric is the F1 measure, so we pick the epoch and corresponding model checkpoint which has the highest F1 score on the validation set. Note that all models are trained using Adam optimizer until convergence and we use a learning rate of 0.001 along with a multi-step scheduler and a batch size of 32.

Model	F1	Prec.	Recall	EM	Acc. (%)
Dataset: Basketball					
Counts	0.203	0.202	0.205	**0.342**	**57.81**
LSTM	0.329	0.233	0.656	0.174	44.11
Transformer	0.316	0.198	**1.0**	0.0	19.89
Vanilla GNN	0.349	0.243	0.695	0.194	42.26
TimeGraphs	**0.377**†	**0.273**†	0.676†	0.278†	44.41
+ E2E	0.359	0.259	0.648	0.261	46.34†
Dataset: Football					
Counts	0.158	0.159	0.158	0.187	81.41
LSTM	0.712	0.649	0.854	0.610	81.03
Transformer	0.706	0.639	**0.901**	0.602	81.10
Vanilla GNN	0.723	0.664	0.873	0.637	81.32
TimeGraphs	0.775	0.738	0.841†	0.670	**82.03**†
+ E2E	**0.794**†	**0.762**†	0.838	**0.702**†	81.84
Dataset: Resistance Games					
Counts	0.469	0.473	0.465	0.236	50.0
LSTM	0.519	0.521	0.526	0.513	50.0
Transformer	0.482	0.521	0.517	0.494	50.0
Vanilla GNN	0.613	0.625	0.601	0.607	50.0
TimeGraphs	0.642	0.607	**0.712**†	0.519	50.0
+ E2E	**0.685**†	**0.688**†	0.684	**0.681**†	50.0

Table 4.2: Evaluation of different models on event classification task for all datasets. The best results across models are highlighted in boldface. † denotes the best results within TimeGraphs, i.e., default two-phase approach (pre-training and fine-tuning) or end-to-end (E2E) training.

4.3.1 Event Classification

We can make the following observations from the experimental results on the event classification task (see Table 4.2). The count-based baseline performs poorly which indicates task difficulty. The modeling of spatio-temporal video data as a temporal knowledge graph improves the performance as Vanilla GNN outperforms other baselines. Finally, TimeGraphs, either with a two-phase approach (TimeGraphs) or end-to-end training (E2E), achieves the highest F1 score across all datasets, both TimeGraphs variants surpassing all other methods and achieving an 8% improvement over Vanilla GNN for basketball, 7.2% for football and 4.7% for Resistance games. This performance demonstrates the effectiveness of the TimeGraphs approach for downstream tasks as well as confirms our premise that hierarchy helps in learning short- and long-term events. An unexpected outcome is that Transformer performs slightly worse than LSTM, which could be because the positional encoding is not sufficient to capture the sequential aspect of data and it fuses together the data from all the time steps. We also note that accuracy is not suitable for measuring the performance because of skewed label distributions in the data, and therefore, F1 score is a more reliable performance indicator.

4.3.2 End-to-end Training

We can also observe in Table 4.2 that end-to-end training of the event model and classifier (E2E) improves performance over the two-phase TimeGraphs in the case of the football and Resistance datasets. This is likely due to the shared structure between the two tasks, which enables the models to exploit the information from both tasks in a better way. This result also affirms our intuition that building an event hierarchy over the spatial-temporal graph is an important intermediate step, and can boost the performance of downstream tasks. However, the performance deteriorates in case of the basketball dataset, which needs further investigation. Even then, TimeGraphs with E2E training performs better than all the other baselines.

Model	**Activity Class.**	**Sub-activity Class.**	**Atomic action**	
			Class.	**Local.**
GCN	70.0	37.4	29.4	29.1
HGAP	73.9	42.5	29.2	30.3
HGAP Oracle	88.4	44.1	31.3	32.7
TimeGraphs	**95.3**	**69.5**	**35.8**	**44.1**

Table 4.3: Evaluating on activity classification, sub-activity classification and atomic action classification and localization on the MOMA dataset using mAP metric (%).

4.3.3 MOMA Dataset

Here, we compare TimeGraphs with strong baseline models using the mean average precision (mAP) metric for activity and sub-activity classification sub-tasks, and multi-label mAP metric for atomic action classification [35]. The results for MOMA are shown in Table 4.3. Compared with baseline methods, our proposed TimeGraphs using temporal knowledge graph significantly improved the HGAP Oracle model on activity, sub-activity, atomic action classification, and localization sub-tasks by 6.9%, 25.4%, 4.5% and 11.4% respectively. Here, the atomic action sub-task further includes a localization sub-task that classifies all atomic actions involved by the actor. The results show that our proposed TimeGraphs approach utilizing the temporal knowledge graph is effective for capturing fine-grained actor-centric temporal relationship at the object entity level, and also suggest its efficacy for hierarchy-aware human activity parsing tasks in a real-world setting.

Chapter 5

Analysis

In this chapter, we illustrate additional powerful capabilities of TimeGraphs using a range of experiments.

5.1 Generalization to Unseen Teams and Games

In this experiment, our goal is to investigate if a transfer learning task can be used to apply knowledge gained from a model trained on a subset of teams to predict events for unseen teams (teams not seen during training) at test time. This setting resembles a real-world scenario when a model needs to learn from a limited number of teams and then make predictions for new teams. We consider the NBA basketball dataset for this task. Instead of splitting games randomly, we now split teams – 24 for training and validation, and 6 for testing. We employ a min-cut graph algorithm to find the optimal team division such that the maximum number of games are admissible (games played between teams from training/validation set and test set are ignored). The prediction task is the same as before, i.e., identifying key event categories in the game. The results, shown in Table 5.1, show that graph-based Vanilla GNN and TimeGraphs outperform all baselines, which are sequence based. Note that although Vanilla GNN uses only level 0 of the temporal knowledge graph without any hierarchical events, it is outperforming the best sequential method by 10.7% on the F1 score. The results further demonstrate that the TimeGraphs approach is able to generalize to unseen teams and games, as the F1 score on the test set is comparable to that of the validation set. Note that the validation set contains matches between teams that appear in the training set; while the test set contains purely matches between unseen teams.

Model	F1 Score	Precision	Recall
Validation Set			
Counts	0.202	0.201	0.202
LSTM	0.317	0.210	0.642
Transformer	0.294	0.172	**1.0**
Vanilla GNN	0.351	**0.251**	0.662
TimeGraphs	**0.354**	0.247	0.709
Test Set			
Counts	0.194	0.194	0.194
LSTM	0.309	0.204	0.637
Transformer	0.291	0.172	**1.0**
Vanilla GNN	0.342	**0.243**	0.650
TimeGraphs	**0.345**	0.241	0.703

Table 5.1: Evaluating generalization to unseen teams and games at test time for the basketball dataset.

5.2 Effect of Frame Rate

We vary the stride hyperparameter over the history frames in order to study the effect of data sparsity and frame granularity. The basketball dataset has the highest number of frames per second (fps) and thereby rendering it suitable for this experiment. The trends are visualized in Figure 5.1. It can be seen that the F1 score drops gradually as frame rate decreases. While this behavior is not surprising, it is important to note that even at a stride of 20 (picking every 20^{th} frame and discarding the other 19 frames), the model is able to achieve an F1 score of 0.355 (relative to 0.377 with full information). This suggests that our model is robust to sparsity in video frames, and is able to extract useful information even when presented with a limited amount of data. The results provide valuable insights into the potential of our model in real-world applications where data sparsity and noisy/missing information are important factors.

5.3 Performance over Time

Here, we analyze how prediction accuracy changes with game time. For this task, we consider the Resistance dataset as each data sample represents an independent round within a game. Our input to the model is all frames from the start of a round to the current time. That is, we gradually increase the history window, similar to an online model where a stream of data is continuously flowing in. The goal is to predict the outcome of that round. We want to analyze how the prediction accuracy changes as time progresses and the round nears its conclusion. The result is summarized in Figure 5.2 which shows that predictions get better as the game is getting closer to the end. This is justified because there is high uncertainty in the beginning and as the model receives more

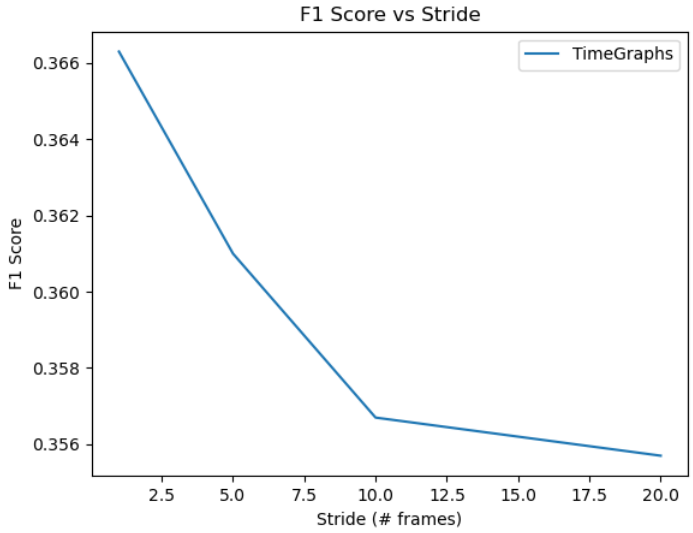

Figure 5.1: F1 score versus stride for the basketball dataset.

information, it can make decisions more accurately. Notably, the peaks and drops in the first half of the plot can be attributed to the nature of the game, where certain unseeming moments can affect the outcomes drastically, e.g., when a key player role is revealed. It is also interesting to note that even at halftime, the prediction accuracy of TimeGraphs is significantly better than a statistical prediction. This highlights the benefits of our approach for understanding how game data can be used for forecasting and for developing better strategies for playing games.

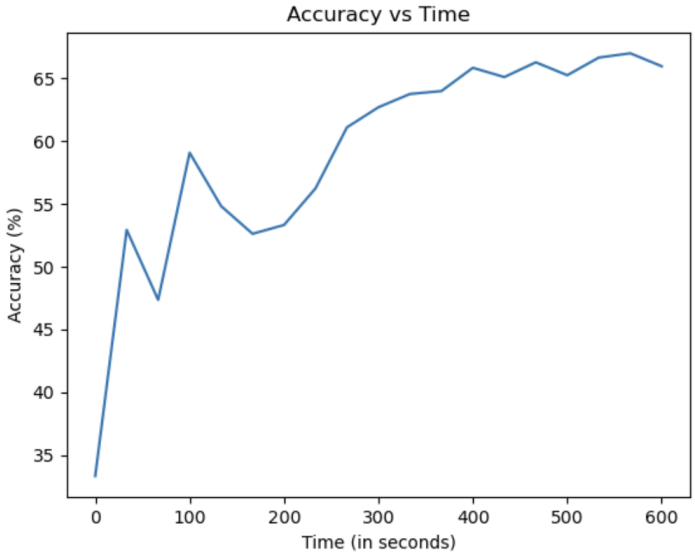

Figure 5.2: Performance accuracy as a function of time as a round of the Resistance game progresses.

5.4 Predicting the Future

An interesting experiment is to assess how far into the future we can make reasonable predictions using TimeGraphs. In particular, our input to the model is a sequence of history frames $[t - \Delta t, t]$ and the task is to predict an event that is some seconds away, i.e., at frame $t + F$. This is different from the last experiment on performance with time as now our history window is of a fixed length. In our experiments, $\Delta t = 5$ and we vary the future prediction time F from 0 frames (0 seconds) to 100 frames (180 seconds) to evaluate the model performance as the time horizon gets longer. The trends are summarized in Figure 5.3. The performance drops gradually and monotonically over time, and this is expected behavior as task difficulty increases exponentially with time. It can be observed that the results range from an F1 score of 0.551 for predicting 9 seconds ahead to 0.239 for 180 seconds ahead. It is promising to see that the TimeGraphs method outperforms the statistical count-based baseline by 240% at the shortest duration of 9 seconds and 50% even at the longest duration of 180 seconds. Notably, this experiment suggests that our TimeGraphs approach can be used to make reasonable predictions even for longer time horizons, which is a viable result for developing a real-time prediction system (where lag time is critical) for sports.

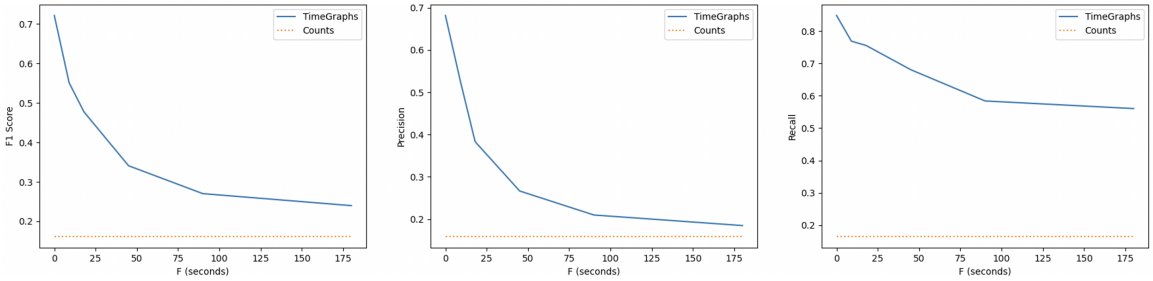

Figure 5.3: Test metric versus F, when predicting F seconds into the future on the football dataset.

Chapter 6

Conclusion

We consider the task of temporal reasoning which involves predicting and recognizing events in complex human behaviors. In this work, we develop and introduce TimeGraphs, a novel graph-based architecture for temporal reasoning. Our approach is based on temporal knowledge graphs which represent important events in a hierarchical fashion. Temporal knowledge graphs allow us to handle complex dependencies in time and facilitate reasoning across different time scales. We curated novel datasets for complex inference tasks that feature dynamic multi-agent adversarial games such as basketball, football and Resistance games. We demonstrate state-of-the-art performance of our approach compared to sequence-based and straightforward graph-based baselines. We also present several interesting ablation analyses, which highlights the generalization capabilities and zero-shot effectiveness of TimeGraphs.

As part of future work, we plan to explore model adaptation methods that enable us to learn general aspects of the game (in the form of meta-parameters) and game- or team-specific characteristics (in the form of task-parameters). Such a framework could be trained using optimization-based meta learning frameworks [9] or black-box adaptation techniques [30].

Bibliography

[1] Chongyang Bai, Srijan Kumar, Jure Leskovec, Miriam Metzger, Jay F Nunamaker Jr, and VS Subrahmanian. Predicting the visual focus of attention in multi-person discussion videos. In *IJCAI*, 2019.

[2] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 59–66. IEEE, 2018.

[3] Roger G Barker and Herbert F Wright. Midwest and its children: The psychological ecology of an american town. 1955.

[4] Aaron F Bobick. Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1257–1265, 1997.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[6] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[7] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[8] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 2019.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[10] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the conference on computer vision and pattern recognition*, 2019.

[11] Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions & temporal reasoning. In *International Conference on Learning Representations*, 2019.

[12] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proceedings of the conference on computer vision and pattern recognition*, 2017.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14024–14034, 2020.

[15] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] Yu Kong and Yun Fu. Human action recognition and prediction: A survey. *International Journal of Computer Vision*, 130(5):1366–1401, 2022.

[18] Ranjay Krishna, Ines Chami, Michael Bernstein, and Li Fei-Fei. Referring relationships. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[19] Srijan Kumar, Chongyang Bai, VS Subrahmanian, and Jure Leskovec. Deception detection in group video conversations using dynamic interaction networks. In *ICWSM*, 2021.

[20] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[21] Christopher A Kurby and Jeffrey M Zacks. Segmentation in the perception and memory of events. *Trends in cognitive sciences*, 12(2):72–79, 2008.

[22] Maosen Li, Siheng Chen, Ya Zhang, and Ivor Tsang. Graph cross networks with vertex infomax pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the conference on computer vision and pattern recognition*, pages 143–152, 2020.

[24] Zelun Luo, Wanze Xie, Siddharth Kapoor, Yiyun Liang, Michael Cooper, Juan Carlos Niebles, Ehsan Adeli, and Fei-Fei Li. Moma: Multi-object multi-actor activity parsing. *Advances in Neural Information Processing Systems*, 34, 2021.

[25] Paridhi Maheshwari, Ritwick Chaudhry, and Vishwa Vinay. Scene graph embeddings using relative similarity supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2328–2336, 2021.

[26] Albert Michotte. *The perception of causality*. Routledge, 2017.

[27] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[28] Kien Nguyen, Subarna Tripathi, Bang Du, Tanaya Guha, and Truong Q Nguyen. In defense of scene graphs for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1407–1416, 2021.

[29] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[30] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*. PMLR, 2016.

[31] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[32] Brigit Schroeder and Subarna Tripathi. Structured query-based image retrieval using scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 178–179, 2020.

[33] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the conference on computer vision and pattern recognition*, pages 12026–12035, 2019.

[34] Yoav Shoham. *Reasoning about change: time and causation from the standpoint of artificial intelligence*. Yale University, 1987.

[35] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

[36] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3716–3725, 2020.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[38] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the conference on computer vision and pattern recognition*, 2015.

[39] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*. Springer, 2016.

[40] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[41] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[42] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the conference on computer vision and pattern recognition*, 2015.

[43] Jeffrey M Zacks, Barbara Tversky, and Gowri Iyer. Perceiving, remembering, and communicating structure in events. *Journal of experimental psychology: General*, 130(1):29, 2001.

[44] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.