frontiers in Big Data

ORIGINAL RESEARCH
published: 13 June 2019
doi: 10.3389/fdata.2019.00014

Deep Neural Networks for Optimal Team Composition

Anna Sapienza[†], *Palash Goyal*[†] *and Emilio Ferrara*[*]

USC Information Sciences Institute, Los Angeles, CA, United States

Cooperation is a fundamental social mechanism, whose effects on human performance have been investigated in several environments. Online games are modern-days natural settings in which cooperation strongly affects human behavior. Every day, millions of players connect and play together in team-based games: the patterns of cooperation can either foster or hinder individual skill learning and performance. This work has three goals: (i) identifying teammates' influence on players' performance in the short and long term, (ii) designing a computational framework to recommend teammates to improve players' performance, and (iii) setting to demonstrate that such improvements can be predicted via deep learning. We leverage a large dataset from Dota 2, a popular Multiplayer Online Battle Arena game. We generate a directed co-play network, whose links' weights depict the effect of teammates on players' performance. Specifically, we propose a measure of network influence that captures skill transfer from player to player over time. We then use such framing to design a recommendation system to suggest new teammates based on a modified deep neural autoencoder and we demonstrate its state-of-the-art recommendation performance. We finally provide insights into skill transfer effects: our experimental results demonstrate that such dynamics can be predicted using deep neural networks.

Keywords: recommendation system, link prediction, deep neural network, graph factorization, multiplayer online games

OPEN ACCESS

Edited by:
Hanghang Tong,
Arizona State University, United States

Reviewed by:
Yidong Li,
Beijing Jiaotong University, China
Jingrui He,
Arizona State University,
United States

***Correspondence:**
Emilio Ferrara
emiliofe@usc.edu

[†]*These authors have contributed equally to this work*

Specialty section:
This article was submitted to Data Mining and Management, a section of the journal Frontiers in Big Data

Received: *13 September 2018*
Accepted: *27 May 2019*
Published: *13 June 2019*

Citation:
Sapienza A, Goyal P and Ferrara E (2019) Deep Neural Networks for Optimal Team Composition. Front. Big Data 2:14. doi: 10.3389/fdata.2019.00014

1. INTRODUCTION

Cooperation is a common mechanism present in real world systems at various scales and in different environments, from biological organization of organisms to human society. A great amount of research has been devoted to study the effects of cooperation on human behavior and performance (Deutsch, 1960; Johnson and Johnson, 1989; Beersma et al., 2003; Tauer and Harackiewicz, 2004; Levi, 2015). These works include domains spanning from cognitive learning to psychology, and cover different experimental settings (e.g., classrooms, competitive sport environments, and games), in which people were encouraged to organize and fulfill certain tasks (Johnson et al., 1981; Battistich et al., 1993; Cohen, 1994; Childress and Braswell, 2006). These works provide numerous insights on the positive effect that cooperation has on individual and group performance.

Many online games are examples of modern-day systems that revolve around cooperative behavior (Hudson and Cairns, 2014; Losup et al., 2014). Games allow players to connect from all over the world, establish social relationships with teammates (Ducheneaut et al., 2006; Tyack et al., 2016), and coordinate together to reach a common goal, while trying at the same time to compete with the aim of improving their performance as individuals

(Morschheuser et al., 2018). Due to their recent growth in popularity, online games have become a great instrument for experimental research. Online games provide indeed rich environments yielding plenty of contextual and temporal features related to player's behaviors as well as social connection derived from the game organization in teams.

In this work, we focus on the analysis of a particular type of online games, whose setting boosts players to collaborate to enhance their performance both as individuals and teams: Multiplayer Online Battle Arena (MOBA) games. MOBA games, such as League of Legends (LoL), Defense of the Ancient 2 (Dota 2), Heroes of the Storm, and Paragon, are examples of match-based games in which two teams of players have to cooperate to defeat the opposing team by destroying its base/headquarter. MOBA players impersonate a specific character in the battle (a.k.a., hero), which has special abilities and powers based on its role, e.g., supporting roles, action roles, etc. The cooperation of teammates in MOBA games is essential to achieve the shared goal, as shown by prior studies (Drachen et al., 2014; Yang et al., 2014). Thus, teammates might strongly influence individual players' behaviors over time.

Previous research investigated factors influencing human performance in MOBA games. On the one hand, studies focus on identifying player's choices of role, strategies as well as spatio-temporal behaviors (Drachen et al., 2014; Yang et al., 2014; Eggert et al., 2015; Sapienza et al., 2018b) that drive players to success (Sapienza et al., 2017; Fox et al., 2018). On the other hand, performance may be affected by player's social interactions: the presence of friends (Pobiedina et al., 2013a; Park and Kim, 2014; Sapienza et al., 2018b), the frequency of playing with or against certain players (Losup et al., 2014), etc.

Despite the efforts of quantifying performance in presence of social connections, little attention has been devoted to connect the effect that teammates have in increasing or decreasing the actual player's skill level. Our study aims to fill this gap. We hypothesize that some teammates might indeed be beneficial to improve not only the strategies and actions performed but also the overall skill of a player. On the contrary, some teammates might have a negative effect on a player's skill level, e.g., they might not be collaborative and tend to obstacle the overall group actions, eventually hindering player's skill acquisition and development.

Our aim is to study the interplay between a player's performance improvement (resp., decline), throughout matches in the presence of beneficial (resp., disadvantageous) teammates. To this aim, we build a directed co-play network, whose links exist if two players played in the same team and are weighted on the basis of the player's skill level increase/decline. Thus, this type of network only take into account the short-term influence of teammates, i.e., the influence in the matches they play together. Moreover, we devise another formulation for this weighted network to take into account possible long-term effects on player's performance. This network incorporates the concept of "memory", i.e., the teammate's influence on a player persists over time, capturing temporal dynamics of skill transfer. We use these co-play networks in two ways. First, we set to quantify the structural properties of player's connections related to skill performance. Second, we build a teammate recommendation system, based on a modified deep neural network autoencoder, that is able to predict their most influential teammates.

We show through our experiments that our teammate autoencoder model is effective in capturing the structure of the co-play networks. Our evaluation demonstrates that the model significantly outperforms baselines on the tasks of (i) predicting the player's skill gain, and (ii) recommending teammates to players. Our predictions for the former result in a 9.00 and 9.15% improvement over reporting the average skill increase/decline, for short and long-term teammate's influence respectively. For individual teammate recommendation, the model achieves an even more significant gain of 19.50 and 19.29%, for short and long-term teammate's influence respectively. Furthermore, we show that using a factorization based model only marginally improves over average baseline, showcasing the necessity of deep neural network based models for this task.

2. DATA COLLECTION AND PREPROCESSING

2.1. Dota 2

Defense of the Ancient 2 (Dota 2) is a well-known MOBA game developed and published by Valve Corporation. First released in July 2013, Dota 2 rapidly became one of the most played games on the Steam platform, accounting for millions of active players.

We have access to a dataset of one full year of Dota 2 matches played in 2015. The dataset, acquired via *OpenDota*[1], consists of 3,300,146 matches for a total of 1,805,225 players. For each match, we also have access to the match metadata, including winning status, start time, and duration, as well as to the players' performance, e.g., number of kills, number of assists, number of deaths, etc., of each player.

As in most MOBA games, Dota 2 matches are divided into different categories (lobby types) depending on the game mode selected by players. As an example, players can train in the "Tutorial" lobby, or start a match with AI-controlled players in the "Co-op with AI" lobby. However, most players prefer to play with other human players, rather than with AIs. Players can decide whether the teams they form and play against shall be balanced by the player's skill levels or not, respectively in the "Ranked matchmaking" lobby and the "Public matchmaking" lobby. For Ranked matches, Dota 2 implements a matchmaking system to form balanced opposing teams. The matchmaking system tracks each player's performance throughout her/his entire career, attributing a skill level that increases after each victory and decreases after each defeat.

[1] Cui, A., Chung, H., and Hanson-Holtry, N. (2015). Yasp 3.5 million data dump.

For the purpose of our work, we take only into account the Ranked and Public lobby types, in order to consider exclusively matches in which 10 human players are involved.

2.2. Preprocessing

We preprocess the dataset in two steps. First, we select matches whose information is complete. To this aim, we first filter out matches ended early due to connection errors or players that quit at the beginning. These matches can be easily identified through the winner status (equal to a null value if a connection error occurred) and the leaver status (players that quit the game before end have leaver status equal to 0). As we can observe in **Figure 1**, the number of matches per player has a broad distribution, having minimum and maximum values of 1 and 1,390 matches respectively. We note that many players are characterized by a low number of matches, either because they were new to the game at the time of data collection, or because they quit the game entirely after a limited number of matches.

In this work we are interested in assessing a teammate's influence on the skill of a player. As described in the following section, we define the skill score of a player by computing his/her TrueSkill (Herbrich et al., 2007). However, the average number of matches per player that are needed to identify the TrueSkill score in a game setting as the one of Dota 2 is 46[2]. For the scope of this analysis, we then apply a second preprocessing step: we select all the players having at least 46 played matches. These two filtering steps yielded a final dataset including 87,155 experienced players.

3. SKILL INFERENCE

Dota 2 has an internal matchmaking ranking (MMR), which is used to track each player's level and, for those game modes requiring it, match together balanced teams. This is done with the main purpose of giving similar chance of winning to both teams. The MMR score depends both on the actual outcome of the matches (win/lose) and on the skill level of the players involved in the match (both teammates and opponents). Moreover, its standard deviation provides a level of uncertainty for each player's skill, with the uncertainty decreasing with the increasing number of player's matches.

Player's skill is a fundamental feature that describes the overall player's performance and can thus provide a way to evaluate how each player learns and evolves over time. Despite each player having access to his/her MMR, and rankings of master players being available online, the official Dota 2 API does not disclose the MMR level of players at any time of any performed match. Provided that players' MMR levels are not available in any Dota 2 dataset (including ours), we need to reconstruct a proxy of MMR.

We overcome this issue by computing a similar skill score over the available matches: the TrueSkill (Herbrich et al., 2007). The TrueSkill ranking system has been designed by Microsoft Research for Xbox Live and it can be considered as a Bayesian extension of the well-known Elo rating system, used in chess (Elo, 1978). The TrueSkill is indeed specifically developed to compute the level of players in online games that involve more than two players in a single match, such as MOBA games. Another advantage of using such ranking system is its similarity with the Dota 2 MMR. Likewise MMR, the TrueSkill of a player is represented by two main features: the average skill of a player μ and the level of uncertainty σ for the player's skill[3].

Here, we keep track of the TrueSkill levels of players in our dataset after every match they play. To this aim, we compute the TrueSkill by using its open access implementation in Python[4]. We first generate for each player a starting TrueSkill which is set to the default value in the Python library: $\mu = 25$, and $\sigma = \frac{25}{3}$. Then, we update the TrueSkill of players on the basis of their matches' outcomes and teammates' levels. The resulting timelines of scores will be used in the following to compute the link weights of the co-play network.

For illustrative purposes, **Figure 2** reports three aggregate TrueSkill timelines, for three groups of players: (i) the 10th percentile (bottom decile), (ii) the 90th percentile (top decile), and (iii) the median decile (45–55th percentile). The red line shows the evolution of the average TrueSkill scores of the 10% top-ranked players in Dota 2 (at the time of our data collection); the blue line tracks the evolution of the 10% players reaching the lowest TrueSkill scores; and, the green line shows the TrueSkill progress of the "average players." The confidence bands (standard deviations) shrinks with increasing number of matches, showing how the TrueSkill converges with increasing observations of players' performance[5]. The variance is larger for high TrueSkill scores. Maintaining a high rank in Dota 2 becomes increasingly more difficult: the game is designed to constantly pair players with opponents at their same skill levels, thus competition in "Ranked matches" becomes increasingly harsher. The resulting score timelines will be used next to compute the link weights of the co-play network. Note that, although we selected only players with at least 46 matches, we observed timelines spanning terminal TrueSkill scores between 12 and 55. This suggests that experience alone (in terms of number of played matches) does not guarantee high TrueSkill scores, in line with prior literature (Herbrich et al., 2007).

4. NETWORK GENERATION

In the following, we explain the process to compute the co-play performance networks. In particular, we define a short-term performance network of teammates, whose links reflect TrueSkill score variations over time, and a long-term performance

[2]https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/

[3]https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/

[4]https://pypi.python.org/pypi/trueskill

[5]Note that the timelines have different length due to the varying number of matches played by players in each of the three deciles. In particular, in the bottom decile just one player has more than 600 matches

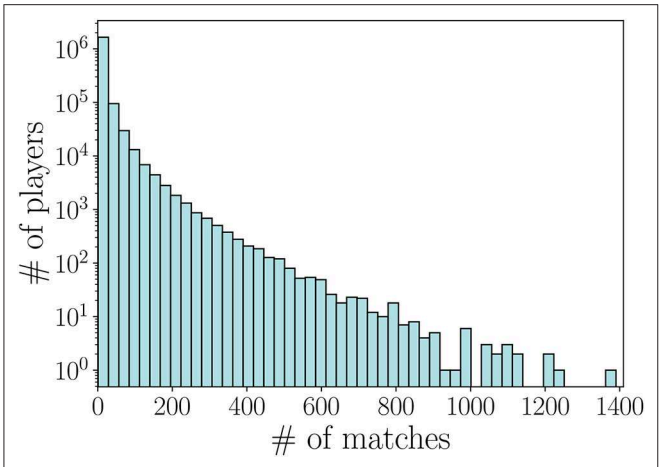

FIGURE 1 | Distribution of the number of matches per player in the Dota 2 dataset.

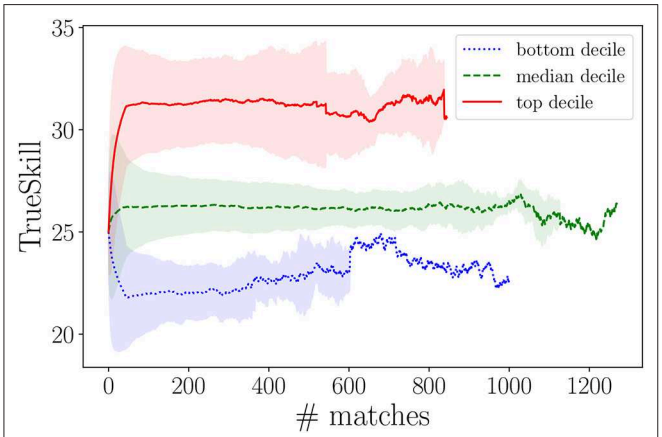

FIGURE 2 | TrueSkill timelines of players in the top, bottom, and median decile. Lines show the mean of TrueSkill values at each match index, while shades indicate the related standard deviations.

network, which allows to take memory mechanisms into account, based on the assumption that the influence of a teammate on a player can persist over time.

4.1. Short-Term Performance Network

Let us consider the set of 87,155 players in our post-processed Dota 2 dataset, and the related matches they played. For each player p, we define $TS_p = [ts_{-1}, ts_0, ts_1, \cdots, ts_N]$ as the TrueSkill scores after each match played by p, where ts_{-1} is the default value of TrueSkill assigned to each player at the beginning of their history. We also define the player history as the temporally ordered set $M_p = [m_0, m_1, \cdots, m_N]$ of matches played by p. Each $m_i \in M_p$ is the 4-tuple (t_1, t_2, t_3, t_4) of player's teammates. Let us note that each match m in the dataset can be represented as a 4-tuple because we consider just Public and Ranked matches, whose opposing teams are composed by 5 human players each. We can now define for each teammate t of player p in match $m_i \in M_p$ the corresponding performance weight, as:

$$w_{p,t,i} = ts_i - ts_{i-1}, \tag{1}$$

where, $ts_i \in TS_p$ is the TrueSkill value of the player p after match $m_i \in M_p$. Thus, weight $w_{p,t,i}$ captures the TrueSkill gain/loss of player p when playing with a given teammate t. This step generates as a result a time-varying directed network in which, at each time step (here the temporal dimension is defined by the sequence of matches), we have a set of directed links connecting together the players active at that time (i.e., match) to their teammates, and the relative weights based on the fluctuations of TrueSkill level of players.

Next, we build the overall Short-term Performance Network (SPN), by aggregating the time-varying networks over the matches of each player. This network has a link between two nodes if the corresponding players were teammates at least once in the total temporal span of our dataset. Each link is then characterized by the sum of the previously computed weights. Thus, given player p and any possible teammate t in the network, their aggregated weight $w_{p,t}$ is equal to

$$w_{p,t} = \sum_{i=0}^{N} w_{p,t,i}, \tag{2}$$

where $w_{p,t,i} = ts_i - ts_{i-1}$ if $t \in m_i$, and 0 otherwise. The resulting network has 87,155 nodes and 4,906,131 directed links with weights $w_{p,t} \in [-0.58, 1.06]$.

It is worth noting that the new TrueSkill value assigned after each match is computed on the basis of both teammates and opponents current skill levels. However, the TrueSkill value depends on the outcome of each match, which is shared by each teammate in the winner/loser team. With this system in place, players that do not cooperate in the game, such as players that do not perform any kill or assist, and win will improve their skill level because of the teammates' effort. Nevertheless, this anomalous behavior is rare (i.e., less than 1% of matches are affected) and it is smoothed by our network model. By aggregating the weights over a long period of time, we indeed balance out these singular instances.

4.2. Long-Term Performance Network

If skills transfer from player to player by means of co-play, the influence of a teammate on players should be accounted for in their future matches. We therefore would like to introduce a memory-like mechanism to model this form of influence persistence. Here we show how to generate a Long-term Performance Network (LPN) in which the persistence of influence of a certain teammate is taken into account. To this aim, we modify the weights by accumulating the discounted gain over the subsequent matches of a player as follows. Let us consider player p, his/her TrueSkill scores TS_p and his/her temporally ordered sequence of matches M_p. As previously introduced, $m_i \in M_p$ corresponds to the 4-tuple (t_1, t_2, t_3, t_4) of player's teammates in that match. For each teammate t of

player p in match $m_i \in M_p$ the long-term performance weight is defined as

$$w_{p,t,i} = \exp^{i-i_{p,t}}(ts_i - ts_{i-1}), \quad (3)$$

where $i_{p,t}$ is the index of the last match in M_p in which player p played with teammate t. Note that, if the current match m_i is a match in which p and t play together than $i_{p,t} = i$.

Analogously to the SPN construction, we then aggregate the weights over the temporal sequence of matches. Thus, the links in the aggregated network will have final weights defined by Equation (2). Conversely to the SPN, the only weights $w_{p,t,i}$ in the LPN being equal to zero are those corresponding to all matches previous to the first one in which p and t co-play. The final weights of the Long-term Performance Network are $w_{p,t} \in [-0.54, 1.06]$.

As we can notice, the range of weights of SPN is close to the one found in LPN. However, these two weight formulations lead not only to different ranges of values but also to a different ranking of the links in the networks. When computing the Kendall's tau coefficient between the ranking of the links in the SPN and LPN, we find indeed that the two networks have a positive correlation ($\tau = 0.77$ with p-value $< 10^{-3}$) but the weights' ranking is changed. As our aim is to generate a recommending system for each player based on these weights, we further investigate the differences between the performance networks, by computing the Kendall's tau coefficient over each player's ranking. **Figure 3** shows the distribution of the Kendall's tau coefficient computed by comparing each player's ranking in the SPN and LPN. In particular, we have that just a small portion of players have the same teammate's ranking in both networks, and that the 87.8% of the remaining players have different rankings for their top-10 teammates. The recommending system that we are going to design will then provide a different recommendation based on the two performance networks. On the one hand, when using the SPN the system will recommend a teammate that leads to an instant skill gain. As an example, this might be the case of a teammate that is good in coordinating the team but from which not necessarily the player learns how to improve his/her performance. On the other hand, when using the LPN the system will recommend a teammate that leads to an increasing skill gain over the next matches. Thus, even if the instant skill gain with a teammate is not high, the player could learn some effective strategies and increase his/her skill gain in the successive matches.

4.3. LCC and Network Properties

Given a co-play performance network (short-term or long-term), to carry out our performance prediction we have to take into account only the links in the network having reliable weights. If two players play together just few times, the confidence we have on the corresponding weight is low. For example, if two players are teammates just one time their final weight only depends on that unique instance, and thus might lead to biased results. To face this issue, we computed the distribution of the number of occurrences a couple of teammates play together in our network (shown in **Figure 4**) and set a threshold based on these values. In particular, we decided to retain only pairs that played more than 2 matches together.

Finally, as many node embedding methods require a connected network as input (Ahmed et al., 2017), we extract the Largest Connected Component (LCC) of the performance network, which will be used for the performance prediction and evaluation. The LCC include the same number of nodes and links for both the SPN and the LPN. In particular, it includes 38,563 nodes and 1,444,290 links. We compare the characteristics of the initial network and its LCC in **Table 1**.

5. PERFORMANCE PREDICTION

In the following, we test whether the co-play performance networks have intrinsic structures allowing us to predict

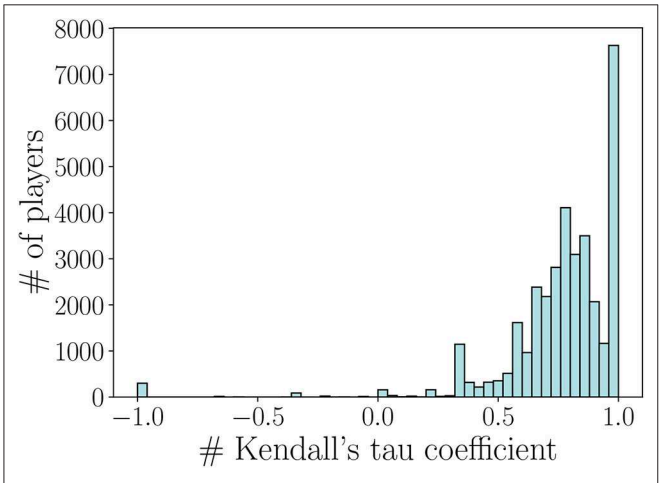

FIGURE 3 | Kendall's tau coefficient distribution computed by comparing each player's ranking in the short-term and long-term performance networks.

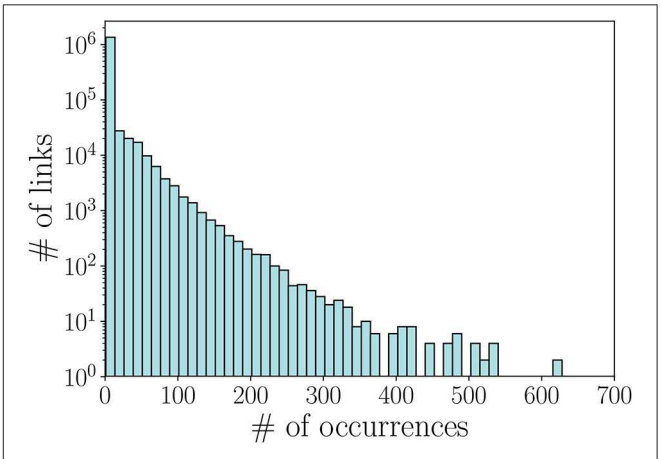

FIGURE 4 | Distribution of the number of occurrences per link, i.e., number of times a couple of teammates play together.

TABLE 1 | Comparison of the overall performance networks' characteristics and its LCC.

	# Nodes	# Links	SPN weights	LPN weights
Network	87,155	4,906,131	[−0.58, 1.06]	[−0.54, 1.06]
LCC	38,563	1,444,290	[−0.58, 1.06]	[−0.54, 1.06]

Note that the number of nodes and links are the same for both the Short-term Performance Network (SPN) and the Long-term Performance Network (LPN), while the range of weights varies from one case to the other.

performance of players when matched with unknown teammates. Such a prediction, if possible, could help us in recommending teammates to a player in a way that would maximize his/her skill improvement.

5.1. Problem Formulation

Consider the co-play performance network $G = (V, E)$ with weighted adjacency matrix W. A weighted link (i, j, w_{ij}) denotes that player i gets a performance variation of w_{ij} after playing with player j. We can formulate the recommendation problem as follows. Given an observed instance of a co-play performance network $G = (V, E)$ we want to predict the weight of each unobserved link $(i, j) \notin E$ and use this result to further predict the ranking of all other players $j \in V (\neq i)$ for each player $i \in V$.

5.2. Network Modeling

Does the co-play performance network contain information or patterns which can be indicative of skill gain for unseen pairs of players? If that is the case, how do we model the network structure to find such patterns? Are such patterns best represented via deep neural networks or more traditional factorization techniques?

To answer the above questions, we modify a deep neural network autoencoder and we test its predictive power against two classes of approaches widely applied in recommendation systems: (a) factorization based (Koren et al., 2009; Su and Khoshgoftaar, 2009; Ahmed et al., 2013), and (b) deep neural network based (Cao et al., 2016; Kipf and Welling, 2016; Wang et al., 2016). Note that the deep neural network based approaches on recommendations use different variations of deep autoencoders to learn a low-dimensional manifold to capture the inherent structure of the data. More recently, variational autoencoders have been tested for this task and have been shown to slightly improve the performance over traditional autoencoders (Kipf and Welling, 2016). In this paper, we focus on understanding the importance of applying neural network techniques instead of factorization models which are traditionally used in recommendation tasks and subtle variations in the autoencoder architecture to further improve performance is left as future work.

5.2.1. Factorization

In a factorization based model for directed networks, the goal is to obtain two low-dimensional matrices $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{n\times d}$ with number of hidden dimensions d such that the following function is minimized:

$$f(U, V) = \sum_{(i,j)\in E} (w_{ij} - < \boldsymbol{u_i}, \boldsymbol{v_j} >)^2 + \frac{\lambda}{2}(\|\boldsymbol{u_i}\|^2 + \|\boldsymbol{v_j}\|^2) \quad (4)$$

The sum in (4) is computed over the observed links to avoid of penalizing the unobserved one as overfitting to 0s would deter predictions. Here, λ is chosen as a regularization parameter to give preference to simpler models for better generalization.

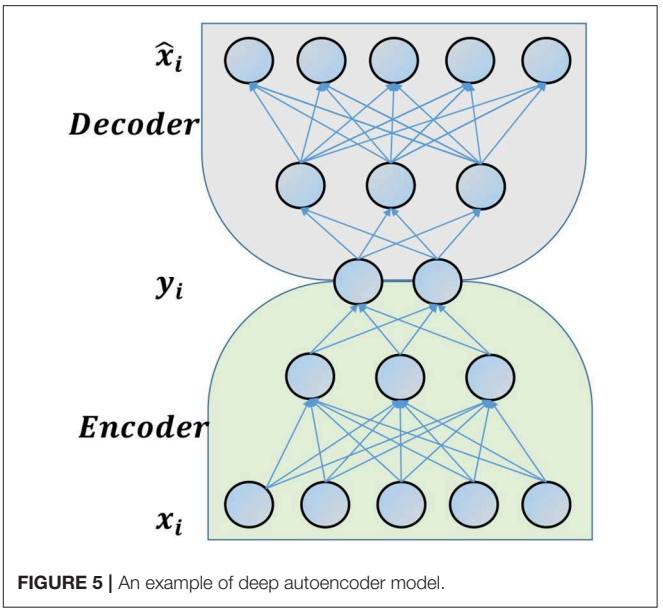

FIGURE 5 | An example of deep autoencoder model.

5.2.2. Traditional Autoencoder

Autoencoders are unsupervised neural networks that aim at minimizing the loss between reconstructed and input vectors. A traditional autoencoder is composed of two parts(cf., **Figure 5**): (a) an encoder, which maps the input vector into low-dimensional latent variables; and, (b) a decoder, which maps the latent variables to an output vector. The reconstruction loss can be written as:

$$L = \sum_{i=1}^{n} \|(\hat{\boldsymbol{x}}_i - \boldsymbol{x_i})\|_2^2, \quad (5)$$

where $\boldsymbol{x_i}$s are the inputs and $\hat{\boldsymbol{x}}_i = f(g(\boldsymbol{x_i}))$. $f(.)$ and $g(.)$ are the decoder and encoder functions respectively. Deep autoencoders have recently been adapted to the network setting (Cao et al., 2016; Kipf and Welling, 2016; Wang et al., 2016). An algorithm proposed by Wang et al. (2016) jointly optimizes the autoencoder reconstruction error and Laplacian Eigenmaps (Belkin and Niyogi, 2001) error to learn representation for undirected networks. However, this "Traditional Autoencoder" equally penalizes observed and unobserved links in the network, while the model adapted to the network setting cannot be applied when the network is directed. Thus, we propose to modify the Traditional Autoencoder model as follows.

Algorithm 1: Teammate Autoencoder

Function *CreatePerformanceNetwork (Player histories* $\{M_p\}$*, TrueSkill scores* $\{TS_p\}$*, NetworkType* $\{SPN, LPN\}$*)*
- **for** $p = 1 \dots P$ **do**
  - $\Delta TS_p = TS_p[0:] - TS_p[:-1]$ // compute the TrueSkill gains of p
  - **for** $i = 0 \dots N$ **do**
    - **if** *NetworkType is SPN* **then**
      - $W[p,t] = W[p,t] + \Delta TS_p[i]$ for each $t \in M_p[i]$ // compute the weight for each (p,t)
    - **else**
      - $idx \leftarrow$ *getLastSeenIndex*() // get the index of the last match played by p with t
      - $W[p,t] = W[p,t] + exp(i - idx)\Delta TS_p[i]$ for each $t \in M_p[i]$ // compute the weight for each (p,t)
      - *updateLastSeenIndex*(p,t,i) // update index
- return W

Function *Teammate Autoencoder (Player history* $\{M_p\}$*, TrueSkill scores* $\{TS_p\}$*)*
- $W \leftarrow$ CreatePerformanceNetwork($\{M_p\}$,$\{TS_p\}$) // create performance network using player history and TrueSkill scores
- $\vartheta \leftarrow$ RandomInit() // initialize the autoencoder weights
- Set $\mathcal{F} = \{(w_i)\}$ for each $w_i \in W$ // construct input set for autoencoder from weight matrix
- **for** $iter = 1 \dots I$ **do**
  - Randomly sample minibatch M from $\mathcal{F}$ // get minibatch
  - $L = \sum_{i=1}^{n} \|(\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}}) \odot [a_{i,j}]_{j=1}^{n}\|_2^2$ // compute loss function
  - $grad \leftarrow \partial L / \partial \vartheta$ // compute gradient
  - $\vartheta \leftarrow$ UpdateGradSGD$(\vartheta, grad)$ // upgrade model weights using the gradient
- $Y \leftarrow$ EncoderForwardPass(G, ϑ) // compute embedding using encoder
- return Y

5.2.3. Teammate Autoencoder

To model directed networks, we propose a modification of the Traditional Autoencoder model, that takes into account the adjacency matrix representing the directed network. Moreover, in this formulation we only penalize the observed links in the network, as our aim is to predict the weight and the corresponding ranking of the unobserved links. We then write our "Teammate Autoencoder" reconstruction loss as:

$$L = \sum_{i=1}^{n} \|(\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}}) \odot [a_{i,j}]_{j=1}^{n}\|_2^2, \tag{6}$$

where $a_{ij} = 1$ if $(i,j) \in E$, and 0 otherwise. Here, x_i represents i^{th} row of the adjacency matrix and n is the number of nodes in the network. Thus, the model takes each row of the adjacency matrix representing the performance network as input and outputs an embedding for each player such that it can reconstruct the observed edges well. For example, if there are 3 players and player 2 helps improve player 1's performance by a factor of α, player 1's row would be $[0, \alpha, 0]$. We train the model by minimizing the above loss function using stochastic gradient descent and calculate the gradients using back propagation. Minimizing this loss functions yields the neural network weights W and the learned representation of the network $Y \in \mathbb{R}^{n \times d}$. The layers in the neural network, the activation function and regularization coefficients serve as the hyperparameters in this model. Algorithm 1 summarizes our methodology.

5.3. Evaluation Framework

5.3.1. Experimental Setting

To evaluate the performance of the models on the task of teammates' recommendation, we use the cross-validation framework illustrated in **Figure 6**. We randomly "hide" 20% of the weighted links and use the rest of the network to learn the embedding, i.e., representation, of each player in the network. We then use each player's embedding to predict the weights of the unobserved links. As the number of player pairs is too large, we evaluate the models on multiple samples of the co-player performance networks [similar to Ou et al. (2016); Goyal and Ferrara (2018)] and report the mean and standard deviation of the used metrics. Instead of uniformly sampling the players as performed in Ou et al. (2016); Goyal and Ferrara (2018), we use random walks (Backstrom and Leskovec, 2011) with random restarts to generate sampled networks with similar degree and weight distributions as the original network. **Figure 7** illustrates these distributions for the sampled network of 1,024 players (nodes).

Further, we obtain the optimal hyperparater values of the models used using a grid search over a set of values. For Graph Factorization, we vary the regularization coefficient in powers of 10, $\lambda \in [10^{-5}, 1]$. For deep neural network based models, we use *ReLU* as the activation function and choose the neural network structure by an informal search over a set of architectures. We set the l_1 and l_2 regularization coefficients by performing grid search on $[10^{-5}, 10^{-1}]$.

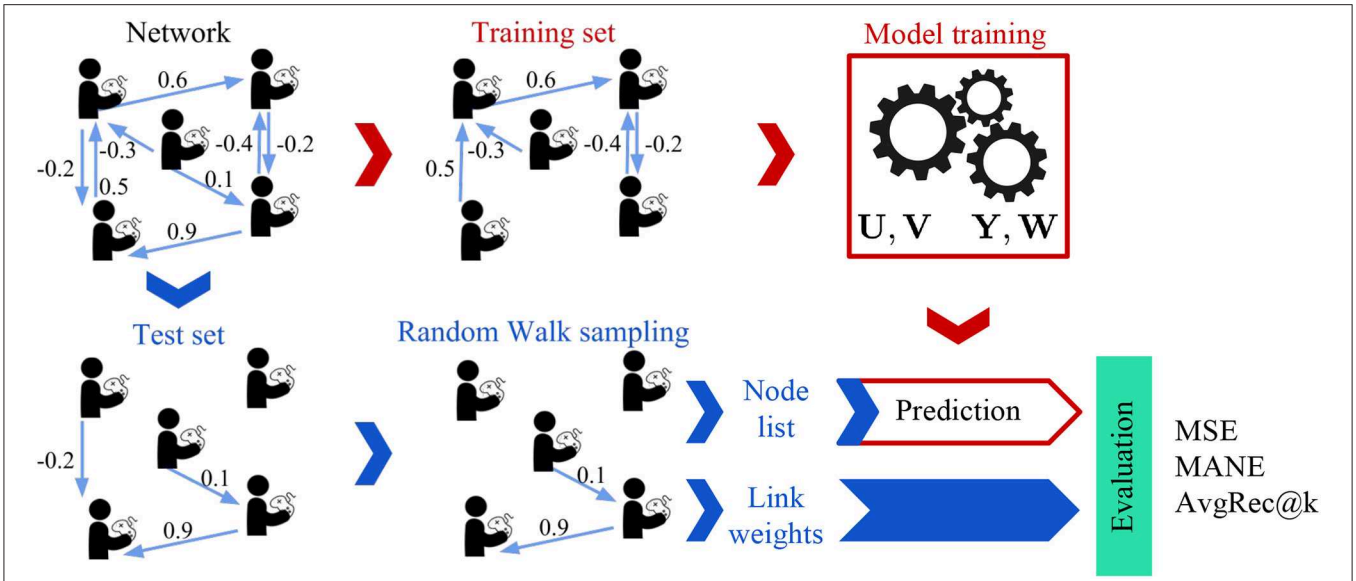

FIGURE 6 | Evaluation Framework: The co-play network is divided into training and test networks. The parameters of the models are learned using the training network. We obtain multiple test subnetworks by using a random walk sampling with random restart and input the nodes of these subnetworks to the models for prediction. The predicted weights are then evaluated against the test link weights to obtain various metrics.

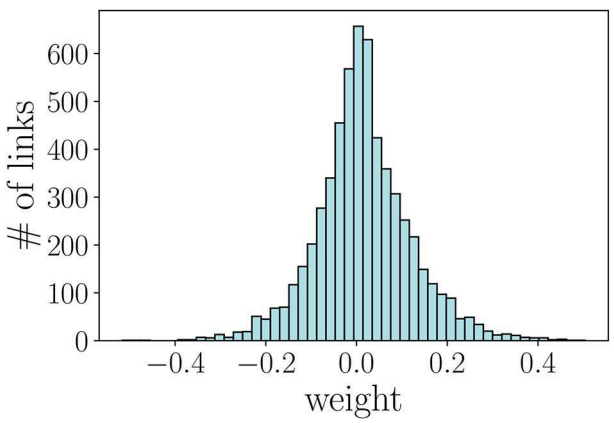

FIGURE 7 | Distribution of the weights of the network sampled by using random walk.

5.3.2. Evaluation Metrics

We use Mean Squared Error (*MSE*), Mean Absolute Normalized Error (*MANE*), and *AvgRec@k* as evaluation metrics. *MSE* evaluates the accuracy of the predicted weights, whereas *MANE* and *AvgRec@k* evaluate the ranking obtained by the model.

First, we compute *MSE*, typically used in recommendation systems, to evaluate the error in the prediction of weights. We use the following formula for our problem:

$$MSE = \|\boldsymbol{w}^{test} - \boldsymbol{w}^{pred}\|^2, \tag{7}$$

where $\boldsymbol{w}_{test}$ is the list of weights of links in the test subnetwork, and $\boldsymbol{w}_{pred}$ is the list of weights predicted by the model. Thus, *MSE* computes how well the model can predict the weights of the network. A lower value implies better prediction.

Second, we use *AvgRec@k* to evaluate the ranking of the weights in the overall network. It is defined as:

$$AvgRec@k = \frac{\sum_{i=1}^{k} \boldsymbol{w}^{test}_{index(i)}}{k}, \tag{8}$$

where $index(i)$ is the index of the i^{th} highest predicted link in the test network. It computes the average gain in performance for top k recommendations. A higher values implies the model can make better recommendations.

Finally, to test the models' recommendations for each player, we define the Mean Absolute Normalized Error (*MANE*), which computes the normalized difference between predicted and actual ranking of the test links among the observed links and averages over the nodes. Formally, it can be written as:

$$MANE(i) = \frac{\sum_{j=1}^{|E_i^{test}|} \left| rank_i^{pred}(j) - rank_i^{test}(j) \right|}{|E_i^{train}||E_i^{test}|},$$
$$MANE = \frac{\sum_{i=1}^{|V|} MANE(i)}{|V|}, \tag{9}$$

where $rank_i^{pred}(j)$ represents the rank of the j^{th} vertex in the list of weights predicted for the player i. A lower *MANE* value implies that the ranking of recommended players is similar to the actual ranking according to the test set.

5.4. Results and Analysis

In the following, we evaluate the results provided by the Graph Factorization, the Traditional Autoencoder and our Teammate Autoencoder. To this aim we first analyze the models' performance on both the SPN and the LPN with respect to the MSE measure, computed in Equation (7), respectively in

Figures 8A, **9A**. In this case, we compare the models against an "average" baseline, where we compute the average performance of the players' couples observed in the training set and use it as a prediction for each hidden teammate link.

Figures 8A, **9A** show the variation of the percentage of the *MSE* gain (average and standard deviation) while increasing the number of latent dimensions d for in each model. We can observe that the Graph Factorization model generally performs worse than the baseline, with values in $[-1.64\%, -0.56\%]$ and average of -1.2% for the SPN and values in $[-1.35\%, -0.74\%]$ and average of -1.05% for the LPN. This suggests that the performance networks of Dota 2 require the use of deep neural networks to capture their underlying structure. However, a traditional model is not enough to outperform the baseline. The Traditional Autoencoder reaches indeed marginal improvements: values in $[0.0\%, 0.55\%]$ and average gain of 0.18% for the SPN; values in $[0.0\%, 0.51\%]$ and average gain of 0.20% for the LPN. On the contrast, our Teammate Autoencoder achieves substantial gain over the baseline across the whole spectrum and its performance in general increases for higher dimensions (they can retain more structural information). The average MSE gain for different dimensions over the baseline of the Teammate Autoencoder spans between 6.34% and 11.06% in the SPN and from 6.68% to 11.34% for the LPN, with an average gain over all dimensions of 9.00% for the SPN and 9.15% for the LPN. We also computed the MSE average over 10 runs and $d = 1,024$, shown in **Table 2**, which decreases from the baseline prediction of 4.55 to our Teammate Autoencoder prediction of 4.15 for the SPN, and from 4.40 to 3.91 for the LPN.

We then compare the models' performance in providing individual recommendations by analyzing the *MANE* metric, computed in Equation (9). **Figures 8B**, **9B** show the percentage of the *MANE* gain for different dimensions computed against the average baseline respectively for the SPN and the LPN. Analogously to the *MSE* case, the Graph Factorization performs worse than the baseline (values in $[-3.34\%, -1.48\%]$ with average gain of -2.37% for SPN and values in $[-3.78\%, -0.78\%]$ -2.79% for LPN) despite the increment in the number of dimensions. The Traditional Autoencoder achieves marginal gain over the baseline for dimensions higher than 128 ($[0.0\%, 0.37\%]$ for SPN and $[0.0\%, 0.5\%]$ for LPN), with an average gain over all dimensions of 0.16% for SPN and 0.19% for LPN. Our model attains instead significant percentage gain in individual recommendations over the baseline. For the SPN, it achieves an average percentage of *MANE* gain spanning from 14.81 to 22.78%, with an overall average of 19.50%. For the LPN, the average percentage of *MANE* gain spans from 16.81 to 22.32%, with an overall average of 19.29%. It is worth noting that the performance in this case does not monotonically increase with dimensions. This might imply that for individual recommendations the model overfits at higher dimensions. We report the average value of *MANE* in **Table 2** for $d = 1,024$. Our model obtains average values of 0.059 and 0.062, for the SPN and LPN respectively, compared to 0.078 of the average baseline for both cases.

Finally, we compare our models against the ideal recommendation in the test subnetwork to understand how close our top recommendations are to the ground truth. To this aim, we report the *AvgRec@k* metric, which computes the average weight of the top k links recommended by the models as in Equation (8). In **Figures 8C**, **9C**, we can observe that the Teammate Autoencoder significantly outperforms the other models, both for the SPN and LPN respectively. The theoretical maximum line shows the *AvgRec@k* values by selecting the top k recommendations for the entire network using the test set. For the SPN, the link with the highest predicted weight by our model achieves a performance gain of 0.38 as opposed to 0.1 for Graph Factorization. This gain is close to the ideal prediction which achieves 0.52. For the LPN, instead, our model achieves a performance gain of 0.3 as opposed to 0.1 for Graph Factorization. The performance of our model remains higher for all values of k. This shows that the ranking of the links achieved by our model is close to the ideal ranking. Note that the Traditional Autoencoder yields poor performance on this task which signifies the importance of relative weighting of observed and unobserved links.

6. RELATED WORK

There is a broad body of research focusing on online games to identify which characteristics influence different facets of human behaviors. On the one hand, this research is focused on the cognitive aspects that are triggered and affected when playing online games, including but not limited to gamer motivations to play (Choi and Kim, 2004; Yee, 2006; Jansz and Tanis, 2007; Tyack et al., 2016), learning mechanisms (Steinkuehler, 2004, 2005), and player performance and acquisition of expertise (Schrader and McCreery, 2008). On the other hand, players and their performance are classified in terms of in-game specifics, such as combat patterns (Drachen et al., 2014; Yang et al., 2014), roles (Eggert et al., 2015; Lee and Ramler, 2015; Sapienza et al., 2017), and actions (Johnson et al., 2015; Xia et al., 2017; Sapienza et al., 2018a).

Aside from these different gaming features, multiplayer online games especially distinguish from other games because of their inherent cooperative design. In such games, players have not only to learn individual strategies, but also to organize and coordinate to reach better results. This intrinsic social aspect has been a focal research topic (Ducheneaut et al., 2006; Hudson and Cairns, 2014; Losup et al., 2014; Schlauch and Zweig, 2015; Tyack et al., 2016). In Cole and Griffiths (2007), authors show that multiplayer online games provide an environment in which social interactions among players can evolve into strong friendship relationships. Moreover, the study shows how the social aspect of online gaming is a strong component for players to enjoy the game. Another study (Pobiedina et al., 2013a,b) ranked different factors that influence player performance in MOBA games. Among these factors, the number of friends resulted to have a key role in a successful teams. In the present work, we focused on social contacts at a higher level: co-play relations. Teammates, either friends or strangers, can affect other players' styles through communication, by trying to exert influence over others, etc. (Kou and Gui, 2014; Leavitt et al., 2016; Zeng et al.,

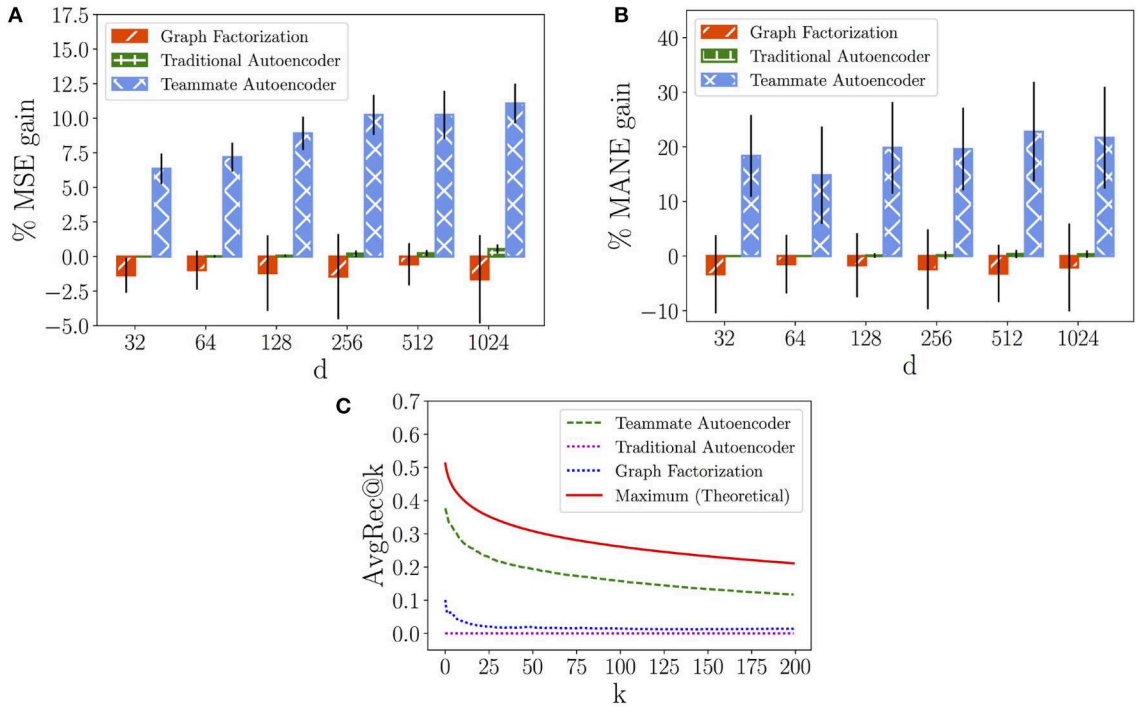

FIGURE 8 | Short-term Performance Network. **(A)** Mean Squared Error (*MSE*) gain of models over average prediction. **(B)** Mean Absolute Normalized Error (*MANE*) gain of models over average prediction. **(C)** *AvgRec@k* of models.

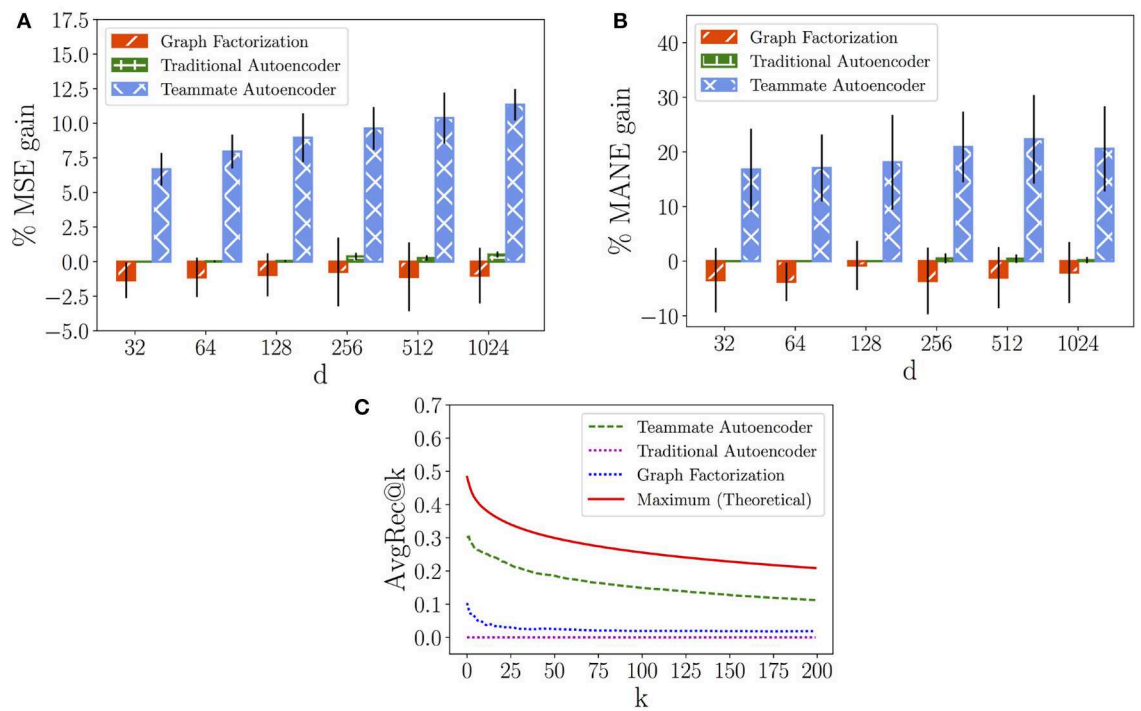

FIGURE 9 | Long-term Performance Network. **(A)** Mean Squared Error (*MSE*) gain of models over average prediction. **(B)** Mean Absolute Normalized Error (*MANE*) gain of models over average prediction. **(C)** *AvgRec@k* of models.

TABLE 2 | Average and standard deviation of player performance prediction (*MSE*) and teammate recommendation (*MANE*) for $d = 1,024$ in both SPN and LPN.

	MSE_{SPN}	$MANE_{SPN}$	MSE_{LPN}	$MANE_{LPN}$
Baseline prediction	4.55/0.14	0.078/0.02	4.40/0.14	0.078/0.01
Graph factorization	4.59/0.17	0.081/0.02	4.45/0.18	0.084/0.021
Traditional autoencoder	4.54/0.15	0.074/0.01	4.37/0.13	0.075/0.012
Teammate autoencoder	**4.15/0.14**	**0.059/0.008**	**3.91/0.10**	**0.062/0.008**

Bold values show the best performance for that metric.

2018). Moreover, we leveraged these teammate-related effects on player performance to build a teammate recommendation system for players in Dota 2.

Recommendation systems have been widely studied in the literature on applications such as movies, music, restaurants and grocery products (Lawrence et al., 2001; Lekakos and Caravelas, 2008; Van den Oord et al., 2013; Fu et al., 2014). The current work on such systems can be broadly categorized into: (i) collaborative filtering (Su and Khoshgoftaar, 2009; Kluver et al., 2018; Liang et al., 2018), (ii) content based filtering (Pazzani and Billsus, 2007; Shu et al., 2018), and (iii) hybrid models (Burke, 2002; Ji et al., 2019). Collaborative filtering is based on the premise that users with similar interests in the past will tend to agree in the future as well. Content based models learn the similarity between users and content descriptions. Hybrid models combine the strength of both of these systems with varying hybridization strategy.

In the specific case of MOBA games, recommendation systems are mainly designed to advise players on the type of character (hero) they impersonate[6] (Conley and Perry, 2013; Agarwala and Pearce, 2014; Chen et al., 2018). Few works addressed the problem of recommending teammates in MOBA games. In Van De Bovenkamp et al. (2013), authors discuss how to improve matchmaking for players based on the teammates they had in their past history. They focus on the creation and analysis of the properties of different networks in which the links are formed based on different rules, e.g., players that played together in the same match, in the same team, in adversarial teams, etc. These networks are then finally used to design a matchmaking algorithm to improve social cohesion between players. However, the author focus on different relationships to build their networks and on the strength of network links to design their algorithm, while no information about the actual player performance is taken into account. We here aim at combining both the presence of players in the same team (and the number of times they play together) and the effect that these combinations have on player performance, by looking at skill gain/loss after the game.

7. CONCLUSIONS

In this paper, we set to study the complex interplay between cooperation, teams and teammates' recommendation, and players' performance in online games. Our study tackled three specific problems: (i) understanding short and long-term teammates' influence on players' performance; (ii) recommending teammates with the aim of improving players skills and performance; and (iii) demonstrating a deep neural network that can predict such performance improvements.

We used Dota 2, a popular Multiplayer Online Battle Arena game hosting millions of players and matches every day, as a virtual laboratory to understand performance and influence of teammates. We used our dataset to build a co-play network of players, with weights representing a teammate's short-term influence on a player performance. We also developed a variant of this weighting algorithm that incorporates a memory mechanism, implementing the assumption that player's performance and skill improvements carry over in future games (i.e., long-term influence): influence can be intended as a longitudinal process that can improve or hinder player's performance improvement over time.

With this framework in place, we demonstrated the feasibility of a recommendation system that suggests new teammates, which can be beneficial to a player to play with to improve their individual performance. This system, based on a modified autoencoder model, yields state-of-the-art recommendation accuracy, outperforming graph factorization techniques considered among the best in recommendation systems literature, closing the existing gap with the maximum improvement that is theoretically achievable. Our experimental results suggest that skill transfer and performance improvement can be accurately predicted with deep neural networks.

We plan to extend this work in multiple future directions. First, our current framework takes only into account the individual skill of players to recommend teammates that are indeed beneficial to improve a player's performance in the game. However, multiple aspects of the game can play a key role in influencing individual performance. These are aspects such as the impersonated role, the presence of friends or strangers in the team, cognitive budget of players, and their personality. Thus, we are planning to extend our current framework to take into account these aspects of the game and train a model that recommends teammates on the basis of these multiple factors.

Second, from a theoretical standpoint, we intend to determine whether our framework can be generalized to generate recommendations and predict team and individual performance in a broader range of scenarios, beyond online games. We will explore whether more sophisticated factorization techniques based on tensors, rather than matrices, can be leveraged within our framework, as such techniques have recently shown promising results in human behavioral modeling (Hosseinmardi et al., 2018a,b; Sapienza et al., 2018a). We also plan to demonstrate, from an empirical standpoint, that the recommendations produced by our system can be implemented in real settings. We will carry out randomized-control trials in lab settings to test whether individual performance in teamwork-based tasks can be improved. One additional direction will be to extend our framework to recommend incentives alongside teammates: this to establish whether we can computationally suggest

[6]DotaPicker: http://dotapicker.com/

incentive-based strategies to further motivate individuals and improve their performance within teams.

AUTHOR CONTRIBUTIONS

AS, PG, and EF designed the research framework, analyzed the results, wrote, and reviewed the manuscript. AS and PG collected the data and performed the experiments.

FUNDING

The authors are grateful to DARPA for support (grant #D16AP00115). This project does not necessarily reflect the position/policy of the Government; no official endorsement should be inferred. Approved for public release; unlimited distribution.

REFERENCES

Agarwala, A. and Pearce, M. (2014). *Learning Dota 2 Team Compositions*. Technical report, Technical report, Stanford University.

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., and Smola, A. J. (2013). "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd International Conference on World Wide Web* (New York, NY: ACM), 37–48. doi: 10.1145/2488388.2488393

Ahmed, N. K., Rossi, R. A., Zhou, R., Lee, J. B., Kong, X., Willke, T. L., et al. (2017). A framework for generalizing graph-based representation learning methods. *arXiv* arXiv:1709.04596

Backstrom, L., and Leskovec, J. (2011). "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (New York, NY: ACM), 635–644.

Battistich, V., Solomon, D., and Delucchi, K. (1993). Interaction processes and student outcomes in cooperative learning groups. *Element. School J.* 94, 19–32. doi: 10.1086/461748

Beersma, B., Hollenbeck, J. R., Humphrey, S. E., Moon, H., Conlon, D. E., and Ilgen, D. R. (2003). Cooperation, competition, and team performance: toward a contingency approach. *Acad. Manage. J.* 46, 572–590. doi: 10.5465/30040650

Belkin, M., and Niyogi, P. (2001). "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, eds T. G. Dietterich, S. Becker, and Z. Ghahramani (Cambridge, MA: MIT Press), 585–591.

Burke, R. (2002). Hybrid recommender systems: survey and experiments. *User Model. User Adapt. Inter.* 12, 331–370. doi: 10.1023/A:1021240730564

Cao, S., Lu, W., and Xu, Q. (2016). "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Menlo Park, CA: AAAI Press), 1145–1152.

Chen, Z., Nguyen, T.-H. D., Xu, Y., Amato, C., Cooper, S., Sun, Y., et al. (2018). The art of drafting: a team-oriented hero recommendation system for multiplayer online battle arena games. in *Proceedings of the 12th ACM Conference on Recommender Systems* (New York, NY: ACM), 200–208.

Childress, M. D., and Braswell, R. (2006). Using massively multiplayer online role-playing games for online learning. *Dist. Educ.* 27, 187–196. doi: 10.1080/01587910600789522

Choi, D., and Kim, J. (2004). Why people continue to play online games: In search of critical design factors to increase customer loyalty to online contents. *CyberPsychol. Behav.* 7, 11–24. doi: 10.1089/109493104322820066

Cohen, E. G. (1994). Restructuring the classroom: conditions for productive small groups. *Rev. Educ. Res.* 64, 1–35. doi: 10.3102/00346543064001001

Cole, H., and Griffiths, M. D. (2007). Social interactions in massively multiplayer online role-playing gamers. *CyberPsychol. Behav.* 10, 575–583. doi: 10.1089/cpb.2007.9988

Conley, K., and Perry, D. (2013). *How Does He Saw Me? A Recommendation Engine for Picking Heroes in Dota 2*. Np Web, 7.

Deutsch, M. (1960). *The Effects of Cooperation and Competition Upon Group Process*. Cambridge, MA: Group Dynamics, 552–576.

Drachen, A., Yancey, M., Maguire, J., Chu, D., Wang, I. Y., Mahlmann, T., et al. (2014). "Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2)," in *Games Media Entertainment (GEM), 2014 IEEE* (Toronto, ON: IEEE), 1–8. doi: 10.1109/GEM.2014.7048109

Ducheneaut, N., Yee, N., Nickell, E., and Moore, R. J. (2006). "Alone together?: exploring the social dynamics of massively multiplayer online games," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY: ACM), 407–416.

Eggert, C., Herrlich, M., Smeddinck, J., and Malaka, R. (2015). "Classification of player roles in the team-based multi-player game dota 2," in *International Conference on Entertainment Computing* (Trondheim: Springer), 112–125. doi: 10.1007/978-3-319-24589-8-9

Elo, A. E. (1978). *The Rating of Chessplayers, Past and Present*. New York, NY: Arco Pub.

Fox, J., Gilbert, M., and Tang, W. Y. (2018). Player experiences in a massively multiplayer online game: a diary study of performance, motivation, and social interaction. *New Media Soc.* 20, 4056–4073. doi: 10.1177/1461444818767102

Fu, Y., Liu, B., Ge, Y., Yao, Z., and Xiong, H. (2014). "User preference learning with multiple information fusion for restaurant recommendation," in *Proceedings of the 2014 SIAM International Conference on Data Mining* (Philadelphia, PA: SIAM), 470–478. doi: 10.1137/1.9781611973440.54

Goyal, P., and Ferrara, E. (2018). Graph embedding techniques, applications, and performance: a survey. *Knowl. Based Syst.* 151, 78–94. doi: 10.1016/j.knosys.2018.03.022

Herbrich, R., Minka, T., and Graepel, T. (2007). "Trueskill: a bayesian skill rating system," in *Advances in Neural Information Processing Systems*, eds B. Schölkopf, J. C. Platt, and T. Hoffman (Cambridge, MA: MIT Press), 569–576.

Hosseinmardi, H., Ghasemian, A., Narayanan, S., Lerman, K., and Ferrara, E. (2018a). Tensor embedding: a supervised framework for human behavioral data mining and prediction. *arXiv*. arXiv:1808.10867

Hosseinmardi, H., Kao, H.-T., Lerman, K., and Ferrara, E. (2018b). "Discovering hidden structure in high dimensional human behavioral data via tensor factorization," in *WSDM Heteronam Workshop* (Los Angeles, CA).

Hudson, M., and Cairns, P. (2014). "Measuring social presence in team-based digital games," in *Interacting With Presence: HCI and the Sense of Presence in Computer-Mediated Environments* (Berlin), 83.

Jansz, J., and Tanis, M. (2007). Appeal of playing online first person shooter games. *CyberPsychol. Behav.* 10, 133–136. doi: 10.1089/cpb.2006.9981

Ji, Z., Pi, H., Wei, W., Xiong, B., Wozniak, M., and Damasevicius, R. (2019). *Recommendation Based on Review Texts and Social Communities: A Hybrid Model*. Adelaide, SA: IEEE Access.

Johnson, D., Nacke, L. E., and Wyeth, P. (2015). "All about that base: differing player experiences in video game genres and the unique case of moba games," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (New York, NY: ACM) ,2265–2274. doi: 10.1145/2702123.2702447

Johnson, D. W., and Johnson, R. T. (1989). *Cooperation and Competition: Theory and Research*. Edina, MN: Interaction Book Company.

Johnson, D. W., Maruyama, G., Johnson, R., Nelson, D., and Skon, L. (1981). Effects of cooperative, competitive, and individualistic goal structures on achievement: a meta-analysis. *Psychol. Bull.* 89:47. doi: 10.1037//0033-2909.89.1.47

Kipf, T. N., and Welling, M. (2016). Variational graph auto-encoders. *arXiv*. arXiv:1611.07308

Kluver, D., Ekstrand, M. D., and Konstan, J. A. (2018). "Rating-based collaborative filtering: algorithms and evaluation," in *Social Information Access*, eds P. Brusilovsky and D. He (Cham: Springer International Publishing), 344–390.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer* 42, 30–37. doi: 10.1109/MC.2009.263

Kou, Y., and Gui, X. (2014). "Playing with strangers: understanding temporary teams in league of legends," in *Proceedings of the First ACM SIGCHI Annual*

Symposium on Computer-Human Interaction in Play (New York, NY: ACM), 161–169.

Lawrence, R. D., Almasi, G. S., Kotlyar, V., Viveros, M., and Duri, S. S. (2001). "Personalization of supermarket product recommendations," in *Applications of Data Mining to Electronic Commerce* (Boston, MA: Springer), 11–32.

Leavitt, A., Keegan, B. C., and Clark, J. (2016). "Ping to win?: non-verbal communication and team performance in competitive online multiplayer games," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (New York, NY: ACM), 4337–4350.

Lee, C.-S., and Ramler, I. (2015). "Investigating the impact of game features and content on champion usage in league of legends," in *Proceedings of the Foundation of Digital Games* (Grove, CA).

Lekakos, G., and Caravelas, P. (2008). A hybrid approach for movie recommendation. *Multi. Tool. Appl.* 36, 55–70. doi: 10.1007/s11042-006-0082-7

Levi, D. (2015). *Group Dynamics for Teams*. Thousand Oaks, CA: Sage Publications.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. (2018). "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web* (Lyon: International World Wide Web Conferences Steering Committee), 689–698. doi: 10.1145/3178876.3186150

Losup, A., Van De Bovenkamp, R., Shen, S., Jia, A. L., and Kuipers, F. (2014). Analyzing implicit social networks in multiplayer online games. *IEEE Int. Comput.* 18, 36–44. doi: 10.1109/MIC.2014.19

Morschheuser, B., Hamari, J., and Maedche, A. (2018). Cooperation or competition–when do people contribute more? a field experiment on gamification of crowdsourcing. *Int. J. Hum. Comput. Stud.* 124, 7–24. doi: 10.1016/j.ijhcs.2018.10.001

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY: ACM), 1105–1114.

Park, H., and Kim, K.-J. (2014). "Social network analysis of high-level players in multiplayer online battle arena game," in *International Conference on Social Informatics* (Cham: Springer), 223–226.

Pazzani, M. J., and Billsus, D. (2007). "Content-based recommendation systems," in *The Adaptive Web* (Berlin; Heidelberg: Springer), 325–341.

Pobiedina, N., Neidhardt, J., Calatrava Moreno, M. d. C., and Werthner, H. (2013a). "Ranking factors of team success," in *Proceedings of the 22nd International Conference on World Wide Web* (New York, NY: ACM), 1185–1194.

Pobiedina, N., Neidhardt, J., Moreno, M. C., Grad-Gyenge, L., and Werthner, H. (2013b). *On Successful Team Formation*. Technical report, Technical report, Vienna University of Technology, 2013. Available online at: http://www.ec.tuwien.ac.at/files/OnSuccessfulTeamFormation.pdf (accessed May 30, 2019).

Sapienza, A., Bessi, A., and Ferrara, E. (2018a). Non-negative tensor factorization for human behavioral pattern mining in online games. *Information* 9:66. doi: 10.3390/info9030066

Sapienza, A., Peng, H., and Ferrara, E. (2017). "Performance dynamics and success in online games," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (New Orleans, LA: IEEE), 902–909.

Sapienza, A., Zeng, Y., Bessi, A., Lerman, K., and Ferrara, E. (2018b). Individual performance in team-based online games. *R. Soc. Open Sci.* 5:180329. doi: 10.1098/rsos.180329

Schlauch, W. E., and Zweig, K. A. (2015). "Social network analysis and gaming: survey of the current state of art," in *Joint International Conference on Serious Games* (Cham: Springer), 158–169. doi: 10.1007/978-3-319-19126-3_14

Schrader, P., and McCreery, M. (2008). The acquisition of skill and expertise in massively multiplayer online games. *Educ. Tech. Res. Dev.* 56, 557–574. doi: 10.1007/s11423-007-9055-4

Shu, J., Shen, X., Liu, H., Yi, B., and Zhang, Z. (2018). A content-based recommendation algorithm for learning resources. *Multi. Syst.* 24, 163–173. doi: 10.1007/s00530-017-0539-8

Steinkuehler, C. A. (2004). "Learning in massively multiplayer online games," in *Proceedings of the 6th International Conference on Learning Sciences* (International Society of the Learning Sciences) (Santa Monica, CA), 521–528.

Steinkuehler, C. A. (2005). *Cognition and Learning in Massively Multiplayer Online Games: A Critical Approach*. Madison, WI: The University of Wisconsin-Madison.

Su, X., and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Adv. Art. Intel.* 2009:4. doi: 10.1155/2009/421425

Tauer, J. M., and Harackiewicz, J. M. (2004). The effects of cooperation and competition on intrinsic motivation and performance. *J. Person. Soc. Psychol.* 86:849. doi: 10.1037/0022-3514.86.6.849

Tyack, A., Wyeth, P., and Johnson, D. (2016). "The appeal of moba games: What makes people start, stay, and stop," in *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play* (New York ,NY: ACM), 313–325.

Van De Bovenkamp, R., Shen, S., Iosup, A., and Kuipers, F. (2013). "Understanding and recommending play relationships in online social gaming," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)* (Bangalore: IEEE), 1–10.

Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, eds C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Lake Tahoe: Curran Associates, Inc.), 2643–2651.

Wang, D., Cui, P., and Zhu, W. (2016). "Structural deep network embedding," in *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining* (New York ,NY: ACM), 1225–1234.

Xia, B., Wang, H., and Zhou, R. (2017). What contributes to success in moba games? an empirical study of defense of the ancients 2. *Games Cult.* 14, 498–522. doi: 10.1177/1555412017710599

Yang, P., Harrison, B. E., and Roberts, D. L. (2014). "Identifying patterns in combat that are predictive of success in moba games," in *FDG*, eds T. Barnes and I. Bogost (Santa Cruz, CA: Society for the Advancement of the Science of Digital Games), 281–288.

Yee, N. (2006). Motivations for play in online games. *CyberPsychol. Behav.* 9, 772–775. doi: 10.1089/cpb.2006.9.772

Zeng, Y., Sapienza, A., and Ferrara, E. (2018). The influence of social ties on performance in team-based online games. *arXiv* arXiv:1812.02272

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.