Meta-Reinforcement Learning via Buffering Graph Signatures for Live Video Streaming Events

Stefanos Antaris
KTH Royal Institute of Technology
Hive Streaming AB
Sweden
antaris@kth.se

Dimitrios Rafailidis
University of Thessaly
Greece
draf@uth.gr

Sarunas Girdzijauskas
KTH Royal Institute of Technology
Sweden
sarunasg@kth.se

***Abstract*—In this study, we present a meta-learning model to adapt the predictions of the network's capacity between viewers who participate in a live video streaming event. We propose the MELANIE model, where an event is formulated as a Markov Decision Process, performing meta-learning on reinforcement learning tasks. By considering a new event as a task, we design an actor-critic learning scheme to compute the optimal policy on estimating the viewers' high-bandwidth connections. To ensure fast adaptation to new connections or changes among viewers during an event, we implement a prioritized replay memory buffer based on the Kullback-Leibler divergence of the reward/throughput of the viewers' connections. Moreover, we adopt a model-agnostic meta-learning framework to generate a global model from past events. As viewers scarcely participate in several events, the challenge resides on how to account for the low structural similarity of different events. To combat this issue, we design a graph signature buffer to calculate the structural similarities of several streaming events and adjust the training of the global model accordingly. We evaluate the proposed model on the link weight prediction task on three real-world datasets of live video streaming events. Our experiments demonstrate the effectiveness of our proposed model, with an average relative gain of 25% against state-of-the-art strategies. For reproduction purposes, our evaluation datasets and implementation are publicly available at https://github.com/stefanosantaris/melanie**

***Index Terms*—Meta-reinforcement learning, graph signatures, video streaming**

I. Introduction

Live video streaming services have been widely adopted by large enterprises to enable the communication among thousands of employees who are located at different offices across the world. Accounting for the high impact of video to the employee engagement and productivity, Fortune-500 companies converted more than 93% of their physical to live video streaming events [1]. However, network inefficiency and bandwidth congestion is experienced when several employees attend a high-quality live video streaming event at the

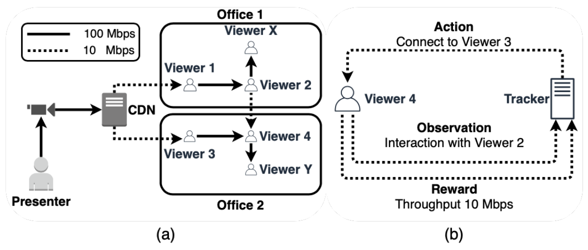

Fig. 1. (a) A distributed live video streaming process in enterprise offices with limited network capacity. (b) Selection process of each viewer's connections via the tracker.

same time. This negatively impacts the performance of the event, with more than 50% of the viewers not attending the overall event [2]. To address this problem, large enterprises exploit distributed solutions provided by live video streaming companies e.g., Hive Streaming AB[1]. Such solutions leverage the office's internal high-bandwidth network to efficiently distribute the live video stream between viewers [3], [4]. For example, as shown in Figure 1(a), Viewers 1 and 3 directly connect to the Content Delivery Network (CDN) through a low-bandwidth connection to download the live video stream of the presenter. Viewers 1 and 3 exploit the internal high-bandwidth network to distribute the live video stream to Viewers 2 and 4, respectively. Then, Viewers 2 and 4 distribute the video stream to the remaining viewers at the same office, for example to Viewers X and Y, accordingly.

To immediately select the internal high bandwidth connections, distributed video streaming solutions require prior knowledge of the company's network topology, for instance, Viewers 1 and 2 are in the same Office 1. Such information is not always feasible to be acquired, since large enterprises regularly adapt their network [3]. Therefore, it is essential to predict the network capacity of each connection in real-time during the live video streaming event based on the observed viewers' interactions. To infer the high-bandwidth connections, each viewer periodically reports various data, such as throughput, interactions, and so on, to a centralized server, namely tracker. The tracker exploits the reported data to predict the network capacity among any two viewers of

[1]https://www.hivestreaming.com/

ASONAM '21, November 8-11, 2021, Virtual Event, The Netherlands
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9128-3/21/11…$15.00
http://dx.doi.org/10.1145/3487351.3490973

the event. Then, the tracker selects the connections with the highest predicted throughput of the viewers and adapts their connections. For example, as shown in Figure 1(b), the tracker probes Viewer 4 to adjust the low-bandwidth connection (10 Mbps) with Viewer 2 and connect to Viewer 3 with a high-bandwidth connection (100 Mbps). The problem of selecting the proper connections among viewers becomes even more challenging at the beginning of the event, only after "exploring" a limited number of connections. To avoid network congestion, it is necessary to predict the best possible connection almost immediately. Nonetheless, this might only be possible by relying on the knowledge from the similar events that happened in the past on the same or similar networks.

The viewers' interactions during a video streaming event can be modeled as a temporal interaction network. The edge/interaction weight of the network corresponds to the measured throughput of the connection among two nodes/viewers. Graph representation learning has been proven to be an efficient strategy to learn low-dimensional node embeddings that capture the evolution of the network [5]–[8]. Baseline approaches exploit Recurrent Neural Networks (RNNs) [6], [7], self-attention mechanisms [8] and Long-Short Term Memory (LSTMs) units [5] to capture the network's evolution. Although evolving graph representation learning strategies can capture the temporal evolution of the graph, such strategies do not necessarily work on streaming events. At the beginning of a live video streaming event a viewer may have more low-bandwidth interactions than high-bandwidth connections [9]. Existing strategies learn to accurately predict the low weight edges/interactions and require a large number of interactions to estimate the high weights of the connections among viewers [3], [6]. The challenge here is how to accurately predict the high weights of the edges/interactions, given only a few interactions per viewer. Recently, graph representation learning approaches have been proposed to predict the high weights of edges/interactions in a live video streaming event [3], [10]. However, these approaches exploit the viewers' interactions in a single event, ignoring the knowledge of past live video streaming events.

Extracting knowledge from past streaming events is a challenging task, as a significantly low number of viewers attends multiple live video streaming events, as it will be demonstrated in Section I of our supplementary [11]. In addition, each event has different network characteristics such as edges/interactions weight distribution, viewers' emerging patterns, and so on. Therefore, each live video streaming event is considered as a new task, which the prediction model needs to adapt. *Meta-Learning* has achieved remarkable performance on computing generic models, able to adjust to new tasks when only a limited amount of information is available [12]–[14]. The goal of meta-learning is to derive a global knowledge from previous experiences, so as to rapidly adapt to a new task with only a small amount of training data. Meta-learning approaches have successfully been employed on various machine learning tasks, such as image classification [15]–[17] and recommender systems [14], [18]. However, meta-learning on graphs, where each task corresponds to a new graph, has received little attention. Meta-Graph adopts the Model-Agnostic Meta-Learning framework [13] and graph signature functions [19] to perform link prediction on new graphs with limited edges [12]. Nonetheless Meta-Graph works on static graphs, and does not reflect on the dynamic case of live video streaming events. Recently, GELS employed gradient boosting to extract information from past streaming events for improving user experience [9]. However, GELS considers equal contribution of each event when learning the global model. Provided that each event exhibits different characteristics, it is essential to compute the similarity of different events and generate an accurate global model.

In this paper, we propose a Meta-r**E**inforcement **L**e**A**r**N**ing model via buffering graph s**I**gnatur**E**s, namely MELANIE, making the following contributions: i) *We implement a task adaption component to update the global MELANIE model for a new streaming event. We adopt the Actor-Critic reinforcement learning scheme, where the agent/tracker learns an optimal policy by incorporating the viewers' interactions. To quickly adapt to a new event, we design a replay memory buffer and prioritize the stored interactions based on the viewers' throughput divergence. Hence, the agent/tracker is trained on diverse experiences and converges to the optimal policy after a few interactions.; ii) We propose a meta-learning component to generate the global MELANIE model by combining the learned policy by the task adaptation component of the current live video streaming event and the previously trained events. We formulate two objective functions to update the actor and critic parameters, respectively, based on the structural similarities of different streaming events; iii) We design a graph signature memory buffer in the meta-learning component to compute the network's structure similarity between different events. In doing so, we enforce to produce similar representations of viewers with high structural similarities in different events. The graph signature memory buffer allows the meta-learning component to generate the global MELANIE model by generalizing over several events with a few common viewers.*

Our experiments on three real-world datasets, generated by live video streaming events, demonstrate that superiority of MELANIE over several baseline approaches. The remainder of the paper is organized as follows, in Section II we formally define the problem of meta-reinforcement learning in live video streaming events and detail the proposed MELANIE model in Section III. Our experimental evaluation is presented in Section IV, and we conclude the study in Section V.

II. Preliminaries

A distributed live video streaming event is represented as a temporal interaction network, which is defined as follows [5], [6].

Temporal Interaction Network. A temporal interaction network is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with N nodes/viewers $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and an ordered sequence of T

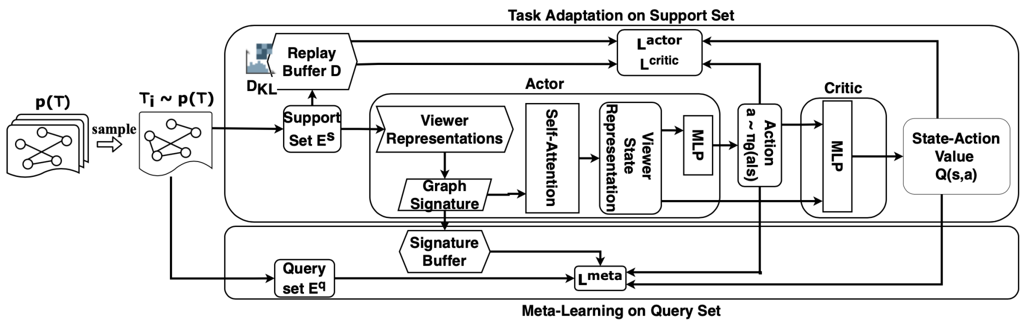

Fig. 2. Overview of the MELANIE model, given a new task, that is a new temporal interaction network $p(\mathcal{T})$. MELANIE consists of (i) the task-adaptation component on the support set $\mathcal{E}^s$ and (ii) the meta-learning component on the query set $\mathcal{E}^q$.

node/viewer interactions/connections $\mathcal{E} = \{e^1, e^2, \dots, e^T\}$. An interaction/connection $e^t_{u,v} = (u, v, t, b^t_{u,v})$ occurs between two nodes/ viewers $u \in \mathcal{V}$ and $v \in \mathcal{V}$ at time $t \in \mathbb{R}^+$, $0 \leq t \leq T$. Each interaction/connection $e^t_{u,v}$ has a weight $b^t_{u,v} \in \mathcal{B}^t$, which corresponds to the throughput measurement of node/viewer u, reported in the tracker. The set $\mathcal{B}^t$ contains the network's weights at time t.

The nodes/viewers periodically communicate with the tracker to retrieve the interactions/connections, established during a live video streaming event (Section I). Therefore, the tracker is responsible to determine the viewers' edges/connections. We model the selection of the node/viewer connections/interactions during a live video streaming event as a Markov Decision Process (MDP).

Live Video Streaming MDP A MDP is defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, with $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$ and γ being the state, action, transition probability and reward sets, and the discount factor, respectively. At each time step $t \in \mathbb{R}^+$, the tracker/agent takes an action $a^t \in \mathcal{A}$ to connect/interact node/viewer $u \in \mathcal{V}$ with node $v \in \mathcal{V}$, based on state $s^t \in \mathcal{S}$ of the node/viewer $u \in \mathcal{V}$. The tracker receives the measured throughput $b^t_{u,v}$ of the interaction $e^t_{u,v}$ as a reward $R(s^t, a^t) \in \mathcal{R}$, and the node/viewer u's state is updated to s^{t+1} with transition probability $p(s^{t+1}|s^t, a^t) \in \mathcal{P}$. The objective of the tracker/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, that maximizes the expected cumulative rewards from any state-action pairs as follows:

$$\mathcal{Q}^*(s, a) = \max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\{\sum_{k=0}^{\infty} \gamma^k R(s, a)|s = s^t, a = a^t\}$$

where $\mathbb{E}_{\pi_\theta}$ is the expectation based on the policy π_θ, γ is the discount factor, and $R(s, a) \in \mathcal{R}$ is the reward received at the time step t, given the action a^t taken at the state s^t.

Finding the optimal policy π_θ of the agent/tracker in a live video streaming event typically requires a large number of node/viewer interactions to achieve efficient performance. Our study is inspired by the gradient based meta-learning framework in reinforcement learning (meta-RL), which optimizes the global policy π_θ of the agent over several tasks, so as to rapidly adapt to a new task [13], [20]. Extracting knowledge from live video streaming events is defined as the following meta-RL process:

Meta-RL in Live Video Streaming Events In meta-RL for live video streaming events, we consider a distribution of tasks $p(\mathcal{T})$, where each task corresponds to a live video streaming event. We denote each task by $\mathcal{T}_i = (\mathcal{M}_i, \mathcal{G}_i)$, where $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ and $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ is the MDP and the temporal interaction network of the live video streaming event, respectively. During meta-training, for each task $\mathcal{T}_i \sim p(\mathcal{T})$ we divide the node/viewer interactions $\mathcal{E}_i$ in a support set $\mathcal{E}^s_i \subseteq \mathcal{E}_i$ and a query set $\mathcal{E}^q_i \subseteq \mathcal{E}_i$, with $\mathcal{E}^s_i \cap \mathcal{E}^q_i = \emptyset$. The agent/tracker adapts the policy π_θ to learn the task $\mathcal{T}_i$ based on the loss on the support set $\mathcal{E}^s_i$. Then, the meta-learner exploits the query set $\mathcal{E}^q_i$ to optimize the global policy π_θ across all tasks $p(\mathcal{T})$. The goal of meta-RL is to learn a global policy π_θ to maximize the following expected cumulative reward

$$\max_{\theta} \sum_{\mathcal{T}_i \in p(\mathcal{T})} \mathbb{E}_{\pi_\theta}\{\sum_{k=0}^{\infty} \gamma^k R(s, a)|s = s^t, a = a^t\}$$

III. Proposed Model

A. Overview of MELANIE

As illustrated in Figure 2, the proposed MELANIE model consists of two main components: the task adaptation and the meta-learning component. The goal of the proposed MELANIE model is to learn a generic model that achieves fast adaptation to new MDPs, that is new live video streaming events.

Task Adaptation The role of this component is to adapt the global policy π_θ of the tracker to a new task $\mathcal{T}_i \sim p(\mathcal{T})$. The input of the task adaptation component is the support set $\mathcal{E}^s_i$ of the sampled task $\mathcal{T}_i$. We adopt an Actor-Critic reinforcement learning scheme to model the interactions between viewers and the tracker. The policy π_θ of the tracker is trained based on the state-action transitions stored in a replay memory buffer. Following [21], we prioritize the state-action transitions in the replay memory buffer based on the KL-divergence of the viewers' reward distribution between consecutive time steps. Provided that most viewer interactions may have low reward at the beginning of the live video streaming event, we train the policy π_θ on dissimilar experiences.

Meta-learning The role of the meta-learning component is to evaluate the policy π_θ, learned by the task adaptation component, to the query set $\mathcal{E}^q_i$ of the sampled task $\mathcal{T}_i$. To measure the similarity of the sampled task $\mathcal{T}_i$ against the previously trained tasks, we introduce a signature buffer during the meta-learning process. The signature buffer contains a signature of each task $\mathcal{T}_i \sim p(\mathcal{T})$, which represents the structural information of the task's network. According to the gradient-based Model Agnostic meta-learning approach [13], we update the global policy π_θ to generate the global MELANIE model over all the computed signatures of the tasks.

B. Task Adaptation on the Support Set

The input of the task adaptation component is the support set $\mathcal{E}_i^s$ of the sampled task $\mathcal{T}_i \sim p(\mathcal{T})$. Given the viewers' interactions of the support set $\mathcal{E}_i^s$, we aim to learn the optimal policy π_θ of the specific task $\mathcal{T}_i$, by adopting a deep RL framework based on the Actor-Critic learning scheme. Therefore, the task adaptation component consists of three main parts: i) the Actor network, ii) the Critic network, and iii) the replay memory buffer.

Actor Network At each time step t, the actor network takes as input an interaction $e_{u,v}^t \in \mathcal{E}_i^s$ and then generates an action a^t for the node/viewer $u \in \mathcal{V}_i$ of the task $\mathcal{T}_i \sim p(\mathcal{T})$. We represent each node/viewer $u \in \mathcal{V}_i$ with a d-dimensional node/viewer embedding $\mathbf{X}_u \in \mathbb{R}^d$ based on the node/viewer features [22]. If the nodes/viewers do not have any node features, as in the case of live video streaming events, we calculate an embedding look up. Given an interaction $e_{u,v}^t \in \mathcal{E}_i^s$, we compute a d_s-dimensional state representation $s_u^t \in \mathbb{R}^{d_s}$ of the viewer $u \in \mathcal{V}_i$, as follows [19], [23]:

$$\mathbf{s}_u^t = g_\phi(u, \mathcal{N}_u^t) = \sigma\left(\sum_{v \in \mathcal{N}_u^t} c_{u,v} \mathbf{W}^S \mathbf{X}_u\right) \tag{1}$$

where $g_\phi(u, \mathcal{N}_u^t)$ is the attention function of the node/viewer $u \in \mathcal{V}_i$ with its neighborhood $\mathcal{N}_u^t = \{v : e_{u,v}^t \in \mathcal{E}_u^t\}$, parameterized by $\phi = \{\mathbf{W}^S\}$. The parameter weight matrix $\mathbf{W}^S \in \mathbb{R}^{d_s \times d}$ is the shared weight transformation of the node/viewer embedding $\mathbf{X}_u$ to a d_s-dimensional vector and $\sigma(\cdot)$ is the exponential linear unit (ELU) activation function. The attention coefficient $c_{u,v}$ measures the similarity of the node/viewer v to the node/viewer u as follows:

$$\begin{aligned} c_{u,v} &= \frac{exp(o_{u,v})}{\sum_{w \in \mathcal{N}_u^t} exp(o_{u,w})} \\ o_{u,v} &= \delta\left(b_{u,v}^t \cdot \mathbf{H}^T [\mathbf{W}^S \mathbf{X}_u || \mathbf{W}^S \mathbf{X}_v]\right) \end{aligned} \tag{2}$$

where $b_{u,v}^t$ is the connection weight/throughput between nodes/ viewers u and v at the time step t, $||$ is the concatenation of the node/viewer representations $\mathbf{X}_u$ and $\mathbf{X}_v$, and $\delta(\cdot)$ is the LeakyRELU non-linearity activation function. Provided that the temporal interaction network $\mathcal{G}_i$ of the sampled task $\mathcal{T}_i$ evolves over time, we capture the evolution in the state representation s_u^t, by parameterizing the attention function with a $2d_s$-dimensional graph signature $\mathbf{H} \in \mathbb{R}^{2d_s}$, which is defined as follows:

$$\mathbf{H} = \psi_\beta(\mathbf{X}) = \sigma(\mathbf{W}^\mathbf{H}\mathbf{X} + \mathbf{b}^\mathbf{H}) \tag{3}$$

where ψ_β is the aggregation of the node representations $\mathbf{X}$ to a N-dimensional signature vector $\mathbf{H}$, parameterized by $\beta = \{\mathbf{W}^\mathbf{H}, \mathbf{b}^\mathbf{H}\}$. Intuitively, the role of the graph signature $\mathbf{H}$ is to capture the structural and node/viewer features similarities of the temporal interaction network over time, and bias the node/viewer feature aggregation towards nodes/viewers with similar interactions. By computing the graph signature $\mathbf{H}$, we determine the structural similarity of viewers, and promote similar viewers in the attention process [19].

The state $\mathbf{s}_u^t$ captures the preferences of the viewer u at the t-th time step. An interaction $e_{u,v}$ with high connection throughput corresponds to a high attention coefficient $c_{u,v}$, which reflects on the preferences of the viewer u in the state representation $\mathbf{s}_u^t$. The actor network transforms the state representation s_u^t of the viewer $u \in \mathcal{V}_i$ to an N-dimensional action $\mathbf{a}^t$ vector. In our implementation, we employ a two-layer perceptron (MLP) on the state representation s_u^t as follows:

$$\mathbf{a}^t = \pi_\theta(s^t) = MLP(s^t) \tag{4}$$

We normalize the action vector $\mathbf{a}^t$ based on the softmax function and select the node/viewer with the highest value. We employ the ϵ-greedy exploration technique to learn an accurate policy π_θ [24].

Critic Network The input of the Critic network is the node/ viewer state s_u^t and the action a^t generated by the policy π_θ. The critic network outputs a scalar, which is an approximation of the true state-action value function $Q_w(s^t, a^t)$, that is the following Q-value function:

$$Q_w(s_u^t, a^t) = f_w(s_u^t, a^t) = MLP(s_u^t \oplus a^t) \tag{5}$$

where $\oplus$ denotes the concatenation of the state representation s_u^t and the action representation a^t. Here f_w is the Q-value approximation function paremeterized by w, which consists of the weights and biases in the MLP. The Q-value function represents the benefit of the action a^t generated by the Actor network, given the node/viewer state s_u^t and the policy π_θ. Following the deterministic policy gradient theorem [25], we update the parameters θ of the Actor based on the sampled policy gradient as [24]:

$$\begin{aligned} &\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{actor}(\pi_\theta; \mathcal{E}_i^s) \\ &\mathcal{L}^{actor}(\pi_\theta; \mathcal{E}_i^s) = \\ &-\tfrac{1}{K}\textstyle\sum_{t=0}^{K} log\pi_\theta(a|s)[R(s,a) - Q_w(s,a)]|_{s=s^t, a=\pi_\theta(s^t)} \end{aligned} \tag{6}$$

where $R(s,a) - Q_w(s,a)$ indicates the merit of the action a, given the state s, compared with the random action, and η is defined as the step size. The term K is the number of interactions exploited to train the actor network. The weight parameters w of the Critic network are updated accordingly via the temporal-difference learning approach, that is minimizing the following mean squared error:

$$w \leftarrow w - \eta \nabla_w \mathcal{L}^{critic}(f_w; \mathcal{E}_i^s)$$

$$\mathcal{L}^{critic}(f_w; \mathcal{E}_i^s) = \frac{1}{K}\sum_{t=0}^{K}(R(s,a) - Q_w(s,a))^2|_{s=s^t, a=\pi_\theta(s^t)} \tag{7}$$

Replay Memory Buffer To optimize the Actor and the Critic networks, we employ a replay memory buffer $\mathcal{D}$, that contains the $D = |\mathcal{D}|$ latest state-action transitions. The agent/tracker adapts the nodes/viewers connections based on their reward until the viewers weight/throughput distribution converges. Therefore, we store the D latest state-action transitions of the nodes/viewers in the replay memory buffer $\mathcal{D}$. The replay memory buffer allows the agent/tracker to learn from earlier memories, and break undesirable temporal correlations, that is similar state-action transitions in a short period of time. Instead of exploiting all the state-action transitions to train the agent, we sample a subset of the stored transitions. The most popular sampling strategy is uniform sampling, where each transition in the replay memory buffer is selected with equal probability [21], [26]. However, provided that in our case the connections at the beginning of a live video streaming event may be among viewers with a low network's capacity, the majority of the stored state-action transitions in the replay memory buffer have low reward values. Hence, uniformly sampling the state-action transitions to train the Actor and Critic network prevents the agent/tracker to learn from new experiences. To overcome this problem, we prioritize the state-action transitions based on the Kullback-Leibler (KL) divergence values $D_{KL}(\mathcal{B}^t||\mathcal{B}^{t-1})$ of the weight/throughput distribution of each viewer between consecutive time steps. In particular, to measure the difference between the weight/throughput distributions $\mathcal{B}^t$ and $\mathcal{B}^{t-1}$ of two consecutive steps, the Kullback-Leibler (KL) divergence [27] is defined as follows:

$$D_{KL}(\mathcal{B}^t||\mathcal{B}^{t-1}) = \sum_{x\in\mathcal{X}} \mathcal{B}^t(x) log\left(\frac{\mathcal{B}^t(x)}{\mathcal{B}^{t-1}(x)}\right)$$

where $\mathcal{X}$ is the equally partitioned probability space of the interactions weight/throughput. A high KL-divergence value corresponds to significant changes on the weight/throughput distributions between consecutive live video streaming minutes, whereas a low value indicates that viewers converge to their optimal connections. We retrieve the top-$K \ll D$ state-action transitions from the buffer $\mathcal{D}$ and train the tracker/agent based on Equations 6 and 7.

C. Meta-Learning on the Query Set

The goal of the meta-learning component is to learn the global weight parameters θ and w, so that the actor and critic networks can quickly adjust to a novel task $\mathcal{T}$. The input of the meta-learning component is the query set $\mathcal{E}_i^q$ of the sampled task $\mathcal{T}_i \sim p(\mathcal{T})$ and the weight parameters θ and w calculated by the task adaptation component. According to the gradient-based model of agnostic meta-learning [13], we update the weight parameters θ and w on the query set $\mathcal{E}_i^q$ as follows:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_\theta^{meta}(\pi_\theta; \mathcal{E}_i^q)$$
$$w \leftarrow w - \eta \nabla_w \mathcal{L}_w^{meta}(f_w; \mathcal{E}_i^q) \tag{8}$$

where η is the learning rate. We formulate the loss functions $\mathcal{L}_\theta^{meta}$ and $\mathcal{L}_w^{meta}$ as follows:

$$\mathcal{L}_\theta^{meta}(\pi_\theta; \mathcal{E}_i^q) = \mathcal{L}^{actor}(\pi_\theta; \mathcal{E}_i^q) - \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} KL(\mathbf{H}_i||\mathbf{H}_j)$$
$$\mathcal{L}_w^{meta}(w; \mathcal{E}_i^q) = \mathcal{L}^{critic}(f_w; \mathcal{E}_i^q) - \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} KL(\mathbf{H}_i||\mathbf{H}_j) \tag{9}$$

where the loss functions $\mathcal{L}^{actor}(\pi_\theta; \mathcal{E}_i^q)$ and $\mathcal{L}^{critic}(f_w; \mathcal{E}_i^q)$ measure the accuracy of the actor and the critic networks on the query set $\mathcal{E}_i^q$ based on Equation 6 and 7, respectively. The term $(\mathbf{H}_i||\mathbf{H}_j)$ is the KL divergence among different graph signatures. The intuition behind this divergence is to provide a model with parameters θ and w and generalize over all the previously trained graph signatures $H_j \in \mathcal{J}$.

Graph Signature Buffer The graph signature buffer $\mathcal{J}$ contains the C signatures $\mathbf{H}_i$ of each task $\mathcal{T}_i \sim p(\mathcal{T})$ that has been previously trained in the past events. Given the graph signature buffer, we compute the similarity among the newly sampled task $\mathcal{T}_i$ and the C previous trained tasks. During the update of the parameters θ and w, we minimize the average divergence of the signature $\mathbf{H}_i$ of task $\mathcal{T}_i$ and all the stored graph signatures $\mathbf{H}_j \in \mathcal{J}$. Provided that different live video streaming events have a low number of common viewers, we capture the similarities between the events and promote similar viewers that attend to more than one event. In doing so, as we will show in the experimental evaluation, the meta-learning component can generalize in several divergent live video streaming events and quickly adjust to new events within a limited number of viewers' interactions. Further details about the proposed MELANIE algorithm can be found in [11].

IV. Experimental Evaluation

TABLE I
Summary statistics of the three datasets.

Datasets	LiveStream-1	LiveStream-2	LiveStream-3
#Events	30	30	30
#Offices	62	12	56
#Enterprises	1	1	15
#Viewers (K)	24.752	28.879	148.972
#Connections (M)	1.807	0.590	4.139
Avg. #Events Per Viewer	2.722 ± 2.061	1.349 ± 0.883	1.141 ± 0.348

A. Setup

Datasets In our experiments, we evaluate our proposed MELANIE model on three real-world datasets [9]. In Table I, we summarize the statistics of each dataset and we perform an extensive analysis in Section I of our supplementary [11].

Support/Train and Query/Test Sets In our experiments, we evaluate our proposed MELANIE model in the link weight prediction task on the three examined datasets, with 30 temporal interaction networks (events). For each dataset, we consider 26 live video streaming events as training set, 2 for validation and 2 for testing. Following [14], we divide the interactions of each live video streaming event to support set $\mathcal{E}^s$ and query set $\mathcal{E}^q$ according to the interaction time, with a ratio of 8:2. The task of link weight prediction is to predict the weight $b_{u,v}$ of the interaction $e_{u,v} \in \mathcal{E}^q$ that will occur in the query set $\mathcal{E}^q$ of the test set.

Evaluation Metrics We evaluate the performance of our proposed model in terms of Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which are defined as follows:

$$\begin{aligned} RMSE &= \sqrt{\frac{1}{|\mathcal{E}^q|} \sum_{e^t_{u,v} \in \mathcal{E}^q} (Q(s,a) - R(s,a))^2} \\ MAE &= \frac{\sum_{e^t_{u,v} \in \mathcal{E}^q} (Q(s,a) - R(s,a))^2}{|\mathcal{E}^q|} \end{aligned} \tag{10}$$

where $R(s,a)$ is the reward of the throughput $b^t_{u,v}$ of the interaction $e^t_{u,v} \in \mathcal{E}^q$. Note that RMSE emphasizes more on the large errors than the MAE metric. Therefore, the RMSE metric indicates if the actions taken by the agent/tracker significantly deviate from the received reward. In addition, to measure the performance of our proposed model on how well it can adapt to new live video streaming events, we report the average reward received by the N viewers for the actions taken at each t-th minute based on the interactions in the query set $\mathcal{E}^q$ as follows:

$$r^t = \frac{1}{N} \sum_{e^t_{u,v} \in \mathcal{E}^q} R(s,a) \tag{11}$$

B. Compared Methods

We evaluate the performance of the proposed MELANIE model against several graph representation learning strategies: i) *Jodie*[2] [6] employs recurrent neural networks (RNNs) to update the node embeddings; ii) *EvolveGCN*[3] [7] exploits RNNs between consecutive graph convolutional networks; iii) *DySAT*[4] [8] adopts self-attention mechanism; and iv) *VStreamDRLS*[5] [3] employs self-attention between consecutive graph convolutional networks. Moreover, we evaluate Melanie against the following meta-learning strategies: i) *PolicyGNN*[6] [28] employs deep RL to adapt the aggregation level of convolutional networks; ii) *MetaHIN*[7] [14] captures different semantic facets of each node in a global model; iii) *Meta-Graph*[8] [12] employs model-agnostic meta-learning on static networks; and iv) *GELS*[9] [9] adopts gradient boosting on live video streaming events.

To examine the different components of our model, we compare the proposed MELANIE model with the following variants: i) *MELANIE-T* employs the deep RL scheme described in Section III on a single live video streaming event, without considering the meta-learning component. Moreover, MELANIE-T adopts uniform sampling in the replay memory buffer $\mathcal{D}$, instead of the prioritized sampling based on the KL-divergence. ii) *MELANIE-B* incorporates the prioritized based on the KL-divergence replay memory buffer $\mathcal{D}$. Similar to the MELANIE-T model, MELANIE-B ignores the meta-learning component. iii) *MELANIE-M* uses the meta-learning component in multiple events. However, MELANIE-M generates a global model, without considering the graph signatures $\mathcal{H}$ stored in the graph signature buffer $\mathcal{J}$. For reproduction purposes, the source code of the proposed MELANIE model and its variants are publicly available[10].

Environment and Parameter Settings We tuned the hyper-parameters of each model based on the validation set and a grid-selection strategy. We repeated our experiments five times, and the results were averaged over the five trials. In Section II of our supplementary [11], we detail the experimental environment and we discuss the hyper-parameter settings. Moreover, we evaluate the influence of different configurations of the replay memory and graph signature buffers on the performance of the MELANIE model.

C. Performance Evaluation

In Table II, we evaluate the performance of the examined models in terms of RMSE and MAE. The proposed MELANIE model constantly outperforms the baseline approaches in all datasets. This suggests that MELANIE can efficiently learn a policy that accurately represents the temporal interaction network. Compared with the second best method MetaHIN, the proposed MELANIE model achieves relative drops 31.8 and 34.9% in terms of RMSE and MAE in LiveStream-1, 19.0 and 33.8% in LiveStream-2, 24.6 and 19.2% in LiveStream-3. Note that the MetaHIN model outperforms the other baselines, as it employs a co-adaptation meta-learner component to capture the facets of new nodes that appear in the temporal interaction network and have a few number of interactions, as it happens in the case of live video streaming events. Instead, the other baseline approaches do not handle well the case of new nodes with few interactions in the temporal interaction network. Although MetaHIN works efficiently on a single temporal interaction network, it ignores the auxiliary information of other networks, thus having limited prediction accuracy. MELANIE overcomes this issue by performing meta-learning to generate a global model not only from a single event, but also from several past events. Moreover, we observe that MELANIE constantly outperforms the GELS baseline. Given that GELS considers equal contribution of each event during learning, it negatively impacts the model to derive the similarity between events. In addition, on inspection of Table II, we observe that MELANIE outperforms all its

[2]https://github.com/srijankr/jodie
[3]https://github.com/IBM/EvolveGCN
[4]https://github.com/aravindsankar28/DySAT
[5]https://github.com/stefanosantaris/vstreamdrls
[6]https://github.com/lhenry15/Policy-GNN
[7]https://github.com/rootlu/MetaHIN
[8]https://github.com/joeybose/Meta-Graph
[9]https://github.com/stefanosantaris/GELS

[10]https://github.com/stefanosantaris/melanie

TABLE II
METHODS' COMPARISON IN TERMS OF RMSE AND MAE. THE REPORTED VALUES ARE AVERAGED OVER ALL THE INTERACTIONS IN THE QUERY SET $\mathcal{E}^q$ OF THE TEST SET. BOLD VALUES INDICATE THE BEST METHOD.

	LiveStream-1		**LiveStream-2**		**LiveStream-3**	
Baselines	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
Jodie	0.486 ± 0.122	0.365 ± 0.075	0.398 ± 0.042	0.375 ± 0.122	0.492 ± 0.016	0.426 ± 0.023
EvolveGCN	0.521 ± 0.233	0.496 ± 0.154	0.402 ± 0.136	0.366 ± 0.231	0.566 ± 0.056	0.464 ± 0.016
DySAT	0.359 ± 0.021	0.348 ± 0.143	0.316 ± 0.032	0.297 ± 0.182	0.468 ± 0.025	0.372 ± 0.032
VStreamDRLS	0.323 ± 0.012	0.301 ± 0.053	0.293 ± 0.029	0.284 ± 0.031	0.399 ± 0.015	0.316 ± 0.033
PolicyGNN	0.282 ± 0.105	0.276 ± 0.084	0.250 ± 0.014	0.218 ± 0.029	0.325 ± 0.052	0.301 ± 0.017
MetaHIN	0.274 ± 0.021	0.269 ± 0.013	0.242 ± 0.058	0.204 ± 0.027	0.296 ± 0.055	0.203 ± 0.028
Meta-Graph	0.452 ± 0.057	0.449 ± 0.030	0.437 ± 0.016	0.429 ± 0.053	0.373 ± 0.026	0.297 ± 0.018
GELS	0.302 ± 0.012	0.288 ± 0.011	0.279 ± 0.013	0.224 ± 0.012	0.301 ± 0.019	0.294 ± 0.092
MELANIE-T	0.286 ± 0.032	0.238 ± 0.012	0.251 ± 0.106	0.246 ± 0.37	0.362 ± 0.027	0.312 ± 0.023
MELANIE-B	0.263 ± 0.029	0.224 ± 0.011	0.241 ± 0.235	0.198 ± 0.042	0.355 ± 0.010	0.295 ± 0.028
MELANIE-M	0.216 ± 0.082	0.205 ± 0.016	0.214 ± 0.164	0.183 ± 0.105	0.286 ± 0.010	0.264 ± 0.011
MELANIE	$\mathbf{0.187 \pm 0.042}$	$\mathbf{0.175 \pm 0.026}$	$\mathbf{0.196 \pm 0.013}$	$\mathbf{0.135 \pm 0.052}$	$\mathbf{0.223 \pm 0.015}$	$\mathbf{0.164 \pm 0.011}$

variants. According to the performances of the variants we can measure the impact of the prioritized replay memory and signature buffers, as well as the meta-learning process on the link prediction accuracy of MELANIE. We find that MELANIE-B outperforms MELANIE-T, when the prioritized replay memory buffer based on the KL-divergence is employed in the learning process, demonstrating the importance of training MELANIE on divergent experiences/interactions. Moreover, MELANIE-M achieves superior performance over the variants MELANIE-B and MELANIE-T, by incorporating the information of past events via the meta-learning component. However, MELANIE beats the MELANIE-T variant, showing that the computation of the graph signatures plays a crucial role when generalizing over multiple events.

D. Comparison of Meta-learning Strategies

In Figure 3, we present the performance of the examined meta-learning strategies in terms of RMSE during the evolution of the streaming events. We observe that MELANIE achieves a low RMSE value at the first minutes of LiveStream-1 and LiveStream-2. This occurs because each viewer in LiveStream-1 participates in more than 2 events, as described in Section I of our supplementary [11]. Similarly, the LiveStream-2 events occurred at a low number of offices. Therefore, the temporal interaction networks in the LiveStream-1 and LiveStream-2 datasets share structural similarities, that allows the MELANIE model to accurately learn a global policy that exhibit fast adaptation to new events in the first minutes. However, in LiveStream-3 the MELANIE model requires more streaming minutes (interactions) to adapt to the new event than in the other two datasets. This happens because the LiveStream-3 occured at 15 different enterprise networks. This means that the temporal interaction networks in LiveStream-3 have limited structural similarities. Nevertheless, MELANIE still outperforms the baselines in the LiveStream-3 dataset, reflecting the ability of our model to better adapt to dissimilar new events than the baseline strategies. Moreover, we observe that Meta-Graph underperforms in all datasets, as Meta-Graph employs meta-learning on static graphs ignoring the evolution of the temporal interaction networks. This indicates that capturing the evolution of the network over time during the meta-learning process has a significant impact on the link prediction accuracy of the examined models.

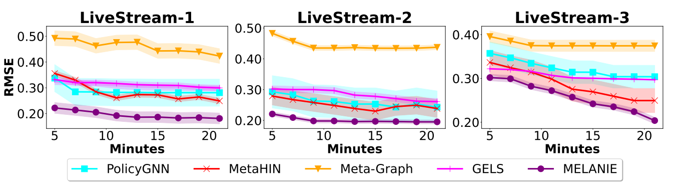

Fig. 3. Performance evaluation of the meta-learning approaches in terms of RMSE.

E. Average Reward Evaluation

In Figure 4, we demonstrate the ability of MELANIE to achieve fast adaptation to new events based on the average reward (Equation 11). The average reward is received by the agent/tracker as a reward for the connections that the N viewers have established at each streaming minute of an event. Therefore, we consider only the approaches that employ deep RL. Note that PolicyGNN, MELANIE and its variants are the only examined strategies that adopt deep RL. However, we omit PolicyGNN from this set of experiments, as the actions in PolicyGNN correspond to the number of convolutional layers applied on each node that participates in the network, rather than the actions to select the high-bandwidth connections between viewers. We observe that MELANIE constantly achieves higher reward than its variants in a low number of interactions (streaming minutes). This indicates the effectiveness of the meta-learning process to correctly learn a global policy that can efficiently adjust to a new event in the first streaming minutes. We also observe that the average

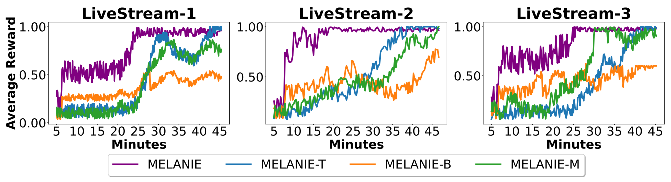

Fig. 4. Average rewards (Equation 11) of MELANIE and its variants.

rewards in MELANIE-T and MELANIE-M converge at a lower pace than MELANIE. This means that in MELANIE-T and MELANIE-M the agent/tracker is biased towards the low-bandwidth connections requiring a significant amount of interactions to optimize the policy. This occurs because MELANIE-T and MELANIE-M employ uniform sampling in the replay memory buffer, instead of prioritizing the state-action transitions based on the KL-divergence, thus they are not necessarily trained on divergent interactions.

V. Conclusions

In this study we presented the MELANIE model, a meta-RL strategy where each task corresponds a live video streaming event on large enterprise networks. We modeled each event as a MDP and then apply meta-RL to compute a global policy. To exhibit fast adaptation to a new event, we train our model on divergent interactions by prioritizing the stored state-action transitions in the replay memory buffer based on the KL-divergence of the viewers' throughputs between consecutive streaming minutes. Moreover, we introduced a graph signature buffer in the meta-learning process to measure the structural similarity among different events, allowing MELANIE to learn an accurate global model that generalizes over different events with less common viewers. Our experiments showed the proposed MELANIE model achieves high link weight prediction accuracy, with average relative drops 25.1 and 29.3% in terms of RMSE and MAE against the second best strategy.

Distributing a high-quality live video stream in an enterprise network is an intensive operation, as network inefficiencies in several offices negatively impact the performance of the live video streaming event. As as consequence, a low number of viewers might attend the event, resulting in limited viewer engagement [29]. Therefore, distributed live video streaming solutions require a fast and accurate selection of the high-bandwidth connections between viewers. Our model can support large enterprises to exploit the structural information from different events and efficiently distribute video content of higher quality, avoiding any network problems. An interesting future direction is to incorporate the video quality of experience perceived by each viewer in the learned policy of MELANIE, so as to maximize the user engagement [30].

References

[1] "Gauging demand for enterprise streaming – 2020 – investment trends in times of global change," https://www.ibm.com/downloads/cas/DEAKXQ5P, 2020, [Online; accessed 29-January-2021].

[2] "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, 2017, [Online; accessed 29-January-2021].

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

[4] R. Roverso, R. Reale, S. El-Ansary, and S. Haridi, "Smoothcache 2.0: Cdn-quality adaptive http live streaming on peer-to-peer overlays," in *MMSys*, 2015, p. 61–72.

[5] Y. Ma, Z. Guo, Z. Ren, J. Tang, and D. Yin, "Streaming graph neural networks," in *SIGIR*, 2020, p. 719–728.

[6] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *KDD*, 2019, p. 1269–1278.

[7] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *AAAI*, 2020, pp. 5363–5370.

[8] 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, p. 519–527.

[9] S. Antaris, D. Rafailidis, and S. Girdzijauskas, "A deep graph reinforcement learning model for improving user experience in live video streaming," 2021.

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

[11] "Supplementary Material," https://github.com/stefanosantaris/melanie/blob/main/supplementary/supplementary.pdf, 2021, [Online; accessed 14-July-2021].

[12] A. J. Bose, A. Jain, P. Molino, and W. L. Hamilton, "Meta-graph: Few shot link prediction via meta learning," 2020.

[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.

[14] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *SIGKDD*, 2020, p. 1563–1573.

[15] S. Khodadadeh, L. Boloni, and M. Shah, "Unsupervised meta-learning for few-shot image classification," in *NeurIPS*, 2019, pp. 10 132–10 142.

[16] Y. Zhu, C. Liu, and S. Jiang, "Multi-attention meta learning for few-shot fine-grained image recognition," in *IJCAI*, 2020, pp. 1090–1096.

[17] X. S. Wei, P. Wang, L. Liu, C. Shen, and J. Wu, "Piecewise classifier mappings: Learning fine-grained learners for novel categories with few examples," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6116–6125, 2019.

[18] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *NeurIPS*, 2017, pp. 6904–6914.

[19] M. Brockschmidt, "Gnn-film: Graph neural networks with feature-wise linear modulation," in *ICML*, 2020, pp. 1144–1152.

[20] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *ICANN*, 2001, pp. 87–94.

[21] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *ICLR*, 2016.

[22] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *SIGKDD*, 2019, p. 1073–1082.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *ICLR*, 2018.

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

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014, p. I–387–I–395.

[26] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting fundamentals of experience replay," in *ICML*, 2020, pp. 3061–3071.

[27] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, pp. 79–86, 1951.

[28] K.-H. Lai, D. Zha, K. Zhou, and X. Hu, "Policy-gnn: Aggregation optimization for graph neural networks," in *KDD*, 2020, p. 461–471.

[29] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *SIGCOMM*, 2011, p. 362–373.

[30] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *MM*, 2018, p. 1208–1216.