A Deep Graph Reinforcement Learning Model for Improving User Experience in Live Video Streaming

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  
sarunag@kth.se

ABSTRACT
In this paper we present a deep graph reinforcement learning model to predict and improve the user experience during a live video streaming event, orchestrated by an agent/tracker. We first formulate the user experience prediction problem as a classification task, accounting for the fact that most of the viewers at the beginning of an event have poor quality of experience due to low-bandwidth connections and limited interactions with the tracker. In our model we consider different factors that influence the quality of user experience and train the proposed model on diverse state-action transitions when viewers interact with the tracker. In addition, provided that past events have various user experience characteristics we follow a gradient boosting strategy to compute a global model that learns from different events. Our experiments with three real-world datasets of live video streaming events demonstrate the superiority of the proposed model against several baseline strategies. Moreover, as the majority of the viewers at the beginning of an event has poor experience, we show that our model can significantly increase the number of viewers with high quality experience by at least 75% over the first streaming minutes. Our evaluation datasets and implementation are publicly available at https://publicresearch.z13.web.core.windows.net

KEYWORDS
User experience, live video streaming, graph reinforcement learning

1 INTRODUCTION
Live video streaming has been established as a primary communication service during the digital transformation of enterprises. As part of the internal communication among employees, Fortune-500 companies on a daily basis schedule live video streaming events such as virtual town halls, employees’ education through webinars, and so on. Accounting for the high value of live video streaming services, companies invest a significant amount of their annual budgets so that every employee who attends an event can have a similar high-quality user experience as in person communication [8]. However, in the real-world setting there are several factors hindering a high-quality user experience, for example, the bandwidth limitations of several offices when several employees attend an event simultaneously. This results in a low user experience which not only negatively impacts the employee’s engagement, but also the event’s return of investment [4, 9, 14].

In an attempt to address this problem large organizations focus on distributed live video streaming solutions, trying to overcome the offices’ bandwidth limitation by reducing the number of viewers in each office that simultaneously fetch the video stream directly by the Content Delivery Network (CDN) server [32, 48]. Having established a limited number of connections, when a viewer receives the video stream she distributes it to the rest of the viewers. Distributed solutions aim at exploiting the internal high-bandwidth network to improve the user experience. As shown in Figure 1, viewers 1 and 3 are connected directly to the CDN server to download the video stream in the offices 1 and M, respectively. Thereafter, viewer
1 distributes the video content to viewers 2 and K, and viewer 3 to viewer N via high-bandwidth connections (100 Mbps). In practice though, immediately selecting the high-bandwidth connections among the viewers is not always feasible. The main reason is the absence of prior knowledge of the organization’s large-scale enterprise network, for instance the information that viewers 3, 4 and N are located in the same office, due to privacy issues and network’s upgrading tasks [12]. Given the information of a limited number of connections, distributed solutions have to predict the high-bandwidth connections in real-time during the live video streaming event. In live video streaming events, distributed solutions set up a centralized server - tracker to select the viewer’s connections. The goal of the tracker is to detect the proper connections, allowing viewers to have a high-quality user experience. To achieve this, the tracker monitors various data that are periodically reported by viewers, such as connection bandwidth, viewer’s player quality, and so on. Then, the tracker processes these data to predict the network capacity among viewers and select the high-bandwidth connections. For example, given the predicted network capacity among viewers in Figure 1, the tracker adjusts the connections of viewer 4 and probes viewer 3 to link to viewer 4 with a high-bandwidth connection.

When an event starts, several viewers join at the same time. Selecting the connections that will allow each viewer to have a high-quality user experience from the first minutes of the event is a challenging task, as most of the viewers initially establish low-bandwidth connections. This happens because the tracker requires a large amount of viewers’ interactions, to accurately predict the high-bandwidth connections among the viewers. Therefore, the majority of the viewers have a poor user experience at the beginning of the event. In addition, the bandwidth capacity among viewers might change over time, as viewers emerge at unfavorable locations with low-bandwidth connections during the event [11, 12]. As a consequence, viewers tend to leave the event when their user experience is significantly reduced [8, 11, 42]. Therefore, it is essential not only to predict the user experience in real-time, but also to predict the proper connections to improve the user experience.

In a live video streaming event, the viewers’ network can be modeled as an evolving graph, where the nodes/viewers adjust their edges/connections based on the tracker selection at each minute of the event. In addition, viewers are tagged with dynamic labels such as bad, poor, fair, good, and excellent, corresponding to the user experience per viewer at a certain minute of the event [38, 45, 47]. The goal of the tracker is to predict the viewers’ dynamic labels and select their connections accordingly to improve the user experience. In the node classification task, evolving graph representation learning strategies compute low-dimensional node embeddings to classify the nodes’ dynamic labels. Baseline approaches run random walks on consecutive graph snapshots to capture the evolution of the graph [29], other approaches apply self-attention mechanisms [12, 33, 44], recurrent neural networks (RNNs) [19, 30], and Long-Short Term Memory (LSTM) units [40]. However, as we will demonstrate in Section 4 baseline evolving graph representation learning strategies require a significant amount of viewers’ connections to achieve high prediction accuracy in the user experience. This occurs because at the beginning of the event most viewers are connected with low-bandwidth connections, thus baseline strategies mainly focus on accurately predicting the poor user experience. Moreover, such approaches are not designed to select the connections that will improve the user experience in the next minute of the event. As a consequence they do not necessarily work on streaming events, where given a limited number of viewers’ interactions at the beginning of an event the tracker needs to predict the viewers’ experience at the next minute and exploit the predictions to adjust the connections accordingly. In addition, learning from various past events can also leverage the prediction of user experience from evolving networks with different sets of users. In several applications boosting strategies have proven an effective means for improving the accuracy of weak classifiers [27, 35]. Nonetheless, the influence of boosting strategies on user experience based on past streaming events has not been studied yet.

To face the user experience challenges, we propose a Graph rEinforcement Learning model for live video Streaming events (GELS), making the following contributions:

- We address the user experience prediction task as a classification problem and solve it with deep reinforcement learning, formulating an event as a Markov Decision Process (MDP) and accounting for different factors that influence the user experience in live video streaming events.
- We train our model on the most diverse viewers’ transitions in user experience, and then exploit the predicted labels to select the connections that will improve the user experience at the first minutes of the event.
- We present a gradient boosting strategy to extract information from past events and improve the user experience via GELS over several video streams.

The remainder of this paper is organized as follows, in Section 2 we detail the proposed GELS model and in Section 3 we analyse the evolution of user experience in real-world events. Then, in Section 4 we present the experimental evaluation of our model against baseline strategies, and conclude the study in Section 5.

2 PROPOSED MODEL

2.1 Quality of User Experience in Live Video Streaming Events

A streaming event is modeled as an evolving graph \( G = (V, E^t) \). At each minute \( t = 1, \ldots, T \) of the event, \( n \) viewers of set \( V = \{u_1, u_2, \ldots, u_n\} \) interact with the tracker to select connections \( e^t_{u,v} \in E^{t+1} \) between viewers \( u \) and \( v \in V \) in the next minute \( t + 1 \). We formulate the selection process of the viewers’ connections during a live video streaming event as a Markov Decision Process (MDP), which is defined as \( M = (S, A, P, R, \gamma) \), with \( S, A, P \) being the state, action and transition probability sets, and \( \gamma \) the discount factor. At the \( t \)-th minute, the tracker/agent takes an action \( a^t \in A \) to connect viewers \( u \) and \( v \) based on a state \( s^t_u \in S \) of viewer \( u \). At the next minute \( t + 1 \), the tracker receives a reward \( R(s^t, a^t) \in R \) for the previously selected action \( a^t \) and the state \( s^t_u \) of viewer \( u \) is updated to \( s^{t+1} \) with transition probability \( p(s^t_{u,v}, a^t, d^t) \in P \). The reward \( R(s^t, a^t) \) corresponds to the user experience \( Q^t_{u,v} \) of viewer \( u \), given her connection \( e^t_{u,v} \) with viewer \( v \).
In the reinforcement learning setting, we consider the user experience $QoE_{u,v}$ as the reward that viewer $u$ provides to the tracker. We first divide the live video stream in multiple segments $c$ [16, 31], and then calculate the user experience $QoE_{u,v}$ of the connection $c_{t_u}$ at the $t$-th minute by considering the following factors [25, 45]:

- **Average Video Quality**: For each viewer $u$ the average quality $q$ of the $K$ video segments distributed by viewer $v$:

  \[
  \frac{1}{K} \sum_{k=1}^{K} q(c_k) 
  \]

- **Average Quality Variations**: The magnitude of changes in the video quality between consecutive segments:

  \[
  \frac{1}{K-1} \sum_{k=1}^{K-1} |q(c_{k+1}) - q(c_k)| 
  \]

- **Rebuffer**: The number of player rebufferings occurred to viewer $u$ when receiving the video segment $c$. Given the size $d_k(c_k)$ of a segment $c$ and the bandwidth $b_u$ of viewer $u$ at the $t$-th minute, a rebuffering occurs when the download time $d_k(c_k) / b_u$ is larger than the player buffer size $B$:

  \[
  \frac{1}{K} \sum_{k=1}^{K} \left\{ \frac{d_k(c_k)}{b_u} > B \right\} 
  \]

Accounting for the quality factors in Equations 1-3, we define the user experience $QoE_{u,v}$ of connection $c_{t_u}$ between viewers $u$ and $v$ at the $t$-th minute as follows:

\[
QoE_{u,v} = \frac{1}{K} \sum_{k=1}^{K} q(c_k) - \lambda \frac{1}{K-1} \sum_{k=1}^{K-1} |q(c_{k+1}) - q(c_k)| - \mu \frac{1}{K} \sum_{k=1}^{K} \left\{ \frac{d_k(c_k)}{b_u} > B \right\}
\]

where $\lambda$ and $\mu$ are non-negative weighting parameters to balance the importance of each factor in the user experience. A low $\lambda$ value indicates that the user experience is more affected by the average video quality than the variability of the quality among consecutive segments. For large $\mu$ values the user experience emphasizes more on the rebuffering occurrences during the streaming event.

Given the user experience $QoE_{u,v}$ between viewers $u$ and $v$ at the $t$-th minute, the objective of the tracker/agent is to compute an optimal policy $\pi_\theta : S \times A \rightarrow [0, 1]$, where $\theta$ are the parameters of policy $\pi$. To learn the optimal policy $\pi_\theta$, the tracker optimizes the Bellman equation, by maximizing the state-action transition value $Q_u(s', a') = \max_{\pi_\theta} V_{\pi_\theta} \left( \sum_{x_{\in V}} R(s', a') \right)$ [13]. The state-action transition value $Q_u(s', a')$, parameterized by $w$, represents the expected value of the action $a'$ based on the policy $\pi_\theta$.

### 2.2 Improving User Experience via GELS

To find the optimal policy $\pi_\theta$, we employ the actor-critic scheme of reinforcement learning [17, 37]. In Figure 2, we illustrate an overview of the interaction of viewer $u$ with the tracker via GELS. At the $t$-th minute of the event, viewer $u$ reports the neighborhood $N_u^t$ to the actor. The actor network computes the state vector $s_u^t \in \mathbb{R}^d$ and outputs the action vector $a_u^t \in \mathbb{R}^d$. The tracker adjusts the connections of viewer $u$, by selecting the action $a_u^t$ with the highest value in the action vector $a_u^t$. Then, viewer $u$ adjusts her connections and gives a reward $R(s', a')$ of the action $a'$ for the next minute $t+1$. The critic network takes as input the state vector $s'$ and action vector $a'$ to estimate the state-action value $Q(s', a')$. The critic network exploits the estimated state-action value $Q(s', a')$ and the reward $R(s', a')$, to optimize the parameters of the model based on a temporal-difference error $L$ (Equation 6). In doing so, the tracker evaluates: i) the performance of the critic network to predict the user experience of viewer $u$ at the next minute $t + 1$, and ii) the ability of the actor network to compute the actions at the $t$-th minute to improve the experience of viewer $u$ at the next minute $t + 1$.

### 2.3 Deep Graph Reinforcement Learning

In the actor-critic model, at each minute $t = 1, \ldots, T$ the input of the actor model is the neighborhood $N_u^t$ of viewer $u$. Provided that viewers have no features, we compute an embedding lookup to extract the node embedding $x_u \in \mathbb{R}^d$ for each viewer $u$ [21, 23]. Given the node embeddings $x_v$ of each neighbor-viewer $v \in N_u^t$, the actor network computes a state vector $s_u^t$, which corresponds to the state $s_u^t$ of viewer $u$ at the $t$-th minute. We calculate the state vector $s_u^t$ based on a self-attention mechanism as follows [39]:

\[
s_u^t = \sigma \left( \sum_{v \in N_u} o_{u,v} Wz_v \right), \quad \text{where } \sigma(\cdot) \text{ is the non-linear activation function ReLU, and } W \in \mathbb{R}^{d \times d} \text{ is the weight matrix parameter.}
\]

Variable $o_{u,v}$ corresponds to the normalized attention coefficient between $u \in V$ and $v \in N_u^t$, which is calculated according to the softmax function as follows:

\[
o_{u,v} = \frac{\exp \left( \sigma(\omega_u \cdot \alpha^T [W_{z_u} || W_{z_v}]) \right)}{\sum_{y \in N_u} \exp \left( \sigma(\omega_u \cdot \alpha^T [W_{z_u} || W_{z_y}]) \right)}
\]

where $\alpha^T$ is the $2d$-dimensional weight vector applied to the attention process. The symbol $||$ denotes the concatenation among the transformed node embeddings $z_u$ and $z_v$. Given the state $s_u^t$ of $u$, we have to compute the action vector $a_u^t$, which reflects on the
probability of improving the experience of viewer \( u \) when connecting with viewer \( v \). For each viewer \( u \), we consider all the viewers \( V \setminus \{u\} \) as the set of possible actions \(\mathcal{A} \) at the \( t \)-th minute. The action vector \( a_{tu}^t \) is computed by employing a two-layer perceptron as follows:

\[
a_{tu}^t = \sigma(W^2 ReLU(W^1 s_{tu}^t)), \quad \text{with } W^1 \in \mathbb{R}^{m \times d} \text{ and } W^2 \in \mathbb{R}^{n \times m},
\]

being the weight matrices of the first and second layers, and \( m \) the intermediate dimensions between the two layers. In our implementation, we select the optimal action, that is the action of connecting \( u \) with the viewer with the highest probability of improving the user experience, by employing the softmax function on the action vector \( a_{tu}^t \).

To evaluate the performance of the actor network when selecting the respective actions for improving the user experience, we adopt the critic network. Having computed the action \( a_{tu}^t \) and the state \( s_{tu}^t \) at the \( t \)-th minute, the role of the critic network is to estimate the improvement of the experience of viewer \( u \) at the next minute \( t + 1 \). Therefore, the input of the critic network is the state vector \( s_{tu}^t \) and the action vector \( a_{tu}^t \) of the node \( u \) at the \( t \)-th minute. The critic network outputs a scalar state-action value \( Q_{tu}(s_{tu}^t, a_{tu}^t) \), parameterized by \( w \), which is an approximation of the true state-action value function, namely the Q-value function. We calculate the state-action value by employing a two-layer perceptron as follows:

\[
Q_{tu}(s_{tu}^t, a_{tu}^t) = MLP(s_{tu}^t, a_{tu}^t).
\]

The actor-critic network is optimized based on a replay memory buffer, which contains the \( k \) latest state-action transitions, resulting in the most diverse user experience. This is achieved by exploiting the KL-divergence on the QoE value of each viewer \( u \in V \) between consecutive minutes as follows [18]:

\[
KL(\text{QoE}_u^t \mid \text{QoE}_u^{t+1}) = \sum_{u \in \mathcal{N}_u} \left( \frac{\text{QoE}_u^{t+1}}{\text{QoE}_u^t} \log \left( \frac{\text{QoE}_u^{t+1}}{\text{QoE}_u^t} \right) \right)
\]

This means that the replay memory buffer contains the state-action transitions of the viewers which significantly change the user experience between consecutive minutes. As we will demonstrate in Section 4, in doing so the tracker/agent learns the optimal policy \( \pi_{tu} \) at the first minutes of the event, allowing the viewers to achieve a high-quality user experience. Based on the replay memory buffer, we optimize the parameters of the actor and critic model, following the temporal-difference learning strategy [37], by minimizing the following mean-squared error loss function:

\[
L = \frac{1}{2} \sum_{t=1}^{T} \sum_{u \in \mathcal{N}_u} \left( E_{u}^{t+1} \left[ R(s^{t+1}, a^{t+1}) + \gamma Q(s^{t+1}, a^{t+1}) - Q(s^{t}, a^{t}) \right] \right)^2
\]

2.4 Gradient Boosting Strategy

To improve the user experience via GELS over several live video streaming events and learn a global policy \( \pi_{\theta} \), we follow a gradient boosting strategy [26, 36]. In particular, we exploit a set \( M \) of historical events and formulate each event as a MDP \( M_i \in \mathcal{M} \). For each event \( M_i \), we divide the connections \( E_i \) in two sets, the training set \( E_i^{t+1} \) and the test set \( E_i^{t+1} \). The training set \( E_i^{t+1} \) is used to adjust the policy \( \pi_{\theta} \) to an event \( M_i \). This is achieved by adopting the actor-critic network of Section 2.3 to learn the optimal policy \( \pi_{\theta} \) on each event. The actor network selects actions based on the connections that exist in the training set \( E_i^{t+1} \). To derive a global model over several events, we adjust the parameters \( \phi \) and \( w \) of the actor and critic network, respectively, by adopting a boosting strategy on the test set \( E_i^{t+1} \). More specifically, we employ back-propagation on the parameters \( \phi \) and \( w \) as follows:

\[
\phi \leftarrow \phi - \eta \frac{\partial \text{L}_{\text{test}}^{E_i}(a_{tu}, Q_u, a_{tu}, E_i^{t+1})}{\partial \phi}
\]

\[
w \leftarrow w - \eta \frac{\partial \text{L}_{\text{test}}^{E_i}(a_{tu}, Q_u(s_{tu}^t, a_{tu}), E_i^{t+1})}{\partial w}
\]

where \( L \) is the loss function in Equation 6. Finally, we generate a global model over the set \( M \) of different events, by minimizing the loss function over all the events \( M_i \in \mathcal{M} \) as follows:

\[
\min_{\theta} \sum_{M_i \in \mathcal{M}} \text{L}_{\text{test}}^{E_i}(a_{tu}, Q_u, E_i^{t+1})
\]

3 USER EXPERIENCE IN LIVE VIDEO STREAMING EVENTS

Evaluation Datasets We collected three real-world datasets, that is the LiveStream-1, LiveStream-2 and LiveStream-3 datasets, which were anonymized and made publicly available. Each dataset consists of 30 evolving graphs, corresponding to 30 different live video streaming events occurred in large enterprise networks. The duration of each event was 45 minutes. To generate each evolving graph, we monitored the viewers’ connections and their interactions with the tracker during the events. The LiveStream-1 dataset consists of 24,752 viewers and 1.8 million connections over 30 events. Each viewer participated in 2.7 different events on average, and the viewers were distributed to 62 offices of the same enterprise network. The LiveStream-2 dataset consists of 28,879 viewers in total, who were spread to 12 offices of the same organization. Each viewer participated in 1.3 events on average, and the viewers established 590,000 connections in total. The LiveStream-3 dataset was generated based on 30 events of 15 different enterprise networks. The dataset consists of 148,982 viewers and 4.1 million connections. Each viewer participated in 1.1 events on average and the viewers were distributed to 56 offices.

Evolution of User Experience. To generate the different user experience labels, we normalized the QoE value of each viewer \( u \) based on the maximum possible QoE value that the viewer \( u \) could have achieved at the \( t \)-th minute, that is when the viewer \( u \) attended at the highest video quality without having quality variations and rebuffal. Given the normalized user experience, we performed binning of equal-length in the QoE values and assigned the following user experience labels to the viewers: bad, poor, fair, good and excellent for every streaming minute [38, 45, 47]. In Figure 3, we present the average number of viewers for each user experience label over the streaming minutes. We observe that in the LiveStream-1 and LiveStream-3 datasets, most of the viewers achieved good and excellent user experience after the first 10 and 15 streaming minutes, respectively. Moreover, the tracker required more than 20 streaming minutes to select the optimal viewers’ connections and achieve excellent user experience for the majority of the viewers. This indicates that the tracker required a large amount
of interactions with the viewers to identify the high-bandwidth connections. In Livestream-2, the majority of the viewers had fair or good user experience at the first 5 streaming minutes. Most of the viewers achieved excellent user experience after 10 streaming minutes. This means that the majority of the connections in Livestream-2 exhibited high network capacity, allowing the viewers to efficiently distribute the video content and achieve excellent user experience.

**Viewers’ Connection Level.** As aforementioned in Section 1, during a live video streaming event each viewer has a limited number of connections. Therefore, it is essential for the tracker to identify the high-bandwidth connections at the first minutes of the event and achieve excellent user experience. In the evaluation datasets we categorized each connection between two viewers as low, medium and high based on the connection throughput. We applied binning on the connection throughput as follows: < 5 Mbps throughput were considered as low connections, 5 – 10 Mbps medium, and > 10 Mbps high. As illustrated in Figure 4, in LiveStream-1 and LiveStream-3 datasets, most of the viewers had less than 1 high-bandwidth connections in the first 10 minutes. After the first 10 minutes, the viewers managed to establish more than 1 high-bandwidth connections. Instead, in LiveStream-2 most of the viewers were able to connect to more than 1 high-bandwidth connections at the first 5 minutes, indicating that the majority of the viewers in LiveStream-2 were located either in the same office, such as the headquarters of the organization, or in different offices connected with a high-bandwidth network such as a fiber optic network [43].

**KL-Divergence Distribution.** In Figure 5, we present the average distribution of the KL- divergence of the user experience during the events (Equation 5). We observe that in all datasets the viewers had significant differences in the user experience, as each live video streaming event occurred in different offices of the organization with various network capacity characteristics. Therefore, the tracker had to adjust the connections’ selection policy in real-time so as to achieve excellent user experience. Moreover, we observe that in all datasets, most of the events have high KL-divergence
in the good and excellent user experience labels. This means that the viewers’ experiences were significantly changing during the streaming minutes.

4 EXPERIMENTS

4.1 Setup

Evaluation Protocol In our experiments, we evaluate the performance of the proposed GELS model on the user experience classification task. Given the viewers’ connections and user experience labels at the $t$-th minute, the goal of the classification task is to predict the user experience label that each viewer will have in the next minute $t + 1$. Provided that we have multiple user experience labels (Section 3), we follow a One-vs-Rest strategy to perform binary classification for each user experience label. The classification accuracy of our model is evaluated based on the area under the ROC curve (AUC), Micro-F1 and Macro-F1 metrics [15, 34]. AUC represents the degree of separability of each user experience label. Micro-F1 emphasizes on the common labels, while Macro-F1 penalizes models with low accuracy in the rare labels. Following the evaluation protocol of [23, 30, 33], the connections and user experience labels of each event are divided into the training $\mathcal{E}^{tr}$ and test sets $\mathcal{E}^{ts}$. The training set $\mathcal{E}^{tr}$ consists of all the user experience labels and connections that viewers established in the first 30 streaming minutes and the rest of user experience labels and connections correspond to the test set $\mathcal{E}^{ts}$. In each dataset, we randomly select 26 events to train our model, 2 events are considered for validation and the remaining 2 events for testing. The goal of the classification task of user experience is to predict the label of the viewers in the test events. We repeated our experiments five times and report average AUC, Micro-F1 and Macro-F1.

Baseline Strategies We evaluate the performance of the proposed GELS model against the following strategies: DyGNN\textsuperscript{1} [24] a graph neural network that captures the evolution of the graph by adopting LSTM units in the evolving connections. EvolveGCN\textsuperscript{2} [30] a dynamic learning approach that adopts Gated Recurrent Units (GRUs) on the weights of consecutive Graph Convolutional Networks (GCNs). DySAT\textsuperscript{3} [33] a graph representation learning strategy that employs a multi-head self-attention mechanism to compute the graph’s structural and temporal evolution. VStreamDRLS\textsuperscript{4} [12] a deep neural network model that applies a self-attention mechanism to calculate the evolution of the graph on the weights of consecutive GCN models. MetaHIN\textsuperscript{5} [23] a meta-learning strategy on evolving graphs that captures the different semantic facets of each node/viewer. PolicyGNN\textsuperscript{6} [20] a graph neural network strategy that performs reinforcement learning to compute the required number of layers in the GCNs. GELS\textsuperscript{7} a variant of our model, which ignores the gradient boosting strategy of past events in Section 2.4 and trains the actor-critic model on a single event.

Environment. All our experiments were conducted on a single server with an Intel Xeon Bronze 3106, 1.70GHz CPU. We accelerated the modes’ training using the GPU GeForce RTX 2080 Ti graphic card. The operating system of the server was Ubuntu 18.04 LTS. We implemented our proposed GELS model in Pytorch 1.7.1 and created the reinforcement learning environment using the OpenAI Gym 0.17.3 library. To measure the user experience, we emulated the video player of each viewer based on the dash.js library [7]. For the models’ training on each event, the reinforcement learning environment was initialized using the training set $\mathcal{E}^{tr}$. This means that the tracker selected an action $a^t$ to establish a connection between viewers $u$ and $v$ based on the training set $\mathcal{E}^{tr}$ at each minute $t$ of the event. Each viewer monitors (i) the quality of the video, reported by the dash.js library, (ii) the connection throughput and (iii) the reward returned to the agent, that is the $QoE_{u,v}^t$ value of the connection $e_{u,v}$ at time $t$. Similarly, the reinforcement learning environment was initialized to evaluate the performance of the learned policy $\pi^t$ in the test set $\mathcal{E}^{ts}$.

Configuration. For each examined model, we tuned the hyperparameters following a grid-selection strategy on the validation set, and report the performance based on the best configuration. In DyGNN we set the node embedding dimension to 256 in all datasets. In EvolveGCN and DySAT the node embedding is fixed to 128 and 256 in all datasets, respectively. Moreover, EvolveGCN exploits the information from 2 consecutive graph snapshots, while DySAT from 3 snapshots in all datasets. In MetaHIN, we use 128-dimensional node embeddings for LiveStream-1 and LiveStream-3, and 32-dimensional embeddings for Livestream-2. PolicyGNN uses 64-dimensional embeddings in all datasets, the discount factor $\gamma$ is set to 0.92 and the $\epsilon$ value in the $\epsilon$-greed exploration-exploitation strategy is fixed to 0.1. In the proposed GELS model and its variant GELS*, we fix the node embeddings size to 128 in all datasets. The size of the replay memory buffer is set to $k=64$ state-action transitions, the discount factor $\gamma$ is 0.96 and the $\epsilon$ value is fixed to 0.2. To calculate the $QoE$ value in Equation 4, we fix the user experience parameters to $\lambda=0.2$ and $\mu=0.3$ [45].

4.2 Performance Comparison over Streaming

In Table 1, we evaluate the performance of the examined models in terms of AUC, Micro-F1, and Macro-F1 over the events’ streaming minutes. The proposed GELS model constantly outperforms the baseline approaches in all datasets. This indicates that GELS can efficiently predict the experience of each viewer. Compared with the second best method MetaHIN, GELS achieves relative improvements of 4.67, 2.16, and 5.88% in terms of AUC, Macro-F1 and Micro-F1, respectively, in LiveStream-1. In the LiveStream-2 dataset, the relative improvements are 3.29, 5.86, and 3.87%, and 7.52, 6.93 and 14.43% in LiveStream-3. MetaHIN performs better than the other baseline approaches by following a meta-learning strategy on the connections, achieving high prediction accuracy for new viewers that emerge during an event. However, MetaHIN ignores the information provided by historical events and therefore requires a large amount of new connections to achieve a high classification accuracy. The proposed GELS model overcomes this shortcoming by adopting the boosting strategy on different streaming events, achieving high classification accuracy at the first minutes of the

---

1https://github.com/alge24/dygnn
2https://github.com/IBM/EvolveGCN
3https://github.com/aravindsankar28/DySAT
4https://github.com/stefanosantaris/vstreamdrls
5https://github.com/rootlu/MetaHIN
6https://github.com/lhenry15/Policy-GNN
7https://publicresearch.x13.web.core.windows.net
| LiveStream-1 | Baselines | Streaming Minutes |
|----------------|----------|------------------|
|                | AUC      |                  |
|                | DyGNN    | 492 ± 355        |
|                | EvolveGCN| 513 ± 112        |
|                | DySAT    | 614 ± 231        |
|                | VStreamDRLS | 638 ± 015       |
|                | MetaHIN | 692 ± 132        |
|                | PolicyGNN| 695 ± 202        |
|                | GELS     | 772 ± 016        |
|                | Micro-F1 |                  |
|                | DyGNN    | 693 ± 256        |
|                | EvolveGCN| 752 ± 173        |
|                | DySAT    | 802 ± 426        |
|                | VStreamDRLS | 833 ± 126       |
|                | MetaHIN | 874 ± 396        |
|                | PolicyGNN| 852 ± 161        |
|                | GELS     | 812 ± 016        |
|                | Macro-F1 |                  |
|                | DyGNN    | 699 ± 224        |
|                | EvolveGCN| 703 ± 296        |
|                | DySAT    | 716 ± 108        |
|                | VStreamDRLS | 739 ± 102       |
|                | MetaHIN | 762 ± 131        |
|                | PolicyGNN| 729 ± 017        |
|                | GELS     | 803 ± 010        |
|                |          |                  |
| LiveStream-2 | Baselines | Streaming Minutes |
|                | AUC      |                  |
|                | DyGNN    | 518 ± 609        |
|                | EvolveGCN| 552 ± 168        |
|                | DySAT    | 567 ± 201        |
|                | VStreamDRLS | 579 ± 012       |
|                | MetaHIN | 618 ± 013        |
|                | PolicyGNN| 602 ± 018        |
|                | GELS     | 782 ± 025        |
|                | Micro-F1 |                  |
|                | DyGNN    | 568 ± 348        |
|                | EvolveGCN| 691 ± 162        |
|                | DySAT    | 718 ± 392        |
|                | VStreamDRLS | 739 ± 102       |
|                | MetaHIN | 769 ± 305        |
|                | PolicyGNN| 742 ± 023        |
|                | GELS     | 783 ± 010        |
|                | Macro-F1 |                  |
|                | DyGNN    | 573 ± 273        |
|                | EvolveGCN| 586 ± 183        |
|                | DySAT    | 595 ± 251        |
|                | VStreamDRLS | 612 ± 035       |
|                | MetaHIN | 698 ± 032        |
|                | PolicyGNN| 726 ± 013        |
|                | GELS     | 765 ± 012        |
|                |          |                  |
| LiveStream-3 | Baselines | Streaming Minutes |
|                | AUC      |                  |
|                | DyGNN    | 587 ± 316        |
|                | EvolveGCN| 612 ± 116        |
|                | DySAT    | 647 ± 251        |
|                | VStreamDRLS | 648 ± 015       |
|                | MetaHIN | 640 ± 001        |
|                | PolicyGNN| 648 ± 001        |
|                | GELS     | 646 ± 012        |
|                | Micro-F1 |                  |
|                | DyGNN    | 596 ± 316        |
|                | EvolveGCN| 612 ± 116        |
|                | DySAT    | 647 ± 251        |
|                | VStreamDRLS | 648 ± 015       |
|                | MetaHIN | 640 ± 001        |
|                | PolicyGNN| 648 ± 001        |
|                | GELS     | 646 ± 012        |

Table 1: Methods’ comparison in terms of AUC, Micro-F1 and Macro-F1 over the streaming minutes. Bold values indicate the best method using a statistical significance t-test with p < 0.01.
Table 2: Models’ comparison in terms of average AUC based on the user experience labels.

| QoE Label | Baselines | Datasets |
|-----------|-----------|---------|
|           | LiveStream-1 | LiveStream-2 | LiveStream-3 |
| Bad       | DyGNN: 417 ± 237 | 569 ± 294 | 443 ± 286 |
|           | EvolveGCN: 492 ± 174 | 397 ± 149 | 514 ± 169 |
|           | DySAT: 513 ± 221 | 413 ± 251 | 542 ± 236 |
|           | VStreamDRLS: 557 ± 017 | 458 ± 014 | 601 ± 011 |
|           | MetaHIN: 618 ± 154 | 589 ± 367 | 719 ± 182 |
|           | PolicyGNN: 502 ± 013 | 513 ± 011 | 693 ± 014 |
|           | GELS*: 782 ± 011 | 812 ± 017 | 730 ± 016 |
|           | GELS: 896 ± 012 | 903 ± 013 | 782 ± 012 |
| Poor      | DyGNN: 551 ± 385 | 623 ± 299 | 678 ± 212 |
|           | EvolveGCN: 562 ± 168 | 654 ± 172 | 689 ± 129 |
|           | DySAT: 589 ± 228 | 682 ± 216 | 692 ± 241 |
|           | VStreamDRLS: 612 ± 013 | 694 ± 014 | 703 ± 013 |
|           | MetaHIN: 745 ± 192 | 731 ± 351 | 712 ± 342 |
|           | PolicyGNN: 702 ± 016 | 708 ± 018 | 706 ± 015 |
|           | GELS*: 815 ± 013 | 801 ± 013 | 772 ± 014 |
|           | GELS: 901 ± 014 | 932 ± 014 | 780 ± 014 |
| Fair      | DyGNN: 679 ± 315 | 731 ± 299 | 658 ± 336 |
|           | EvolveGCN: 656 ± 167 | 742 ± 164 | 664 ± 166 |
|           | DySAT: 692 ± 251 | 761 ± 247 | 679 ± 267 |
|           | VStreamDRLS: 713 ± 013 | 788 ± 021 | 684 ± 022 |
|           | MetaHIN: 814 ± 162 | 843 ± 377 | 738 ± 183 |
|           | PolicyGNN: 732 ± 011 | 795 ± 012 | 692 ± 911 |
|           | GELS*: 819 ± 012 | 800 ± 013 | 714 ± 014 |
|           | GELS: 944 ± 017 | 936 ± 017 | 782 ± 018 |
| Good      | DyGNN: 703 ± 302 | 713 ± 284 | 652 ± 291 |
|           | EvolveGCN: 712 ± 183 | 758 ± 126 | 689 ± 192 |
|           | DySAT: 759 ± 255 | 789 ± 271 | 693 ± 288 |
|           | VStreamDRLS: 812 ± 014 | 802 ± 015 | 712 ± 015 |
|           | MetaHIN: 866 ± 156 | 896 ± 316 | 760 ± 121 |
|           | PolicyGNN: 789 ± 014 | 773 ± 014 | 713 ± 013 |
|           | GELS*: 820 ± 017 | 816 ± 011 | 742 ± 012 |
|           | GELS: 945 ± 015 | 941 ± 015 | 782 ± 013 |
| Excellent | DyGNN: 718 ± 321 | 772 ± 293 | 631 ± 264 |
|           | EvolveGCN: 802 ± 145 | 812 ± 162 | 655 ± 177 |
|           | DySAT: 823 ± 211 | 834 ± 276 | 682 ± 288 |
|           | VStreamDRLS: 844 ± 018 | 847 ± 020 | 719 ± 019 |
|           | MetaHIN: 902 ± 154 | 895 ± 361 | 768 ± 172 |
|           | PolicyGNN: 817 ± 014 | 782 ± 013 | 760 ± 013 |
|           | GELS*: 823 ± 017 | 894 ± 022 | 770 ± 020 |
|           | GELS: 946 ± 016 | 950 ± 018 | 783 ± 016 |

4.3 Evaluation based on QoE Labels and Connection Quality

In Table 2, we evaluate the classification accuracy of the examined models for each user experience label, by averaging the AUC metric over all the streaming minutes. We observe that the proposed GELS model achieves high classification accuracy in all user experience labels for all datasets. This indicates the ability of GELS to accurately classify each viewer in the correct user experience label. When compared with the second best approach MetaHIN, the proposed GELS model achieves relative improvements of 31.03, 17.31, 13.77, 8.36, and 4.65% for LiveStream-1 in the bad, poor, fair, good and excellent user experience labels, respectively, for LiveStream-2 34.77, 21.57, 9.93, 4.78, and 5.70%, while for LiveStream-3 8.05, 8.72, 5.63, 2.81, and 1.92%. On inspection of Table 2, we observe that MetaHIN has higher classification accuracy on the fair, good and excellent user experiences than in the bad and poor labels. This means that MetaHIN is biased on the training data and cannot generalize well on all the user experience labels. As aforementioned in Section 1, the main challenge in live video streaming events is to predict the bad and poor labels, since the majority of the viewers have poor user experience at the first minutes of the event. The proposed GELS model handles this problem by employing a replay memory buffer to store the most diverse state-action transitions in terms of the viewers’ user experience.

In Table 3, we study the impact of the connection level on the classification accuracy of the examined models in terms of average AUC. We observe that the proposed GELS model outperforms the baseline strategies for all connection levels. This occurs because our model exploits information from several past live video streaming events. In doing so, the learned policy accurately predicts the user experience level of each viewer, regardless of the network capacity of their connections. When compared with MetaHIN, the proposed GELS model achieves relative improvements 28.62, 13.11, and 4.44% in LiveStream-1 in the low, medium, and high connection levels, respectively. Similarly, the relative improvements in LiveStream-2 are 44.27, 13.56, and 6.22%, and 26.04, 4.62, 1.41% for the LiveStream-3 dataset. As aforementioned in Section 3, most of the viewers establish low bandwidth connections at the beginning of the event, hence, it is essential to accurately classify the low, medium and medium connection levels. In doing so, GELS can improve the user experience of the viewers with low-bandwidth connections at the first streaming minutes of the live video events.

4.4 Improvement of User Experience

So far, in the previous sets of experiments in Sections 4.3 and 4.4 we evaluated the prediction accuracy of the user experience labels.
A pressing question resides on how well the proposed GELS model can not only accurately predict the user experience label, but also improve the user experience in the next streaming minutes by establishing the right connections between viewers. In this set of experiments, we measure the impact of the proposed GELS model, comparing the initial user experience labels of Figure 3 with the user experience labels when we apply GELS and take the necessary actions to improve the user experience as presented in Section 2.2. In Figure 6, we present the relative improvement of the proposed GELS model in terms of average number of viewers, categorized by the user experience labels over the streaming minutes. We observe that in all datasets our model achieves more than 75% relative improvement on the viewers that achieve good and excellent user experience at the first 5 minutes of the event. Note that, as described in Section 3, in the LiveStream-1 and LiveStream-3 datasets, initially most of the viewers achieve good or excellent user experience after the first 20 minutes of the event. By employing our model the tracker can predict the user experience of each viewer, and furthermore incorporate the predicted user experience in the connection selection process of the tracker. Thus, our model can clearly improve the user experience for most of the viewers at the first minutes of the events.

**5 CONCLUSIONS**

In this paper, we presented a deep graph reinforcement learning model for improving the experience of each viewer when attending enterprise live video streaming events. In the proposed GELS model we formulate each event as a MDP and address the problem of viewers’ connection selection based on a graph reinforcement learning strategy. Our model focuses on the viewers that exhibit poor user experience at the first minutes and are connected with low-bandwidth connections. To improve the user experience of the viewers at the first minutes of the event, our model is trained on diverse state-action transitions based on the KL-divergence of each viewer’s experience. In addition, we employ a gradient boosting strategy, to generate a global model that learns from different events. In doing so, our model achieves high prediction accuracy on the user experience at the first minutes of an event, and incorporates the predictions in the connections selection process of the tracker. Our experiments on three real-world datasets showed that the proposed GELS model achieves high classification accuracy, with average relative improvements of 5.16, 4.98, 8.72% in terms of AUC, Macro-F1 and Micro-F1, respectively, when compared with the second best strategy.

The proposed GELS model can be directly deployed to real-world live video events and improve the user experience over the streaming minutes. Over the last years, there has been a remarkable demand of live video streaming events by large enterprises. Since 2019, Fortune-500 companies host more than 93% events every year for both external and internal communication [5]. This growth occurs because viewers retain 95% of a message when they watch it in a video content, while only 10% when the same message is shared through text [2, 22]. On a daily basis, more than 70% of the employees attend at least one live video streaming [3]. In doing so, large enterprises are able to improve their employees engagement, and thus their productivity [6, 9]. However, viewers might lose their interests of the live video streaming event when they perceive poor user experience, and more than 50% of viewers abandon the event before it finishes [1]. By deploying our model, large enterprises can attract a large audience to their events and increase their employees’ engagement.[14]. Large enterprises invest a significant amount of their budget in live video streaming solutions. In 2018, the enterprise video market was estimated at $14 billion, and is expected to exceed $24 billion by 2026 [10]. This indicates the importance of each live video streaming event to have a high return of investment. This means that the organizations can increase the reach and impact of the video streaming events on most of the employees. Therefore, by employing our model large enterprises can efficiently distribute the video content to all employees at a high-quality, improving significantly the return of investment of each event.

In future work we plan to design a multi-task reinforcement learning strategy, to incorporate the video player’s bitrate selection process in the live video streaming events [28, 41, 46].

**REFERENCES**

[1] 2017. 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].
[2] 2017. Looking at the Facts-Why Video Content Has the Highest Retention Rate. https://www.popvideo.com/blog/looking-at-the-facts-why-video-content has-the-highest-retention-rate. [Online; accessed 29-January-2021].
[3] 2018. Live Video Streaming: A Global Perspective. https://www.iab.com/insights/live-video-streaming-2018/. [Online; accessed 29-January-2021].
