KTH ROYAL INSTITUTE
OF TECHNOLOGY

Doctoral Thesis in Information and Communication Technology

Enabling Enterprise Live Video Streaming with Reinforcement Learning and Graph Neural Networks

STEFANOS ANTARIS

Stockholm, Sweden 2022

Enabling Enterprise Live Video Streaming with Reinforcement Learning and Graph Neural Networks

STEFANOS ANTARIS

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Monday the 21st November 2022, at 9:00 p.m. in SAL-C, Kistagången 16, Stockholm.

Doctoral Thesis in Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden 2022

© Stefanos Antaris

ISBN 978-91-8040-398-6
TRITA-EECS-AVL-2022:69

Printed by: Universitetsservice US-AB, Sweden 2022

Abstract

Over the last decade, video has vastly become the most popular way the world consumes content. Due to the increased popularity, video has been a strategic tool for enterprises. More specifically, enterprises organize live video streaming events for both internal and external purposes in order to attract large audiences and disseminate important information. However, streaming a high- quality video internally in large multinational corporations, with thousands of employees spread around the world, is a challenging task. The main challenge is to prevent catastrophic network congestion in the enterprise network when thousand of employees attend a high-quality video event simultaneously. Given that large enterprises invest a significant amount of their annual budget on live video streaming events, it is essential to ensure that the office network will not be congested and each viewer will have high quality of experience during the event.

To address this challenge, large enterprises employ distributed live video streaming solutions to distribute high-quality video content between viewers of the same network. Such solutions rely on prior knowledge of the enterprise network topology to efficiently reduce the network bandwidth requirements during the event. Given that such knowledge is not always feasible to acquire, the distributed solutions must detect the network topology in real-time during the event. However, distributed solutions require a service to detect the network topology in the first minutes of the event, also known as the joining phase. Failing to promptly detect the enterprise network topology negatively impacts the event's performance. In particular, distributed solutions may establish connections between viewers of different offices with limited network capacity. As a result, the enterprise network will be congested, and the employees will drop the event from the beginning of the event if they experience video quality issues.

In this thesis, we investigate and propose novel machine learning models allowing the enterprise network topology service to detect the topology in real-time. In particular, we investigate the network distribution of live video streaming events caused by the distributed software solutions. In doing so, we propose several graph neural network models to detect the network topology in the first minutes of the event. Live video streaming solutions can adjust the viewers' connections to distribute high-quality video content between viewers of the

same office, avoiding the risk of network congestion. We compare our models with several baselines in real-world datasets and show that our models achieve significant improvement via empirical evaluations.

Another critical factor for the efficiency of live video streaming events is the enterprise network topology service latency. Distributed live video streaming solutions require minimum latency to infer the network topology and adjust the viewers' connections. We study the impact of the graph neural network size on the model's online inference latency and propose several knowledge distillation strategies to generate compact models. Therefore, we create models with significantly fewer parameters, reducing the online inference latency while achieving high accuracy in the network topology detection task. Compared with state-of-the-art approaches, our proposed models have several orders of magnitude fewer parameters while maintaining high accuracy.

Furthermore, we address the continuously evolving enterprise network topology problem. Modern enterprise networks frequently change their topology to manage their business needs. Therefore, distributed live video streaming solutions must capture the network topology changes and adjust their network topology detection service in real time. To tackle this problem, we propose several novel machine learning models that exploit historical events to assist the models in detecting the network topology in the first minutes of the event. We investigate the distribution of the viewers participating in the events. We propose efficient reinforcement learning and meta-learning techniques to learn the enterprise network topology for each new event. By applying meta-learning and reinforcement learning, we can generalize network topology changes and ensure that every viewer will have a high-quality experience during an event. Compared with baseline approaches, we achieved superior performance in establishing connections between viewers of the same office in the first minutes of the event. Therefore, we ensure that distributed solutions provide a high return on investment in every live video streaming event without risking any enterprise network congestion.

Sammanfattning

Under det senaste decenniet har video blivit det mest populära sättet att konsumera innehåll i världen. På grund av dess ökade popularitet har video varit ett strategiskt verktyg för företag. Närmare bestämt organiserar företag livevideoströmmande evenemang för interna och externa syften för att locka en stor publik och företagskritisk. Det är dock en utmaning att strömma videor av hög kvalitet internt i stora multinationella företag med tusentals anställda spridda över världen. Den största utmaningen är att förhindra katastrofal överbelastning av företagsnätverket när tusentals anställda deltar i ett videoevenemang av hög kvalitet. Med tanke på att stora företag investerar en betydande del av sin årliga budget i live-videoströmningsevenemang är det viktigt att säkerställa att kontorsnätverket inte blir överbelastat och att varje tittare får en högkvalitativ upplevelse under evenemanget.

För att möta denna utmaning använder stora företag distribuerade livevideoströmningslösningar för att distribuera videoinnehåll av hög kvalitet mellan tittare i samma nätverk. Sådana lösningar förlitar sig på förkunskaper om företagets nätverkstopologi för att effektivt minska kraven på bandbredd under evenemanget. Med tanke på att sådan kunskap inte alltid är tillgänglig, måste de distribuerade lösningarna detektera nätverkstopologin i realtid under evenemanget. Men distribuerade lösningar kräver en applikation för att upptäcka nätverkstopologin under de första minuterna av evenemanget, även känd som uppkopplingsfasen. Att inte snabbt upptäcka företagets nätverkstopologi påverkar evenemangets prestanda negativt. I synnerhet kan distribuerade lösningar upprätta förbindelser mellan tittare på olika kontor med begränsad nätverkskapacitet. Som ett resultat kommer företagsnätverket att bli överbelastat, och de anställda kommer att avbryta evenemanget från början av evenemanget om de upplever problem med videokvaliteten.

I den här avhandlingen undersöker och föreslår vi nya maskininlärningsmodeller som gör det möjligt för applikationen att upptäcka nätverkets topologi i realtid. I synnerhet undersöker vi nätverksdistributionen av livevideostreaminghändelser orsakade av distribuerade mjukvarulösningar. När vi gör det föreslår vi flera graph neural network models för att upptäcka nätverkstopologin under de första minuterna av evenemanget. Lösningar för direktuppspelning av video kan justera tittarnas anslutningar för att distribuera högkvalitativt videoinnehåll mellan tittare på samma kontor, vilket undviker risken för nätverksstockning. Vi jämför våra modeller med flera baslinjer i verkliga datauppsättningar och visar att våra modeller uppnår betydande förbättringar via empiriska utvärderingar.

En annan kritisk faktor för effektiviteten av live-videoströmningshändelser är om det uppstår lagg på nätverket. Distribuerade livevideoströmnings kräver minimal latens och eller lagg för att härleda nätverkstopologin och justera tit-

tarnas anslutningar. Vi studerar effekten av grafens neurala nätverksstorlek på modellens inferenslatens online och föreslår flera kunskapsdestillationsstrategier för att generera kompakta modeller. Därför skapar vi modeller med betydligt färre parametrar, vilket minskar latensen samtidigt som vi uppnår hög precision i kartläggningen av nätverkstopologin. Jämfört med state-of-the-art tillvägagångssätt har våra föreslagna modeller fundamentalt färre parametrar samtidigt som de bibehåller hög precision.

Dessutom tar vi upp problemet med ständig utveckling av företagets nätverkstopologi. Moderna företagsnätverk ändrar ofta sin topologi för att hantera sina affärsbehov. Därför måste de distribuerade lösningarna fånga nätverkstopologiförändringarna och anpassa sig i realtid. För att ta itu med detta problem föreslår vi flera nya maskininlärningsmodeller som utnyttjar historiska data för att hjälpa modellerna att upptäcka nätverkstopologin under de första minuterna av händelsen. Vi undersöker också trafikflöden hos de som deltar i eventen. Vi föreslår förstärkningsinlärning och meta-inlärningstekniker för att effektivt lära sig företagets nätverkstopologi för varje nytt event. Genom att tillämpa meta-learning and reinforcement learning kan vi generalisera förändringar i nätverkstopologi och säkerställa att varje tittare får en upplevelse av hög kvalitet under ett evenemang. Jämfört med baslinjemetoder uppnådde vi överlägsen prestanda när det gäller att upprätta kontakter mellan tittare på samma kontor under de första minuterna av evenemanget. Därför säkerställer vi att distribuerade lösningar ger en hög avkastning på investeringen i varje live-videevent utan att riskera någon överbelastning av företagets nätverk.

Acknowledgements

Pursuing a doctorate is an incredible journey characterized by excitement, fear, and incredible collaborations. During this journey, I had the opportunity to collaborate and get support from several people, for whom I am deeply thankful.

First, I would like to thank my PhD advisor, Assoc. Prof. Sarunas Girdzijauskas. I am deeply grateful for your support, patience, and guidance when needed. With your critical mindset, knowledge, and motivation, you made it possible for me to finish this journey.

Second, I would like to acknowledge my appreciation and gratitude to my advisor and friend, Assoc. Prof. Dimitrios Rafailidis and my dearest friend Assist. Prof. Eleni Constantinou. Your broad knowledge and expertise were crucial to achieve my goals and providing novel solutions. I could have never imagined having better friends and advisors than you.

I would also like to acknowledge the support from Hive Streaming AB, especially Johan Ljungberg, Niklas Hagen, Riccardo Reale, and Despina Stamkou. I am deeply thankful for allowing me to achieve my goals.

They say, "behind every great man; there is a great woman." In my case, I consider myself lucky to have my wife, Anastasia Michailidou, and my son Philippos-Vaios Antaris in my life. I am incredibly thankful for your continuous support and motivation. More than this, I am sincerely grateful for the courage that I got to overcome my disappointments during this journey.

Last but not least, big thanks to my mother, Ermioni Antari, my father Vaios Antaris, and my sister Vasiliki Antari. Even though you are in Greece, your immense love and support were precious.

Contents

Chapter 1

Introduction

Video technology has become the main communication tool in our everyday life. Nowadays, video content accounts for almost 82% of the internet traffic [1], with more than 500 hours of video being generated and uploaded to social media, such as Youtube[1], Meta[2], etc [2]. Since 2019, when Coronavirus disease 2019 (COVID-19)[3] forced almost everyone to stay and work from home, video technology has changed the way people interact with each other. Tools such as Skype[4], Whatsapp[5], Facebook Messenger[6], provide an essential medium of remote communication similar to meeting in-person. Moreover, recent advances of video technology contributed on improving the way that people gain new knowledge. Massive open online courses provide a flexible way of learning new skills, while most universities nowadays support a hybrid way of teaching with lectures both in physical and online settings. Additionally, video technology has assisted in the way that people have access to entertainment. For example, users attend live video streaming events, through Facebook Live[7], Youtube Live[8], to experience live events, such as football games, and concerts from anywhere around the world.

The recent advances of video technology have also impacted significantly the communication between employees in large enterprises. Organizations have widely adopted the video technology as their mainstay communication tool of their digital transformation. Tools such as Microsoft Teams[9], Zoom[10], and so

[1] https://youtube.com
[2] https://about.facebook.com/meta/
[3] https://en.wikipedia.org/wiki/COVID-19
[4] https://www.skype.com/en/
[5] https://www.whatsapp.com
[6] https://www.messenger.com
[7] https://www.facebook.com/formedia/tools/facebook-live
[8] https://www.youtube.com/live
[9] https://www.microsoft.com/en/microsoft-teams/group-chat-software
[10] https://zoom.us/

on, allow the enterprises to exploit the video technology in two main ways: i) online video meetings, and ii) live video streaming events. Enterprises organize online video meetings mainly to educate their employees, empower and optimize the collaboration between employees of different countries. Moreover, large enterprises such as Fortune-500[11] companies with thousands of employees spread around the world, organize town hall live video streaming events to announce essential information to their employees. Given that large enterprises maintain offices around the world, organizing a town hall event through a live video streaming event is more efficient than all employees attending the town hall in-person. Therefore, large enterprises can improve the employee engagement and empowerment [3].

A live video streaming event is different from the online video meeting. The main difference comes from the scalability limitations of the two video services. More specifically, during an online video meeting, each participant establishes a direct connection with all the other participants, thus creating an all-in-all topology. Such topology requires significant network bandwidth to deliver a high-quality video content. To avoid any catastrophic congestion of the enterprise network, while ensuring high quality experience for all participants, video communication tools limit the meeting calls to maximum 1000 attendees[12]. To overcome the number of participants limitation, live video streaming solutions adopt a different video distribution topology. Instead of establishing an all-in-all topology, the presenter is streaming the video content directly to a Content Delivery Network (CDN), and each participant downloads the video stream directly from the CDN. Therefore, live video streaming solutions establish an one-to-many distribution topology, allowing the presenter to communicate to hundred of thousands of viewers.

Although a live video streaming event has no attendees limitation, streaming a video event in a large enterprise network still remains a challenge. The main challenge stems from the network capacity limitation of each office. Each office in large enterprises hosts thousands of employees, especially in headquarters. Downloading the video stream directly from the CDN for every viewer simultaneously is problematic, as the amount of data that need to be transferred congests the enterprise network, preventing the participants to attend the meeting. Large enterprises invest a significant amount of their annual budget on live video streaming events so as to improve the employee engagement and empowerment [3]. Therefore, such organizations expect consistent high-quality performance of their live video streaming events to ensure high re- turn of investment of each event. To alleviate the network congestion problem, large enterprises apply distributed solutions provided by companies, such as Hive

[11] https://fortune.com/fortune500/

[12] https://docs.microsoft.com/en-us/microsoftteams/troubleshoot/teams-conferencing/teams-meetings-capped-at-250

Streaming AB[13], to reduce the network requirements of transferring the video to each office. Such solutions employ distributed architectures to establish connections between viewers of the same office and exploit the office's internal high bandwidth network. More specifically, during a live video streaming event only a subset of each office's viewers fetch the video directly from the CDN. Once the video is transferred to the office, each viewer creates several connections with the other office viewers and distribute the video content without any risk of network congestion.

To establish a connection between viewers of the same office, distributed video streaming solutions require the prior knowledge of the company's network topology. Streaming a video content without such knowledge limits the distributed solutions to establish connections between viewers of the same office. As a consequence, the participants drop the event, achieving low engagement and significantly reducing the event's return of investment. Moreover, the network congestion caused during the event may disrupt other critical business services. Thus, it is essential to derive and exploit the company's network topology during a live video streaming event. However, such information is difficult to acquire. Large enterprises are reluctant to provide their network topology due to security reasons. The recent data protection regulations (GDPR, CCPA) prevent distributed video streaming providers from collecting network characteristics, such as public and private internet protocol (IP) addresses. Moreover, network administrators frequently change the enterprise network, so as to meet the organisation's needs. Especially since the Covid-19 crisis, enterprise networks are changing regularly, due to the regulations that forced employees to work from home. Distributed solutions have no access to the enterprise network topology, posing significant challenges during the video streaming event. Therefore, it is essential to infer in real-time the enterprise network topology based on the existing viewers interactions, so as to establish connections between viewers that are located in the same office.

Additionally, each live video streaming event corresponds to a significantly evolving distribution topology. As illustrated in Figure 1.1, generated by all the live video streaming events that Hive Streaming supported during September 2022, live video streaming events can be characterized by three main load phases: i) the joining phase, when viewers start joining the event, ii) the attending phase, when the majority of the viewers remain and attend the event, and the iii) the leaving phase, when the event ends and viewers drop their connection. Each viewer establishes a limited number of connections. Given that we usually have no prior information about the enterprise network topology, viewers at the joining phase of the event may establish connections with other viewers from different offices. To identify the high-bandwidth internal connections with the viewers of the same office, each viewer periodically adapts the

[13] `https://www.hivestreaming.com/`

existing connections. Therefore, the distribution topology changes significantly at the joining phase. Detecting the network topology at the attending phase negatively impacts the performance of the event. Viewers will experience video issues, such as buffering, low video quality, and so on, and might lose their interest and drop the event [4]. Accounting to the fact that enterprises invest a significant amount of their annual budget on live video streaming events, they expect each event to achieve high return of investment. This means that distributed video solutions need to detect the network topology with the limited information of the connections established at the joining phase of the event, while the video distribution topology changes significantly at every minute of the event. To address the above problem, enterprises provide centralized services to detect the network topology in real time. Such services exploit graph analysis methods to detect viewers in the same office. Existing approaches apply gossiping [5] and minimum spanning tree algorithms [6] to detect viewers with high bandwidth connections. However, such methods require a significant amount of time to converge. Other approaches employ heuristics based on the Delaunay triangulation and several geographically distributed servers [7]. Such techniques compute the proximity between viewers by triangulating the viewers' network telemetry data from different servers. However, enterprise networks exploit several network security layers, such as virtual private networks (VPNs), Zscaler, and so on, which significantly affect the viewers' network performance. Therefore, such heuristics fail to detect the enterprise network topology properly.

Machine learning, especially deep learning, has shown remarkable performance in various domains, such as computer vision, natural language processing, graph processing, and so on. Detecting the network topology based on the limited information of the viewers' connections through machine learning has been investigated in recent years. More specifically, live video streaming events can be modeled as graphs, and the machine learning task is to identify the high-bandwidth connections between viewers. However, existing solutions that employ machine learning on graphs typically perform link prediction on graphs that do not evolve significantly over a short period of time. The connections evolve considerably over time, and the network topology changes between events. Thus, current approaches are not designed to perform on graphs generated from live video streaming events. Additionally, existing methods perform link prediction on single graphs, ignoring the information from graphs generated from historical live video streaming events. However, enterprise network topologies change frequently. Missing the information from historical events prevents the existing approaches to detecting the network topology at the first minutes of the event.

To evaluate the efficiency of our proposed methods in real-world applications, we collected data from 100 live video streaming events on 50 Fortune 500

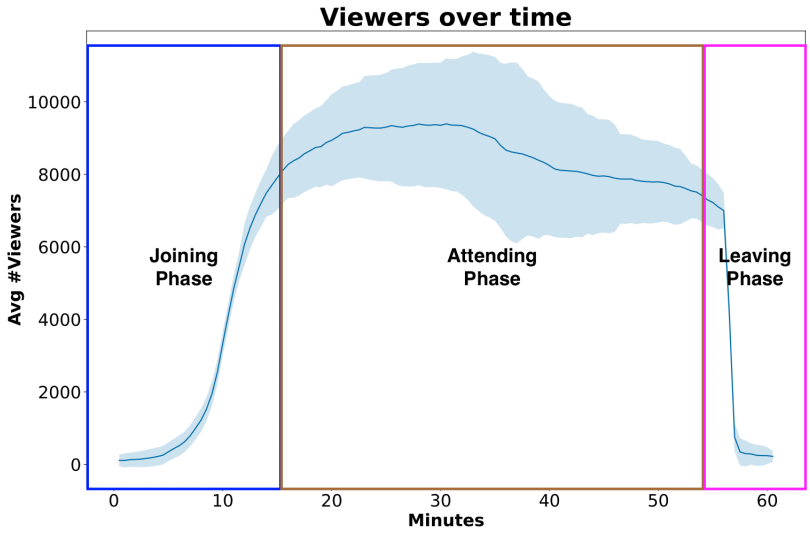

Figure 1.1: Live video streaming load phases. This is an average of the viewers' participation over all the live video streaming events distributed through the Hive Streaming AB solution during the September 2022.

enterprises. The viewers of each event exploit the Hive Streaming AB solution to distribute the video content. In total, viewers are distributed in 200 cities and 80 countries worldwide. Each event consists of a minimum $4,000$ viewers, while the biggest event contains $100,000$ viewers. The selected live video streaming events were from December 2019 to October 2021. The focus of the chosen dataset is threefold: i) to capture the enterprise network evolution on a single enterprise, ii) to identify the different enterprise network topologies between organizations, and iii) to investigate the scalability requirements during an event for a real-time network topology detection service.

This thesis aims at developing machine learning models to detect high-bandwidth viewers' connections. Our models are scalable to thousands of viewers and detect the network topology at the joining phase. This thesis enables distributed video streaming solutions to provide quality video content in the enterprise network without the risk of network congestion. The main statement of this thesis is:

Enterprise live video streaming solutions can detect the high-bandwidth connections on a significantly changing network through Graph Neural Networks. Fast adaptation to new networks with low structural similarity can be achieved by exploiting meta-learning and reinforcement learning. Finally, novel knowledge distillation strategies can reduce the graph neural networks model size

while maintaining high prediction accuracy. This enables enterprise live video streaming solutions to distribute high-quality events on enterprise networks.

1.1 Research Motivation

Our research motivation is threefold; predicting the high-bandwidth viewer connections in a significantly evolving graph, detecting the continuously changing network topology, and reducing the online inference latency of the models. Several machine learning models have been proposed to solve graph link prediction [8]–[11]. The central area of concern is to detect the high-bandwidth viewer connections based on the limited information on the viewer connections established at the joining phase of the event. At the joining step, viewers adjust their connections frequently to identify the high-bandwidth connections. Thus, most graph edges at the joining step contain information for the low-bandwidth connections between viewers. However, existing machine learning approaches on graphs are designed to perform link prediction on graph structures where the edges have positive information between nodes. Thus, these methods not only require high-bandwidth connections to detect the enterprise network topology but also require a significant amount of time to predict the high-bandwidth connections. Additionally, existing machine learning approaches on graphs assume that the connection between nodes remains stable over time. Specifically, the edge weight between two nodes does not change. Either the edge weight is the same until the last graph snapshot, or the edge is removed alongside the weight. In contrast, the office network load might affect the bandwidth between two viewers. In particular, two viewers located in different offices might have high bandwidth to distribute the video content at the joining phase because they are the only viewers attending the event at this specific time. However, their bandwidth might be considerably reduced as more viewers join the event and try to detect the high-bandwidth connections.

Most existing research on machine learning on graphs targets link prediction on domains where graph structures remain primarily stable over time. For example, most of the viewers in social networks remain in the same geographical region each day, online users have similar interests over time, and their interests are adjusted slowly. These solutions aim to detect the link between two nodes that express their interests for this specific time while the graph structure remains mostly the same. However, enterprise network topologies change frequently between events. Especially in companies with hybrid working policies, employees might attend each event from different locations, such as their home, office, or cafeteria. Therefore, the network topology is different for every live video streaming event. Existing machine learning approaches on graphs are not designed to capture the evolution of the network topology in their link prediction task, thus focusing on a single live video streaming event.

In addition to these challenges, machine learning models on evolving graphs require many parameters to train to achieve high prediction accuracy. Despite the high prediction accuracy, such models present high online inference latency to predict the edge importance between two nodes in the graph. This increased online inference latency poses significant scalability problems for services that want to scale to millions or billions of end-users. Mainly, live video streaming solutions rely on the network topology detection service to adjust the viewers' connections and distribute the video content through high-bandwidth links. Such solutions require sub-second responses for hundreds of thousands of viewers simultaneously. Thus, machine learning models on graphs need to reduce the online inference latency to satisfy the service's requirements. Yet, it is challenging to reduce the online inference latency of the machine learning model without deteriorating the prediction accuracy.

In this thesis, we investigate and propose machine learning models to address three main challenges of live video streaming events, as shown in Figure 1.2:

- **Network Congestion**: enterprise networks have limited network capacity. Establishing connections between viewers from different offices may result in network congestion. To address the network congestion challenge, we model the live streaming events as temporal networks that change over a short period of time. In **Paper A** [12], we propose a novel graph neural network architecture to achieve high link prediction on graphs that change significantly over time. Our proposed solution allows live video streaming solutions to ensure high quality of viewer experience, without any risk on network congestion.

- **Continuously Evolving Network**: modern enterprise networks frequently change, where employees adapt to a hybrid workplace. Therefore, distributed video streaming solutions cannot exploit distribution policies that apply to all events, as each event's underlying network topology differs. This prevents distributed solutions from predicting the network topology at the first minutes of the event, thus connecting viewers between different offices. In this thesis, we model the selection of the viewer's connections as a Markov Decision Process and solve it with Reinforcement Learning. To combat the problem of fast adaptation on continuously evolving network topology, we exploit meta-learning in **Paper B** [13] and gradient boosting in **Paper C** [14] techniques. This allows our proposed models to achieve high prediction accuracy and generalizing on evolving network topologies.

- **High Online Inference Latency**: distributed live video streaming solutions require low latency during an event to identify the connection between viewers of the same office efficiently. On the contrary, machine

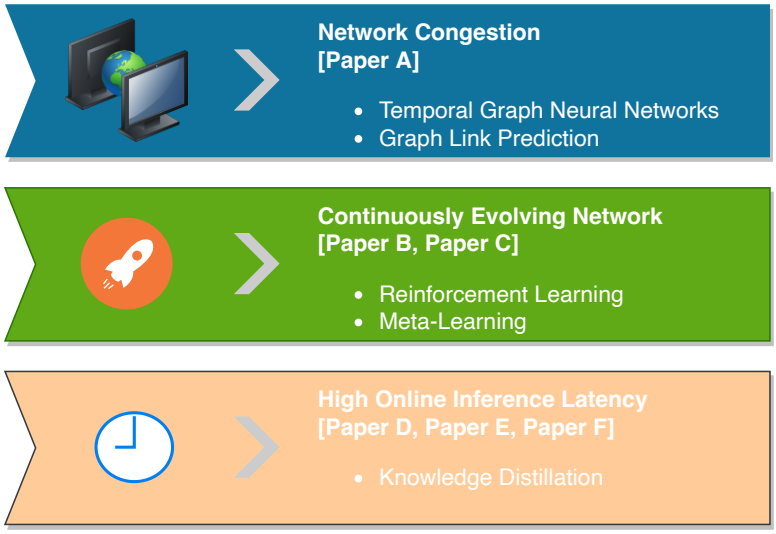

Figure 1.2: Challenges and Proposed Solutions

learning models require many parameters to train to achieve high accuracy. Therefore, existing models are unsuitable for distributed live video streaming solutions due to their high online inference latency, resulting in viewers erroneously establishing connections with other viewers from different offices. On the contrary, machine learning models require many parameters to train to achieve high accuracy. To address the high online inference latency problem, we employ knowledge distillation strategies to generate a compact model without any loss in prediction accuracy. In **Paper D** [15], **Paper E** [16], and **Paper F** [17], we propose several knowledge distillation strategies that extract knowledge from several outputs of the big model (teacher), such as output features and model responses. We apply the proposed knowledge distillation strategies on evolving graph neural networks, allowing distributed video streaming solutions to achieve low latency and high viewers' scalability.

1.2 Research Objectives

Live video streaming events in enterprise networks are complex, given that detecting the high-bandwidth viewer connections is difficult in a significantly changing distribution network during an event. Accordingly, solutions must be investigated that adapt to different enterprise network topologies. Specifically,

the provided machine learning solution needs to support multiple enterprise networks. Such approaches should adapt to continuously evolving network topologies of the same enterprise. Furthermore, reducing the online inference latency of the proposed machine learning models is challenging to allow the network topology detection system to scale to a vast number of concurrent viewers.

Figure 1.3 illustrates our proposed architecture for network topology detection in enterprise live video streaming events. The architecture consists of three main components: i) the enterprise network topology, ii) the video distribution topology, and iii) the network topology detection service. Large enterprises maintain several offices spread around the world. The enterprise network topology is responsible for efficiently connecting the different offices. On top of the enterprise network topology is the video distribution topology. Viewers in different offices leverage the enterprise network topology during a live video streaming event to establish connections to distribute the video stream. Given that connections between viewers of the same office have higher network bandwidth than connections of different offices, we propose a centralized network topology detection service. The centralized network topology detection service predicts high-bandwidth connections and instructs viewers to adjust their connections in real time during an event.

The main goal of this work is to investigate machine learning models to assist the network topology detection service for distributed live video streaming solutions. Our vision is to eliminate prior enterprise network topology knowledge requirements during the event. We aim to exploit the information from previous video distribution topologies to learn models that are adaptive to network topology changes. Accounting for the fact that each enterprise has a different network topology, our objective is to design models that capture similarities between different network topologies and provide quick adaptation to unique topologies. Moreover, we consider that enterprise networks change between events. Thus, our goal is to design models that will assist the network topology evolution detection component in capturing both the differences between enterprise networks and the evolution of enterprise networks over time. Additionally, a video distribution topology during a live video streaming event significantly changes as viewers adapt their connections to establish high-bandwidth connections. Therefore, we focus on designing machine learning models to achieve high prediction accuracy on high-bandwidth connections in a significantly changing network. As shown in Figure 1.3, the real-time video distribution topology detection component exploits the network topology evolution detection component's information and the real-time data the viewers reported during an event. Yet, network topology detection service needs to scale to a significant amount of concurrent viewers. This means that live video streaming solutions require sub-second inference latency to adjust the viewers' connections in real time. Therefore, we aim to provide mechanisms to reduce the online infer-

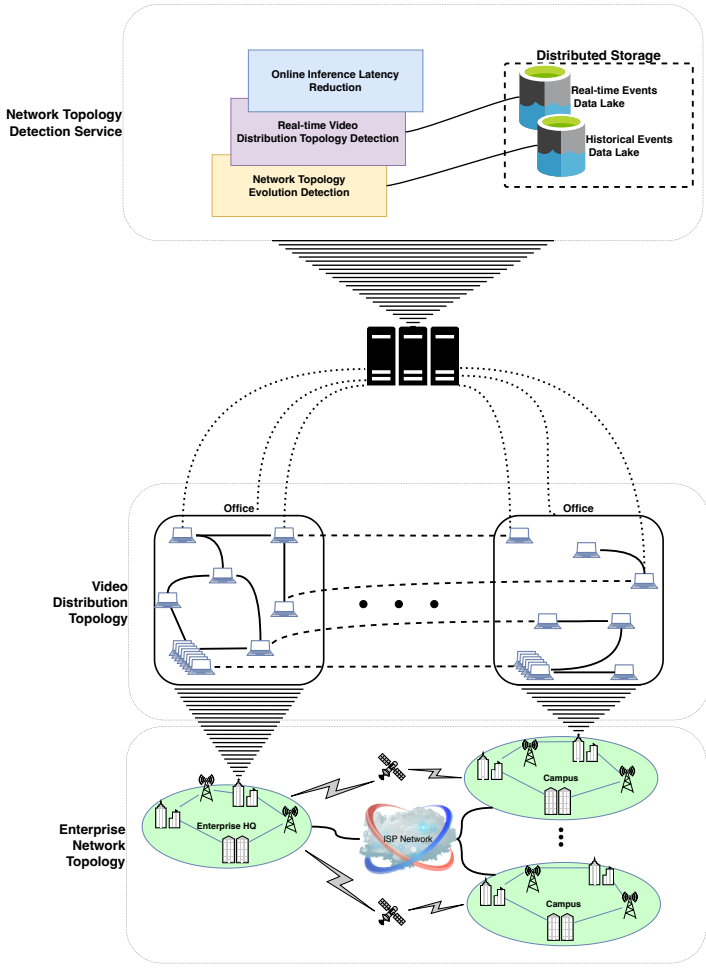

Figure 1.3: The proposed live video streaming solution architecture. The network topology detection service exploits the data from the distributed storage and contains the proposed machine learning models to adjust the distribution topology during the event. The real-time video distribution topology detection exploits the network topology extracted by the network topology evolution detection and the real-time events data and adjust the viewers' connections in real-time. Finally, the online inference latency reduction is responsible to create compact model with low online inference latency.

ence latency on the models applied to the real-time video distribution topology detection models while maintaining high prediction accuracy.

To achieve our vision, we designed the models provided in this thesis based on the following core requirements:

- **Adaptive methods.** Each enterprise network is unique due to the number of offices, employees, and network connectivity between offices. Moreover, enterprise networks evolve. Therefore, we provide adaptive methods to adapt to previously unseen network topologies.
- **Real-time detection.** Viewers in a live video streaming event continuously adapt their connections. This can affect the event's performance since connections between viewers of different offices negatively impact the office network capacity. Therefore, we provide real-time detection methods to predict the high-bandwidth connections.
- **Low online inference latency.** The real-time detection methods require sub-second inference latency to allow the service to scale to a massive number of concurrent viewers. Our research designs a mechanism to create compact models without any prediction accuracy loss.

1.3 Thesis Contributions

This thesis makes the following contributions in the fields of graph neural networks, reinforcement learning and knowledge distillation:

- In **Paper A** [12],we model the video distribution topology as an evolving graph and explore the performance of existing graph neural network approaches on highly evolving graphs, that is, graphs that change significantly over time. We further propose a graph neural network model that exploits a self-attention mechanism to capture the graph evolution on the convolutional layers of the model. This allows the model to propagate the information between viewers with high bandwidth connections while filtering out the connections between viewers from different offices. In doing so, our model achieves high prediction accuracy on graphs that evolve significantly over time.
- We propose a novel meta-reinforcement learning model to leverage historical live video streaming events. We formulate the viewers' connection process as an MDP and learn the optimal policy through reinforcement learning. Considering that every event attracts different employees, while only a small subset of the event's viewers participated in previous events, we exploit meta-learning in **Paper B** [13] and gradient boosting in **Paper**

C [14] to provide fast adaptation to new events. Moreover, we propose a novel graph signature buffer to calculate the structural similarities of several streaming events and adopt the learning of the optimal global policy accordingly.

- To combat the problem of high online inference latency of large models, we propose several novel knowledge distillation strategies in **Paper D** [15], **Paper E** [16], and **Paper F** [17]. We investigate the knowledge distillation strategies that exploit the large model to train a compact model which presents low online inference latency, while achieving high accuracy. Therefore, we design several distillation loss functions to distill knowledge from the large model and propose novel strategies for dynamic graph neural networks.

1.4 List of Publications

1. Stefanos Antaris, Dimitrios Rafailidis. "VStreamDRLS: Dynamic Graph Representation Learning with Self-Attention for Enterprise Distributed Video Streaming Solutions". ASONAM. 2020.

   Contribution: The author of this thesis designed and implemented the proposed graph representation learning method presented in this paper, performed the experimental analysis, wrote majority of the text of the paper, and designed the figures.

2. Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas. "EGAD: Evolving Graph Representation Learning with Self-Attention and Knowledge Distillation for Live Video Streaming Events". IEEE BigData. 2020.

   Contribution: The author of this thesis designed and implemented the proposed knowledge distillaton learning method presented in this paper, performed the experimental analysis, wrote majority of the text of the paper, and designed the figures.

3. Stefanos Antaris, Dimitrios Rafailidis. "Distill2Vec: Dynamic Graph Representation Learning with Knowledge Distillation". ASONAM. 2020.

   Contribution: The author of this thesis designed and implemented the proposed knowledge distillaton learning method presented in this paper, performed the experimental analysis, wrote majority of the text of the paper, and designed the figures.

4. Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas. "Meta-reinforcement learning via buffering graph signatures for live video streaming events". ASONAM, 2021.

Contribution: The author of this thesis designed and implemented the proposed meta-reinforcement learning method presented in this paper, performed the experimental analysis, wrote majority of the text of the paper, and designed the figures.

5. Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas. "A Deep Graph Reinforcement Learning Model for Improving User Experience in Live Video Streaming". IEEE BigData. 2021.

   Contribution: The author of this thesis designed and implemented the proposed graph reinforcement learning method presented in this paper, performed the experimental analysis, wrote majority of the text of the paper, and designed the figures.

6. Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas. "Knowledge distillation on neural networks for evolving graphs". Social Network Analysis and Mining, 11(1):100, 2021.

   Contribution: The author of this thesis designed and implemented the proposed knowledge distillation strategies presented in this paper, performed the experimental analysis, wrote majority of the text of the paper, and designed the figures.

1.5 Outline

This thesis is organized as follows, in Chapter 2 we detail the background related to the proposed models of this thesis. In Chapter 3 we provide a summary of the papers included in the thesis and in Chapter 4 we conclude the thesis. The complete publications supported in this thesis are presented afterwards.

Chapter 2

Background

In this chapter, we briefly describe the machine learning techniques employed to design our proposed solutions. First, we detail the graph neural network approaches. Then, we discuss the reinforcement learning and meta-learning techniques that allow machine learning models to adapt quickly to new environments. Finally, we discuss the existing knowledge distillation strategies to create compact models with low online inference latency.

2.1 Graph Neural Networks

Deep learning models have shown remarkable performance on various machine learning tasks, such as email spam filtering [18], object detection [19], [20], machine translation [21], [22], recommender systems [23]–[25], etc. Recently, graph neural networks have been proposed to capture the dependencies on the data generated by graphs [26], [27]. Graph neural networks aim to capture specific graph statistics in low-dimensional embeddings. More specifically, a graph neural network consists of two main components: i) the encoder and ii) the decoder [27]. The encoder part is responsible for encoding each node to a low-dimensional node embedding based on a function f. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of connections, a graph neural network generates a low dimensional node embedding $\mathbf{z}_v \in \mathcal{Z}$ for each node $v \in \mathcal{V}$, as follows:

$$\mathbf{z}_v = f(v) \tag{2.1}$$

The generated node embeddings $\mathcal{Z}$ should capture the desired graph statistics. Therefore, the decoder is responsible for reconstructing the graph statistics based on the generated node embeddings. In particular, given two embeddings $\mathbf{z}_u$ and $\mathbf{z}_v$ of the nodes u and v, the decoder might attempt to identify if there

is a similarity between the two nodes based on the graph property that we try to capture, as follows:

$$S[u, v] = f'(\mathbf{z}_u, \mathbf{z}_v) \tag{2.2}$$

where f' is the decoding function and S denotes the similarity between the two nodes for the specific property we aim to capture. To optimize the encoding of the graph to low-dimensional embeddings, graph neural networks are trained to minimize an empirical reconstruction loss $\mathcal{L}$ over a set of node pairs, as follows:

$$\mathcal{L} = \min_{(u,v)\in\mathcal{E}} l\big(f'(f(u), f(v)), S[u, v]\big) \tag{2.3}$$

where l is the loss function measuring the error between the reconstruction and the similarity of the two nodes.

Early approaches to graph neural networks aim to learn node embeddings on static graphs. Such static methods exploit many techniques to learn node representations, such as matrix factorization [28], [29], Random Walks [9]–[11] and Deep AutoEncoders [30]. More recently, convolutional neural networks have been adapted to capture graph data [31]. Similarly, attention mechanisms has been employed on graph structures [32], while Adversarial Learning [33] and self-supervised learning [34] has shown remarkable performance. However, real-world applications are dynamic by nature. Thus the graphs are evolving. Such static approaches fail to capture the evolution of the graph.

Recent approaches to evolving graphs aim to learn the temporal dynamics over consecutive graph snapshots. For example, Block-Coordinate Gradient Descent (BCGD) introduces a latent temporal space model extracted by non-negative matrix factorization [35]. Dynamic Joint Variational Graph AutoEncoder (DynVGAE) shares weights between consecutive GCNs and models the graph evolution by formulating a joint loss function [36]. Recently, a set of approaches tries to summarize the graph evolution based on recurrent architectures such as Gated Recurrent Units (GRUs) between GCNs. For instance, Graph Convolutional Recurrent Network (GCRN) exploits GCNs to compute the node representations and then provides the generated representations to Long-Short Term Memory (LSTM) networks to learn the graph dynamics [37]. Dyngraph2vec stacks several LSTMs in the AutoEncoder architecture to learn the long-term dependencies of the dynamic graph [38]. Evolving Graph Convolutional Network (EvolveGCN) employs GRUs to store the importance of the node features in the hidden states and learn the weights of each GCN layer [39]. Dynamic Self-Attention Network (DySAT) captures the graph evolution by applying a self-attention mechanism to focus on the important node features and edges that are preserved over consecutive graph snapshots [40].

Despite the significant performance of the existing graph neural networks on various machine learning tasks, such approaches fail to capture the evolution

of the graphs generated by the live video streaming events. More specifically, current graph neural network approaches are designed to capture the evolution of a graph that does not change significantly between consecutive snapshots. Instead, most of the graph structure generated by a live video streaming event changes completely, especially at the joining phase of the event.

Graph neural networks play an essential role in our proposed solutions. Specifically, in **Paper A** [12], **Paper D** [15], **Paper E** [16],**Paper F** [17] we employ the self-attention mechanism to generate node embeddings that capture the evolution of the graph. After that, the similarity between the embeddings is used to determine the weight between the two nodes. Additionally, graph neural networks are applied in **Paper B** [13] and **Paper C** [14] to compute the viewer's state over time. A centralized server adjusts the viewer's established connections based on the viewer's state.

2.2 Reinforcement Learning

Reinforcement learning is a machine learning field that enables an agent to learn an environment through interaction [41]. More specifically, an agent observes the environment, called the state, and based on the current state it takes an action. Given the state of the agent and the selected action, it receives a reward from the environment and transitions to a new state. More formally, a reinforcement learning problem is typically formulated as a Markov Decision Process (MDP), as $(\mathcal{S}, \mathcal{A}, R, \mathcal{P}, \gamma)$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the set of all available actions of the state in all states, R is the reward function, $\mathcal{P}$ is the transition probability between states given an action, and γ is the discount factor. The objective of the agent is to find an optimal policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where θ are the parameters of the policy π.

A policy π_θ describes the decision-making process of the agent. In the simplest case, the policy for each state refers to an action that the agent should perform in that state. This type of strategy is called deterministic policy. Each state is assigned an action, for example for state $s_1 : \pi_\theta(s_1) = a_1$. In general, a policy sets probabilities for every action in every state. Therefore, the policy represents a probability distribution for every state over all possible actions. Such a policy is called a stochastic policy. In a stochastic policy, several actions can be selected, whereby the actions each have a probability of non-zero, and the sum of all actions is 1. Thus, it can be said that the behavior of an agent can be described by a policy, which assigns states to a probability distribution over actions. The policy only depends on the current state and not the time or the previous states.

The value function represents the value for the agent to be in a particular state. The state value function describes the expected return O_t from a given

state. . In general, a state value function is defined concerning a specific policy since the expected return depends on the policy:

$$v_\pi(s) = \mathbb{E}[O_t|S_t = s] \tag{2.4}$$

where the expected return O_t is defined as follows:

$$O_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.5}$$

The index π indicates the dependency on the policy. Furthermore an action-value function can be defined. The action-value of a state is the expected return if the agent chooses an action according to a policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[O_t|S_t = s, A_t = a] \tag{2.6}$$

Value functions are critical to Reinforcement Learning. They allow an agent to query the quality of his current situation rather than waiting for the long-term result. This has a dual benefit. First, the return is not immediately available, and second, the return can be random due to the stochasticity of the policy as well as the dynamics of the environment. The value function summarizes all future possibilities by averaging the returns. Thus, the value function assesses the quality of different policies.

A fundamental property of value functions used throughout reinforcement learning is that they satisfy recursive relationships. For each policy and state s, the following consistency condition applies between the value of s and the value of its possible subsequent states:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.7}$$

This equation is also called the Bellman equation. For the value function, the Bellman equation defines a relation of the value of State s and its following State s'. The Bellman equation is also used for the Action-Value function. Accordingly, the Action-Value can be calculated from the following state:

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_a \pi(a|s) q_\pi(s', a')] \tag{2.8}$$

In the Bellman equations, the structure of the Markov Decision Process formulation is used to reduce this infinite sum to a system of linear equations. The exact state values can be determined by directly solving the equation.

To solve a task or a problem in reinforcement means to find a policy that will have a great reward in the long run. For finite Markov Decision Processes, an optimal policy can be precisely defined in the following way. Value Functions

define a partial order over different policies. For example, a policy π is better or at least as good as a policy π' if the expected return across all states is greater than or equal to that of π'. In other words, $\pi \geq \pi'$ is better for and only if $v_\pi \geq v_{pi'}$ is better for all states. This is an optimal policy pi_*

Although there may be several optimal policies, they all share the same state value function, which is called the optimal state value function and is defined as follows:

$$v_*(s) = \max_\pi v_\pi(s) \tag{2.9}$$

Optimal policies also share the same optimal action-value function:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{2.10}$$

Because v_* is a value function for a policy, it must meet the condition of uniformity of the Bellman equation. Since it is the optimal value function, the consistency condition of v_* can be written in a special form without reference to a specific policy. This Bellman equation for v_* is also called the Optimal Bellman Equation and can also be written down for the optimal action-value function.

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \tag{2.11}$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')] \tag{2.12}$$

Once v_*, it is straightforward to derive an optimal policy. There will be one or more actions for each state s where a maximum in the optimal Bellman equation is reached. Any policy that assigns a probability greater than zero to only these actions is optimal. Using v_*, the optimal expected long-term return is converted into a quantity immediately available for each state. A one-step predictive search thus yields the optimal long-term actions.

With q_*, on the other hand, the agent does not have to perform a one-step predictive search. For each state s, only one action has to be found, which maximizes $q_*(s,a)$. Effectively, the action-value function combines all results of the single-stage predictive search. For each state-action pair, the optimal expected long-term return is displayed, allowing the selection of optimal actions without the knowledge of future states and their value and thus without knowing anything about the dynamics of the environment.

Current reinforcement learning architectures are divided into two main categories: model-based and model-free approaches. Model-based approaches attempt to model the environment and then choose the optimal policy based

on the learned model. However, in model-free methods, the agent relies on trial-and-error experience for setting up the optimal policy.

Model-based reinforcement learning has a strong influence from control theory, and the goal is to plan through an $f(s, a)$ control function to choose the optimal actions. The drawback of model-based methods is that although they have more assumptions and approximations on a given task, they may be limited only to these specific types of tasks.

On the other hand, model-free algorithms seek to learn the consequences of their actions through experience via algorithms such as Policy Gradient, Q-Learning, A3C, etc. In other words, such an algorithm will act multiple times and adjust the policy (the strategy behind its actions) for optimal rewards based on the outcomes.

Our thesis employs reinforcement learning to find the optimal policy for selecting the viewer's connections during an event. In **Paper B** [13] and **Paper C** [14], we model the viewers' connections selection process as Markov Decision Process and apply the actor-critic learning scheme to identify the optimal policy.

2.3 Meta-Learning

Conventional machine learning approaches improve model predictions over multiple data instances. However, such approaches cannot adapt and generalize to new tasks and environments. In contrast, meta-learning, or learning to learn, is a process to improve a base learning algorithm on new environments over multiple learning episodes [42]. Meta-learning achieves such fast adaptation to new environments/tasks by applying a three-step process: i) a base learning and ii) a meta-learning step, and ii) a meta-testing step. More specifically, during base learning, a learning algorithm solves a task, such as image classification, link prediction, etc, defined by a dataset and objective. During the meta-learning step, an outer (meta) algorithm updates the base learning algorithm such that the model it learns improves an external goal. The model is evaluated over a testing dataset in the meta-testing step.

In conventional machine learning, we are given a training dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, corresponding to (input, output) pairs. We can train a predictive model $\hat{y} = f_\theta(x)$ parameterized by θ, as follows:

$$\theta_* = arg\min_{\theta} \mathcal{L}(\mathcal{D}; \theta, \omega) \tag{2.13}$$

where $\mathcal{L}$ is the loss function that indicates the difference between true labels and the predictions of $f_\theta(\cdot)$. The symbol ω specifies our base learning choices, such as the optimizer for θ and the function class f. We measure the model's generalization by evaluating $\mathcal{L}$ on test data.

During base learning, the optimization of the model is performed from scratch for every task $\mathcal{D}$. Therefore, the learning choices ω are predefined. However, the basic principle of meta-learning is to improve the model's accuracy by learning how it is learning. To achieve such a learning process, meta-learning requires a distribution of tasks, and the model continuously evolves on every task rather than learning a new task from scratch.

The goal of meta-learning is to learn a general-purpose learning model that generalizes across different tasks. Ideally, it enables each new task to be learned faster than before. We can evaluate the performance of ω over a tasl distribution $p(\mathcal{T})$, where each task is defined by a dataset and loss function $\mathcal{T} = \{\mathcal{D}, \mathcal{L}\}$. At a high level, learning how to learn thus becomes

$$\min_{\omega} \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}(\mathcal{D}; \omega) \tag{2.14}$$

where $\mathcal{L}$ is a task specific loss. In typical machine-learning settings, we can split $\mathcal{D} = (\mathcal{D}^{train}, \mathcal{D}^{val})$. We define the task specific loss to be $\mathcal{L}(\mathcal{D}; \omega) = \mathcal{L}(\mathcal{D}^{val}; \theta_*(\mathcal{D}^{train}, \omega), \omega)$, where θ_* is the parameters of the model trained using the meta-knowledge ω on dataset $\mathcal{D}^{train}$.

More concretely, to solve this problem in practice, we often assume a set of M source tasks is sampled from $p(\mathcal{T})$. The set of M source tasks used in the meta training step is defined as $\mathcal{D}_{source} = \{(\mathcal{D}^{train}_{source}, \mathcal{D}^{val}_{source})^{(i)}\}_{i=1}^{M}$ where each task consists of both training and validation data. The source train and validation datasets are respectively called support and query sets, respectively. The meta-training step can be written as follows:

$$\omega_* = arg \min_{\omega} \sum_{i=1}^{M} \mathcal{L}(\mathcal{D}^{(i)}_{source}; \omega) \tag{2.15}$$

The meta-testing step is defined as $\mathcal{D} = \{(\mathcal{D}^{train}_{target}, \mathcal{D}^{test}_{target})^{(i)}\}_{i=1}^{\mathcal{Q}}$ where $\mathcal{Q}$ is the set of target tasks. In the meta-testing stage we use the learned meta-knowledge ω_* to train the base model on each previously unseen target task i:

$$\theta^{(i)}_* = arg \min_{\theta} \mathcal{L}(\mathcal{D}^{train\ (i)}_{target}; \theta, \omega_*) \tag{2.16}$$

Meta-knowledge ω_* assists the learning on the training set of a target task i regarding the algorithm to use. For example, this could affect the initial parameters of the model, the optimization strategy, etc. We evaluate the accuracy of the learned model based on the performance of $\theta^{(i)}_*$ on the test split of each target task $\mathcal{D}^{test\ (i)}_{target}$.

In our work, we employ meta-learning to generalize over different events. In **Paper B** [13], we consider each event as an independent task and the first 26 events as the source tasks M in the meta-training stage. The rest four events

correspond to the target task $\mathcal{Q}$. We train our reinforcement learning agent on the training task M and generalize the trained agent over the target task $\mathcal{Q}$.

2.4 Knowledge Distillation

Knowledge distillation has been recently introduced as a model-independent strategy to generate a small model that exhibits low online latency inference while preserving high accuracy [43]–[45]. The main idea of knowledge distillation is to train a large model, namely the *teacher*, as an offline training process. The teacher model is a neural network architecture that requires training many parameters to learn the structure of offline data. Having pretrained the teacher model, the knowledge distillation strategies compute a smaller *student* model with fewer parameters that are more suitable for deployment in production. In particular, the student model is trained on online data and distills the knowledge of the pretrained teacher model. This means that the student model mimics the teacher model and preserves the high prediction accuracy while at the same time reducing the online inference of the model parameters due to its small size [44], [46].

Knowledge distillation can be divided into three main categories: i) response-based, ii) feature-based, and iii) relation-based strategies. The response-based strategies distill knowledge to the student model by exploiting the output of the teacher model. Response-based approaches consider the final output of the teacher model as a soft label to regularize the output of the student model [47]–[49]. Accounting for the output of the teacher model to distill knowledge, the student model fails to capture the intermediate supervision applied by the teacher model [50]. Instead, feature-based approaches distill the high-level information acquired by the teacher to the output features. FitNet incorporates the features of the intermediate layers to supervise the student model, minimizing the L2 distillation loss function [51]. Moreover, Zhou et al. [52] consider the parameter sharing of intermediate layers among teacher and student models. Recently, Chen et al. [53] formulated a feature embedding task to match the dimensions of the output features generated by the teacher and student models. Although response-based and feature-based strategies distill knowledge from the outputs of specific layers in the teacher model, these approaches ignore the semantic relationship among the different layers. To handle this issue, relation-based methods explore the relationships between other feature maps. SemCKD follows a cross-layer knowledge distillation strategy to address the different semantics of intermediate layers in the teacher and student models [54]. In the IRG model, the student model distills knowledge from the Euclidean distance between examples observed by the teacher model [55]. In [56], derive an attention map from the teacher's features to distill knowledge to the student model. At the same time, KDA [57] employs the Nyström low-

rank approximation for the kernel matrix to distill knowledge through landmark points. Nevertheless, such approaches focus on image processing and have not been evaluated on evolving graphs

A few attempts have been made to follow knowledge distillation strategies to reduce the model size of graph representation learning approaches. DMTKG calculates Heat Kernel Signatures to compute the node features and inputs into Graph Convolutional Networks (GCNs) to learn accurate node embeddings [58]. DMTKG generates a compact student model by applying a response-based knowledge distillation strategy based on the weighted cross entropy loss function. DistillGCN is a feature-based strategy that exploits the output features of the teachers' GCN layers to transfer the structural information of the graph to the student model [59]. The distillation loss function in DistillGCN minimizes the prediction error of the student model on the online data and the Kullback-Leibler divergence of the features generated by the teacher and student models. However, existing knowledge distillation strategies are designed to transfer knowledge from static graphs, ignoring the evolution of dynamic graphs.

In this thesis, we employ knowledge distillation to create compact models on evolving graph neural networks. In **Paper D** [15], we propose a response-based knowledge distillation loss function to create a model with fewer parameters than the teacher model for live video streaming events. Furthermore, in **Paper E** [16], we distill knowledge from the node embeddings generated by the teacher model and demonstrate the superiority of our proposed knowledge distillation strategy on social networks. Finally, we evaluate the different response-based and feature-based knowledge distillation strategies in **Paper F** [17]. We propose a hybrid distillation loss function that acquires knowledge from teacher features and responses.

Chapter 3

Summary of appended papers

This thesis is comprised of five conference papers and one journal. In this chapter, we summarize each paper and detail our main contributions. First, we present the proposed real-time video distribution topology detection component approaches. Then, we summarize the proposed models for the network topology evolution detection component, and we conclude with the proposed approaches for the online inference latency reduction component.

3.1 Real-time video distribution topology detection

We study the problem of detecting the video distribution topology in real time. As mentioned in Chapter 1, to efficiently distribute the video content in the enterprise network, distributed solutions need to predict the high-bandwidth connections between viewers at the first minutes of the event. Failing to predict the high-bandwidth links in real-time negatively impacts the event's performance.

3.1.1 Graph Neural Networks on Evolving Graphs

Live video streaming events can be modeled as an evolving graph, where nodes correspond to the viewers and edges represent the viewer connections. The edge weights in the graph correspond to the network bandwidth between two viewers/nodes. Our goal is to detect the high-bandwidth links at the joining phase, that is, the first minutes of the event. Graphs generated by live video streaming events differ significantly over time. This occurs as viewers adapt their connections to identify the high-bandwidth connections, allowing them to distribute the video content efficiently. Graph neural networks have shown remarkable performance on link prediction tasks for evolving networks. Existing graph neural network approaches employ attention mechanisms [60], variational

graph auto encoders [61], graph convolutional neural networks (GCN) [62] to generate node embeddings that capture the evolution of the network. However, such approaches aim to capture the evolution of the network on the node embeddings at each graph snapshot. Therefore, existing methods assume that the graph structure presents small changes over a short period. Current approaches fail to capture the evolution of the live video streaming event since consecutive graph snapshots significantly differ, forcing the learned node embeddings to change completely. In **Paper A**, we overcome this problem by introducing self-attention on the weights of the GCN rather than the node embeddings. In doing so, we capture the evolution of the graph in the GCN weights and learn the node embeddings efficiently. Compared to state-of-the-art approaches, we achieve 11.4% and 11% relative drop in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics on real datasets, respectively.

Contributions. In summary, we make the following contributions:

- We study the problem of generating node embeddings on evolving graphs that differ significantly between consecutive snapshots.
- We demonstrate the limitations of existing state-of-the-art approaches to evolving graphs
- We propose a novel dynamic graph neural network approach that introduces a self-attention mechanism on the weights of the GCN.
- We experiment on real-world datasets and demonstrate the superiority of our proposed model over state-of-the-art approaches.

Related publications.

- Stefanos Antaris, Dimitrios Rafailidis, *"VStreamDRLS: Dynamic Graph Representation Learning with Self-Attention for Enterprise Distributed Video Streaming Solutions"*, ASONAM, 2020.

3.2 Network topology evolution detection

Next, we move our attention toward investigating enterprise network topology evolution detection. Enterprise network topologies frequently change to address the various business needs and the hybrid workplace environment. The distributed solutions must capture this evolution and adapt the real-time video distribution topology detection algorithms accordingly. However, real-time video distribution topology detection algorithms must adapt quickly to new network topologies. Therefore, we aim to exploit the historical events to detect the network topology changes and train machine learning models that adapt promptly to recent events.

3.2.1 Meta-Learning

During a live video streaming event, each viewer establishes a limited number of connections with other viewers to distribute the video content. To identify the high-bandwidth connections, each viewer reports several telemetry data to a backend server, namely the tracker. The tracker is responsible for detecting the high-bandwidth links and adjusting the viewers' connections accordingly. However, the enterprise network topology change frequently to meet the business requirements. The tracker must adapt quickly to identify the high-bandwidth links based on the new network topology. In **Paper B**, we formulate the selection of the viewers' connections as a Markov Decision Process and employ the actor-critic reinforcement learning scheme to learn the optimal policy on a single event. We consider the connections as a possible action of the tracker and the throughput between two viewers as the reward. In the actor-network, we employ a self-attention mechanism [63] to capture the evolution of the network on the viewer's state. Given the viewer's state, we apply a two-layer perceptron to calculate the action. In the critic network, we employ a two-layer perceptron given the state and the action of the viewer to compute the Q-value function.

A common practice to train the actor and critic network is to exploit a replay memory buffer of state-action transitions. Typically, training of the networks is performed based on a uniform sampling of the state-action transitions. However, uniform sampling prevents our model from learning from new experiences. To ensure fast adaptation to new connections, we employ a prioritized replay memory buffer based on the [64] of the viewer's throughput distribution. This allows our model to train on diverged samples, achieving fast adaptation at the beginning of the event

Additionally, to capture the evolution of the enterprise network topology, we adopt a model-agnostic meta-learning technique on previous events. We train our model on historical events, where each event is a different task. Given that each event has low structural similarity with other events, we apply a graph signature buffer to calculate the similarity between various tasks. Therefore, we manage to generalize across different topologies and adapt to any network evolution. We achieved a 25% average gain on the link weight prediction task compared with several state-of-the-art approaches to real data.

Contributions. In summary, we make the following contributions:

- We adopt the Actor-Critic reinforcement learning scheme to learn the optimal policy by incorporating the viewers' interactions.
- We propose a meta-learning component to generate the global policy of the live video streaming events and formulate two objective functions to update the actor and critic parameters, respectively.
- We design a graph signature buffer in the meta-learning component to

compute the structural similarity of different enterprise networks and events.

- We experiment on real-world datasets and demonstrate the superiority of our proposed model over state-of-the-art approaches.

Related publications.

- Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas, *"Meta-reinforcement learning via buffering graph signatures for live video streaming events"*, ASONAM, 2021

3.2.2 Gradient Boosting

In **Paper C**, we study the influence of detecting the network topology evolution on the quality of experience perceived by the viewers. Similar to our previous work, we consider that the viewers' connections are adjusted through a centralized service, namely the tracker. At the beginning of the event, most viewers have poor quality experiences. This happens because most of the viewers join simultaneously; thus, selecting the connections that will allow each viewer to have a high-quality experience is challenging. Therefore, we need to bootstrap the viewers' connection selection process to ensure a high-quality viewer experience at the beginning of the event. We formulate the viewers' connection selection process as a Markov Decision Process and apply the actor-critic reinforcement learning scheme to find the optimal policy. We exploit a graph self-attention mechanism to calculate the viewer's embedding over time, which correspond to the viewer's state, and a two-layer perceptron to compute the action of the agent/tracker. We consider all the viewers as the possible action set of the agent/tracker. The agent/tracker's reward corresponds to the Quality of Experience (QoE) of each viewer's connection. To improve the viewers' experience at the beginning of the event, we exploit the information from previous live video streaming events. In particular, we apply a gradient-boosting strategy and learn a global policy for all the events. When evaluated against state-of-the-art approaches, we achieve a 75% increase in the number of viewers that achieve a high-quality experience in the event's first minutes.

Contributions. In summary, we make the following contributions:

- We investigate the impact of high-bandwidth connections on the user experience and tackle the problem of the viewer's connection selection through deep reinforcement learning.

- We train our model on diverse state-action transitions based on the viewers' interactions to improve the user experience in the first minutes of the event.

- We propose a gradient-boosting strategy to extract information from previous events and enhance the user experience during the upcoming events' first minutes.
- We compare against several state-of-the-art approaches on real-world datasets and demonstrate the superiority of our proposed model.

Related publications.

- Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas, *"A Deep Graph Reinforcement Learning Model for Improving User Experience in Live Video Streaming"*, IEEE BigData, 2021.

3.3 Online Inference Latency Reduction

Finally, we investigate the online inference latency of the graph neural networks on evolving graphs. In particular, we explore the impact of existing approaches on inference latency in live video streaming events. We consider that live video streaming solutions require sub-second latency to detect the network topology and adjust the viewers' connections. Therefore, we propose several distillation strategies for evolving graphs to generate compact models without reducing prediction accuracy.

3.3.1 Response-based Knowledge Distillation Strategies

In **Paper D**, we investigate the impact of the number of parameters required to train the model on the online inference latency of the model. Graph neural networks achieve remarkable performance in the link prediction task on evolving graphs. However, existing approaches require many parameters to achieve such performance. Therefore, these approaches incur high inference latency during the online inference of the node representations. We employ a knowledge distillation strategy to address this problem to generate a compact model with fewer parameters. First, we capture the network evolution by introducing a self-attention mechanism on the weights of the consecutive graph convolutional networks [12]. Then we train a large model, namely the *teacher*, as an offline training process. Having pretrained the teacher model, we can train a small model with fewer parameters, the *student*. To distill knowledge from the teacher model, we propose a distillation loss function applied during the training of the student model. The distillation loss function considers the inference error of the teacher model responses on the data used to train the student model. Therefore, the student model presents low online inference latency while achieving high prediction accuracy. We evaluate our proposed knowledge distillation strategy against several baseline approaches and demonstrate the effectiveness

of our model in achieving high prediction accuracy with a low number of parameters.

Contributions. In summary, our main contributions are the following:

- We employ a self-attention mechanism on the weights of consecutive GCNs to capture the evolution of the graph.
- We are the first to study knowledge distillation on neural networks for evolving graphs.
- We formulate a distillation loss function to transfer the knowledge from the teacher model to a smaller student model.
- We significantly reduce the number of parameters when training the student mode while maintaining high prediction accuracy.

Related publications.

- Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas, *"EGAD: Evolving Graph Representation Learning with Self-Attention and Knowledge Distillation for Live Video Streaming Events"*, IEEE BigData, 2020.

3.3.2 Hybrid Knowledge Distillation Strategies

Knowledge distillation strategies can benefit graph neural networks by reducing the model size while the student model achieves high prediction accuracy. However, there are several ways of distilling knowledge from the teacher model. In particular, we investigate the impact of feature-based strategies that distill knowledge from the features generated by the intermediate layers of the teacher model and the response-based strategies that leverage the outputs of the teacher model. In particular, we train a teacher model by applying multiple self-attention mechanisms on consecutive graph snapshots [60]. To further evaluate the effectiveness of each distillation strategy on neural graph networks, in **Paper E**, we propose a knowledge distillation strategy that distills knowledge only from the intermediate features of the teacher model. We propose a distillation loss function that considers the Kullback-Leibler divergence of the teacher and student node embeddings. Finally, in **Paper F**, we propose a hybrid distillation strategy that aggregates the teacher features and responses to a joint distillation loss function. We evaluate our proposed distillation loss functions on graphs generated by social networks and live video streaming events. We demonstrate that student models trained based on the hybrid distillation function can extract more knowledge from the teacher model than the other strategies.

Contributions. In summary, our main contributions are the following:

- We propose different knowledge distillation loss functions to transfer knowledge from the teacher model to the student model.
- We propose a hybrid distillation strategy to combine the teacher's features and responses in the distillation loss function.
- We conduct extensive experimentation on social networks and networks derived from live video streaming events.
- We demonstrate that our proposed hybrid distillation strategy significantly reduces the model size while constantly outperforming the teacher model in the link weight prediction and classification tasks.

Related publications.

- Stefanos Antaris, Dimitrios Rafailidis, *"Distill2Vec: Dynamic Graph Representation Learning with Knowledge Distillation"*, ASONAM, 2020.
- Stefanos Antaris, Dimitrios Rafailidis, Sarunas Girdzijauskas, *"Knowledge distillation on neural networks for evolving graphs"*, Social Network Analysis and Mining, 11(1):100, 2021

Chapter 4

Conclusions and Future Work

It is beyond doubt that video plays an integral role in enterprise digital transformations. Large organizations spend a significant amount of their annual budget to organize live video streaming events to communicate with their employees worldwide. In doing so, enterprises anticipate a high return on investment on each live video streaming event and employ distributed video streaming solutions to distribute the video content in every office efficiently. However, such solutions require services to detect the enterprise network topology based on the viewers' interactions to distribute the video content efficiently. Machine learning, especially graph neural networks, are the most promising way to detect network topology. Our work provides methods for this direction.

First, we modeled the viewers' interactions as a graph and designed a graph neural network to capture the evolution of the graph. In particular, we evaluated the existing state-of-the-art graph neural network approaches on graphs generated by lie video streaming events. We demonstrated that current systems are not designed to capture the evolution of graphs that differ significantly among consecutive snapshots. To overcome this problem, we proposed a graph neural network that employs a self-attention mechanism on the weights of consecutive graph convolutional networks. In doing so, we manage to capture the evolution of the graph on the weights of the GCN and propagate the evolution to the generated node embeddings. We empirically showed the superiority of our proposed model over several baseline approaches on multiple real-world datasets.

Second, we address the problem of detecting the evolution of the enterprise network topology and providing fast adaptation to the model that predicts the high-bandwidth connections during an event. We formulated the viewers' connections selection process as a Markov Decision Process and solved the problem through Reinforcement Learning. We adopted the meta-learning paradigm to learn a global policy that generalizes on different network topologies to adapt quickly to new environments. Moreover, we investigated the impact of selecting

high-bandwidth connections on the quality of the user experience. To efficiently establish the links that will improve the user experience, we exploited the information from historical events and applied gradient boosting to generalize over all events. We empirically demonstrated that our approaches generalize with higher prediction accuracy than several baseline approaches on real-world datasets.

Finally, we studied the problem of high online inference latency on graph neural networks for evolving graphs. We demonstrated that the large number of parameters required to train the model impacts the online inference latency. We designed several knowledge distillation strategies on graph neural networks to reduce the latency for evolving graphs. We proposed several distillation strategies that exploit the teacher's features or responses to train the student model. Moreover, we designed a hybrid distillation strategy that combines both the features and the responses. We showed that our proposed hybrid approach achieves a high compression ratio on the student model without sacrificing the model's prediction accuracy.

This work gives rise to further exciting research questions, based on which we plan our future work: how can we train graph neural networks to learn new tasks and support lifelong learning continuously [65], [66]? Graph neural networks have achieved remarkable performance on various machine-learning tasks. However, existing approaches suffer from catastrophic forgetting when training for a new task. For example, you can train a graph neural network for a graph link prediction task for recommender systems. Still, the same model requires training from scratch for a classification task on predicting the categories of an item in the e-commerce platform. Further training a pre-trained model on data from a completely different task forces the model to focus on the new task and forget the previous task. In live video streaming solutions, it is essential to ensure the video distribution topology between viewers and understand the viewer's properties. In particular, we are interested in identifying the viewer's department, role, and interests. Given such information, distributed solutions can provide more sophisticated solutions on how the video will be distributed inside the office.

Additionally, centralized services present significant scalability challenges requiring enormous engineering effort. However, distributed solutions on the enterprise network address the scalability challenges by exploiting the viewers' resources. Therefore, an interesting research question is the decentralization of the viewers' connections selection process with multi-agent reinforcement learning [67], [68]. Without any centralized service to adjust the viewers' connections, the agents/viewers must present fast adaptation to the network topology changes and detect the high-bandwidth connections with limited knowledge based on their partial observation of the existing connections.

Bibliography

[1] *Top video technology trends 2022: The future of streaming is about device reach*, https://bitmovin.com/top-video-technology-trends/, [Online; accessed 25-May-2022], 2022.

[2] *Hours of video uploaded to youtube every minute as of february 2020*, https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/, [Online; accessed 25-May-2022], 2020.

[3] *How video analytics can help understand and increase employee engagement*, https://www.hivestreaming.com/resources/video-analytics-employee-engagement, [Online; accessed 16-May-2021], 2021.

[4] *Bringing down the walls*, https://www.akamai.com/us/en/multimedia/documents/ebooks/broadcast-over-the-internet-book-one-bringing-down-the-walls-ebook.pdf, [Online; accessed 29-January-2021], 2017.

[5] A. Sachin, J. P. S. Singh, and D. Shruti, "Analysis and implementation of gossip-based p2p streaming with distributed incentive mechanisms for peer cooperation," *Advances in Multimedia*, vol. 2007, Oct. 2007. DOI: 10.1155/2007/84150.

[6] K. Ragab and A. U. Haque, "A minimum spanning tree algorithm for efficient p2p video streaming system," in *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, vol. 1, 2010, pp. 93–98.

[7] R. Zhang, A. R. Butt, and Y. C. Hu, "Topology-aware peer-to-peer on-demand streaming," in *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, R. Boutaba, K. Almeroth, R. Puigjaner, S. Shen, and J. P. Black, Eds., 2005, pp. 1–14.

[8] W. L. Hamilton, R. Ying, and J. Leskovec, *Representation learning on graphs: Methods and applications*, 2018. arXiv: 1709.05584.

[9] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[10] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, 2014.

[12] S. Antaris and D. Rafailidis, "Vstreamdrls: Dynamic graph representation learning with self-attention for enterprise distributed video streaming solutions," in *ASONAM*, 2020, pp. 486–493.

[13] S. Antaris, D. Rafailidis, and S. Girdzijauskas, "Meta-reinforcement learning via buffering graph signatures for live video streaming events," in *ASONAM*, 2021, pp. 385–392.

[14] S. Antaris, D. Rafailidis, and S. Gidzijauskas, "A deep graph reinforcement learning model for improving user experience in live video streaming," in *Big Data*, 2021, pp. 1787–1796.

[15] S. Antaris, D. Rafailidis, and S. Girdzijauskas, "Egad: Evolving graph representation learning with self-attention and knowledge distillation for live video streaming events," in *Big Data*, 2020, pp. 1455–1464.

[16] S. Antaris and D. Rafailidis, "Distill2vec: Dynamic graph representation learning with knowledge distillation," in *ASONAM*, 2020, pp. 60–64.

[17] S. Antaris, D. Rafailidis, and S. Girdzijauskas, "Knowledge distillation on graph neural networks for evolving graphs," 2021.

[18] V. Christina, S. Karpagavalli, and G. Suganya, "Email spam filtering using supervised machine learning techniques," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 2, no. 09, pp. 3126–3129, 2010.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: `https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`.

[21] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[22] Y. Wu, M. Schuster, Z. Chen, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[23] S. Antaris and D. Rafailidis, "Sequence adaptation via reinforcement learning in recommender systems," in *RecSys*, 2021, pp. 714–718.

[24] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, "Self-supervised reinforcement learning for recommender systems," in *SIGIR*, 2020, pp. 931–940.

[25] W. Ji, K. Wang, X. Wang, T. Chen, and A. Cristea, "Sequential recommender via time-aware attentive memory network," in *CIKM*, 2020, pp. 565–574.

[26] L. Wu, P. Cui, J. Pei, and L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore, 2022, p. 725.

[27] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[28] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.

[29] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, 2016, pp. 1105–1114.

[30] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.

[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[32] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[33] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Transactions on Cybernetics*, pp. 1–13, 2019.

[34] Z. T. Kefato and S. Girdzijauskas, "Self-supervised graph neural networks without explicit negative sampling," *arXiv preprint arXiv:2103.14958*, 2021.

[35] L. Zhu, D. Guo, J. Yin, G. V. Steeg, and A. Galstyan, "Scalable temporal latent space inference for link prediction in dynamic social networks (extended abstract)," in *ICDE*, 2017, pp. 57–58.

[36] S. Mahdavi, S. Khoshraftar, and A. An, *Dynamic joint variational graph autoencoders*, 2019. arXiv: 1910.01963 [cs.LG].

[37] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *ICONIP*, 2018, pp. 362–373.

[38] P. Goyal, S. R. Chhetri, and A. Canedo, "Dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowl. Based Syst.*, vol. 187, 2020.

[39] A. Pareja, G. Domeniconi, J. Chen, *et al.*, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[40] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *WSDM*, 2020, pp. 519–527.

[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2018.

[42] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[43] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS*, 2015.

[44] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *KDD*, 2006, pp. 535–541.

[45] J. Tang and K. Wang, "Ranking distillation: Learning compact ranking models with high performance for recommender system," in *KDD*, 2018, pp. 2289–2298.

[46] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *ICML*, 2019, pp. 5142–5151.

[47] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS*, 2015.

[48] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *AAAI*, 2020, pp. 5191–5198.

[49] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," in *ICPR*, 2021, pp. 4619–4625.

[50] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[51] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *ICLR*, 2015.

[52] G. Zhou, Y. Fan, R. Cui, W. Bian, X. Zhu, and K. Gai, "Rocket launching: A universal and efficient framework for training well-performing light net," in *AAAI*, 2018.

[53] H. Chen, Y. Wang, C. Xu, C. Xu, and D. Tao, "Learning student networks via feature embedding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 25–35, 2020.

[54] D. Chen, J.-P. Mei, Y. Zhang, *et al.*, "Cross-layer distillation with semantic calibration," in *AAAI*, vol. 35, 2021, pp. 7028–7036.

[55] Y. Liu, J. Cao, B. Li, *et al.*, "Knowledge distillation via instance relationship graph," in *CVPR*, 2019, pp. 7096–7104.

[56] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[57] Q. Qian, H. Li, and J. Hu, "Efficient kernel transfer in knowledge distillation," *CoRR*, vol. abs/2009.14416, 2020.

[58] S. Lee and B. C. Song, "Graph-based knowledge distillation by multi-head attention network," *arXiv preprint arXiv:1907.02226*, 2019.

[59] Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang, "Distilling knowledge from graph convolutional networks," in *CVPR*, 2020.

[60] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *WSDM*, 2020, pp. 519–527.

[61] S. Mahdavi, S. Khoshraftar, and A. An, "Dynamic joint variational graph autoencoders," in *Machine Learning and Knowledge Discovery in Databases*, 2020, pp. 385–401.

[62] A. Pareja, G. Domeniconi, J. Chen, *et al.*, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *AAAI*, 2020.

[63] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[64] J. M. Joyce, "Kullback-leibler divergence," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. 2011, pp. 720–722.

[65] C. Wang, Y. Qiu, D. Gao, and S. Scherer, "Lifelong graph learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 719–13 728.

[66] L. Galke, B. Franke, T. Zielke, and A. Scherp, "Lifelong learning of graph neural networks for open-world node classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1–8.

[67] L. Canese, G. C. Cardarilli, L. Di Nunzio, *et al.*, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, p. 4948, 2021.

[68] W. Mao, L. Yang, K. Zhang, and T. Basar, "On improving model-free algorithms for decentralized multi-agent reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2022, pp. 15 007–15 049.

Chapter 5

Appended papers