Learning Agent Representations for Ice Hockey

Guiliang Liu¹, Oliver Schulte¹,³, Pascal Poupart²,⁴, Mike Rudd²,⁴, Mehrsan Javan³

¹School of Computing Science, Simon Fraser University
²Cheriton School of Computer Science, University of Waterloo
³SLiQ Lab, Sportlogiq
⁴Vector Institute

gli68@sfu.ca, oschulte@cs.sfu.ca
{ppoupart,jmrudd}@uwaterloo.ca, mehrsan@sportlogiq.com

Abstract

Team sports is a new application domain for agent modeling with high real-world impact. A fundamental challenge for modeling professional players is their large number (over 1K), which includes many bench players with sparse participation in a game season. The diversity and sparsity of player observations make it difficult to extend previous agent representation models to the sports domain. This paper develops a new approach for agent representations, based on a Markov game model, that is tailored towards applications in professional ice hockey. We introduce a novel framework player representation via player generation, where a variational encoder embeds player information with latent variables. The encoder learns a context-specific shared prior to induce a shrinkage effect for the posterior player representations, allowing it to share statistical information across players with different participation rates. To capture the complex play dynamics in sequential sports data, we design a Variational Recurrent Ladder Agent Encoder (VaRLAE). This architecture provides a contextualized player representation with a hierarchy of latent variables that effectively prevents latent posterior collapse. We validate our player representations in three major sports analytics tasks. Our experimental results, based on a large dataset that contains over 4.5M events, show state-of-the-art performance for our VarLAE on facilitating 1) identifying the acting player, 2) estimating expected goals, and 3) predicting the final score difference.

1 Introduction

Team sports define complex interactions with a rich action space and a heterogeneous state space [1]. As more and larger event stream datasets for professional sports matches become available, advanced machine learning algorithms have been applied to model the complex dynamics of sports games. These models facilitate many applications with high real-world impact, such as predicting match outcomes [2], assessing the performance of players or teams [3] [4] [5], and recommending tactics to coaches [7]. However, previous models often pool observations of different players without capturing the specific roles and behaviors of each athlete. Neglecting the special characteristics of individual players significantly compromises the model performance for the application tasks [8].

A promising approach to incorporating player information into sports statistics is deep agent representation. Learning agent representation in a sports game raises new challenges that require a novel solution for several reasons: 1) Previous methods commonly learn a representation of agents’ policy [9] [10] [11] [12] [13] [14]. The policy embedding methods model only how a player acts in a given match context, and thus fail to capture the statistical information about which match contexts players are likely to act in, as well as their immediate outcomes (rewards). 2) Although policy embeddings can facilitate the construction of an artificial agent in control problems, the goal of agent representation
in sports analytics is predicting game-relevant events using player identity information [8, 7]. 3) Previous agent representation models are often validated in virtual games instead of physical sports. They assume a limited number of agents (usually less than 10) and sufficient observations for each of them. In contrast, professional sports leagues often have upwards of one-thousand players, and many bench (backup) players play only a few (less than 20) games in a season. The large number of agents and unbalanced observations result in unstable embedding models that overfit to players with high participation during training, which undermines both the training convergence and the predictive performance of underlying policy representations [11].

This work describes a novel player representation framework that learns player representations via player generation. Conditioning on the current game context, the generation model predicts the distribution of the currently acting player. During this process, we learn a distribution of latent variables as the context-specific prior, allowing a neural encoder to derive an approximate posterior as a contextualized representation for the observed player. We train the encoder by maximizing an Evidence Lower Bound (ELBo, Equation (9)) that moves the posterior for each player toward the prior mode. This shrinkage effect [15] is ideal for player representations, because: 1) It allows information to be transferred between the observations of different players and draws closer the representations of players who share statistical similarities. 2) A shrinkage estimator prevents the encoder from overfitting to players with high participation, allowing our representation to generalize to the diversity and the sparsity of context-aware player distributions.

Following our representation framework, we design a Variational Recurrent Ladder Agent Encoder (VaRLAE) to learn player representations. VaRLAE utilizes a ladder structure [16] where, in each layer, latent variables condition on a context variable and the representations from upper layers, based on a causal dependence of latent variables that follows a Markov Game Model [17]. To incorporate play history into player representations, VaRLAE applies a recurrent network to sequential sports data. The ladder hierarchy of latent variables embeds both context and player information, which effectively improves the generative performance and prevents posterior-collapse [16, 18].

We demonstrate the generative performance of our VaRLAE on a massive National Hockey League (NHL) dataset containing over 4.5M events. VaRLAE achieves a leading identification accuracy (>12% for the players with sparse participation) over other deterministic encoders and policy representation models. To study how much the learned player representations improve downstream applications, we assess two major tasks in sports analytics: Predicting expected goals and final match score differences. Empirical results show the improvement in predictive accuracy after incorporating the embeddings generated by VaRLAE as input.

2 Related Work

We describe the previous works that are most related to our approach.

Agent Representation. Previous works [19, 20, 12] represented an agent by imitating its policy in the learning-from-demonstration [21] setting. Some recent works [11, 9, 22, 23] extended behavioral modelling to a multi-agent system. Multi-agent representations include modeling interactions between agents and their opponents’ behavior for reasoning about a specific goal, such as winning a poker or video game. These interactive policy representations scale poorly to a large number of agents [11, 24]. To identify the agent representations, [9] introduced a triplet loss that encourages an exponential distance between deterministic embeddings for different agents. Policy representations have been successful for the design of artificial agents in control problems, such as games or robot control. However, real human players in professional sports cannot be controlled like artificial agents. For sports analytics, the goal of representing players is predicting game-relevant events using player identity information [8, 7]. This goal often requires modeling as many aspects of a players’ behavior as possible, so we also represent not only the agents’ policies, but also the distribution of states and rewards associated with them. Some recent works [25, 26] have proposed modeling the trajectories of multiple agents to predict their future paths. They applied a Generative Adversarial Networks (GAN) to discriminate fake from real trajectories without explicitly learning agent representations, and thus their models are not directly applicable to embed player identities.

Variational Encoders. Variational encoders apply a set of latent variables \( z \) to embed the observations \( o \). The latent variables form a disentangled representation where a change in one dimension corresponds to a change in one factor of variation [27]. Such representations are often
3 Player Representation Framework

We introduce our framework of learning player representations through modeling a player generation process. This process becomes:

\[
q(z|s_t, a_t, r_t) \approx \int p(pl_t|z_t)q(z_t|s_t, a_t, r_t, pl_t)dz_t
\]

During training, our ELBo objective (Equation 9) induces a shrinkage effect through minimizing the Kullback–Leibler (KL) Divergence between the posterior representation for each individual player \(q(z_t|s_t, a_t, r_t, pl_t)\) and a context-specific prior \(p(z_t|s_t, a_t, r_t)\). A shrinkage estimator is ideal for

3.1 Contextual Variables for Ice Hockey Players

We model the ice-hockey games with a Markov Game Model (MGM) \(G = (S, A, T, R, \Omega)\). At each time step \(t\), an agent performs an action \(a_t \in A\) at a game state \(s_t \in S\) after receiving an observation \(o_t \in \Omega\). This process generates the reward \(r_t \sim R(s_t, a_t)\) and the next state \(s_{t+1} \sim T(s_t, a_t)\). For each game, we consider event data of the form \((o_0, p_{l_0}, a_0, r_0), (o_1, p_{l_1}, a_1, r_1), \ldots, (o_t, p_{l_t}, a_t, r_t), \ldots\): at time \(t\), after observing \(o_t\), player \(p_{l_t}\) takes a turn (possesses the puck) and chooses an action \(a_t\), which produces a reward \(r_t\) (goal score).

The play dynamics for the acting player \(p_{l_t}\) can be captured by the following contextual variables:

1. The game state \(s_t\) describes the game environment where the action is performed. To alleviate the partial observability of observations, a game state includes the game history: \(s_t \equiv (o_0, r_{t-1}, a_{t-1}, p_{l_{t-1}}, o_{t-2}, \ldots, o_0)\). We utilize the RNN hidden states \(h_{t-1}\) to capture the game history \(s_t\), so \(s_t \equiv (o_t, h_{t-1})\).

2. The action \(a_t\) records the action of the on-puck player. The reward \(r_t\) denotes whether a goal is scored after performing \(a_t\). As in general RL, the sequence state-action-reward can be interpreted causally in sports: the player makes observations summarized in a state signal \(s_t\), which influences his action \(a_t\); together with the environment, the state and action influence whether the player scores a goal. The corresponding causal graph is \(s_t \rightarrow a_t \rightarrow r_t\).

3.2 Player Representation via Player Generation

We introduce our framework of learning player representations through modeling a player generation distribution: \(p(pl_t|s_t, a_t, r_t)\). This distribution describes the where and what of a player’s characteristics: what game states they tend to act in, what their actions are, and what immediate outcomes they achieve. Inspired by previous work \([30, 31, 38]\), we utilize latent variables \(z_t\) as a representation of game context, which can be decoded to the distribution of current acting players:

\[
p(pl_t|s_t, a_t, r_t) = \int p(pl_t|z_t)p(z_t|s_t, a_t, r_t)dz_t
\]

where, before observing the acting player \(pl_t\), the context-aware prior \(p(z_t|s_t, a_t, r_t)\) models which players are likely to perform \(a_t\) under \(s_t\) and receive \(r_t\). The motivation for a contextualized representation is that the behavior of sophisticated agents, like professional players, is highly sensitive to context and it is difficult to learn a fixed representation that can adequately describe a player’s tendencies under every game context. After observing the target player \(pl_t\), we learn an approximate posterior \(q(z_t|s_t, a_t, r_t, pl_t)\) as a contextualized player representation, so the complete generative process becomes:

\[
p(pl_t|s_t, a_t, r_t) \approx \int p(pl_t|z_t)q(z_t|s_t, a_t, r_t, pl_t)dz_t
\]
learning player representation for team sports because 1) it has strong statistical properties that allow information to be transferred between the observations of different players, or of the same player in different game contexts. The shrinkage effect becomes stronger for players who share many statistical similarities under a game context, which draws their representations closer. This naturally formalizes our intuition that statistically similar players are assigned similar representations under similar game contexts. 2) The Shrinkage term also works as a regularizer that prevents the player representations from overfitting to some frequently-present players. Compare to a deterministic auto-encoder, the stochastic shrinkage estimator can generalize to more agents with sparse observations.

In contrast to a policy representation [9][38][10] that captures how a player acts in a given state, our player representation also models when and where they act, and how successful their actions tend to be. Since our context includes actions, states, and rewards, our learned embeddings reflect how players differ in all three of these dimensions.

4 A Variational Agent Encoder for Learning Agent Representations

Our VaRLAE architecture VaRLAE (Figure 1) for learning player representations follows our our representation framework (Section 3.2). We specify the generation and inference computations.

Generation. To incorporate match history into the game state $s_t$, the generation model combines a CVAE with recurrence following [36]. Previous works [18][39], however, showed that a high-capacity decoder often causes the problem of posterior collapse, where the generator computes the distribution of the currently acting player from context variables without relying on latent variables. To make latent variables embed meaningful player information, we introduce a separate latent variable $z_{c,t}$ for each component of our context, organized in a ladder VAE [16] (Figure 1). To utilize the context information (e.g. game state $s_t$ at the top layer), the generator is forced to utilize hierarchical latent variables. (Detailed analysis appears in the appendix ). The dependency of the ladder latent variables ($z_{a,t} \rightarrow z_{a,t} \rightarrow z_{r,t}$) follows the causal dependencies $s \rightarrow a \rightarrow r$ in a Markov game (Section 3.1).

Latent Priors. The priors are computed as a function of the game context, rather than a context-independent Gaussian distribution. At each time step, we compute the conditional priors (of player representations) following a Gaussian distribution at each layer:

$$p(z_{c,t}(c_t, z_{(+)t})) = \mathcal{N}(\mu_{c,t}(c_t, z_{(+)t}), \sigma_{c,t}(c_t, z_{(+)t}))$$

where $[\mu_{c,t}(c_t, z_{(+)t}), \sigma_{c,t}(c_t, z_{(+)t})] = \psi_{prior,c}([\psi_{c}(c_t, \psi_{(+)t}(z_{(+)t}))])$ (3)

The context variable $c \in \{s, a, r\}$, and $z_{(+)t}$ denotes latent variables from the upper layer. We omit $z_{(+)t}$ at the top layer. The neural functions $\psi_{c}(\cdot)$ are implemented by a Multi-Layer Perceptron (MLP) with batch normalization at each layer. We use linear and soft-plus activation function to compute the Gaussian parameters $\mu$ and $\sigma$ respectively.
Given the latent variables \( z_{r,t} \) sampled from the context-specific Gaussian prior, our model generates the label of the acting (or on-puck) player as follows:

\[
p(\pi_l|z_{r,t}) = \text{Categorical}(\theta_{1,t}, \ldots, \theta_{N,t}), \quad \text{where} \quad \theta_t = \phi\{\psi^{dec}(\psi^{enc}(z_{r,t}))\}
\]

The neural function \( \psi^{enc} \) extracts features from latent variable \( z_{r,t} \). These features are then sent to another decoder function \( \psi^{dec} \). Given the decoder outputs, we apply a softmax function \( \phi \) to generate categorical parameters \( \Theta_t \). The output \( \theta_{1,t} \) represents the probability of player \( i \) acting at time \( t \).

**Inference.** We apply variational inference to derive an objective function for estimating the parameters of our hierarchical model. Our VaRLAE defines a top-down dependency structure by utilizing the hierarchical priors and approximate posteriors on latent variables to derive an approximate log-likelihood function for the observed data [16, 32].

After observing player \( pl_t \), we first implement a deterministic upward pass to recursively compute the approximate posterior. Conditioning on reward variable \( r_t \), the bottom layer computes:

\[
[\hat{\mu}^q_{r,t}, \hat{\sigma}^q_{r,t}] = \psi^{enc}(d_{r,t}) \quad \text{where} \quad d_{r,t} = [pl_t, \psi^{dec}(r_t)]
\]

The higher layers take information from lower layers and compute:

\[
[\hat{\mu}^q_{c,t}, \hat{\sigma}^q_{c,t}] = \psi^{enc}(d_{c,t}) \quad \text{where} \quad d_{c,t} = [d_{(-)t}, \psi^{dec}(ct)]
\]

Here we have \( c \in \{a, s\} \), and \( d_{(-)t} \) denotes deterministic outputs from a lower layer. Similar to the generator, neural functions \( \psi^{enc} \) are implemented by MLP with batch normalization.

We then implement a stochastic downward pass to recursively compute the approximate posterior. At the top layers, the Gaussian posterior applies the estimated parameters from a deterministic function:

\[
q(z_{s,t} | s_t, pl_t) = \mathcal{N}(\hat{\mu}^q_{s,t}, \hat{\sigma}^q_{s,t}) \quad \text{where} \quad [\hat{\mu}^q_{s,t}, \hat{\sigma}^q_{s,t}] = [\hat{\mu}^q_{p,t}, \hat{\sigma}^q_{p,t}]
\]

At the lower layers, the inference model applies a precision-weighted combination of \( (\hat{\mu}^q_{c,t}, \hat{\sigma}^q_{c,t}) \) carrying bottom-up information and \( (\mu^p_{c,t}, \sigma^p_{c,t}) \) from the generative distribution carrying top-down prior information. The approximate posteriors are computed by:

\[
q(z_{c,t} | ct, z_{(+),t}, pl_t) = \mathcal{N}(\hat{\mu}^q_{c,t}, \hat{\sigma}^q_{c,t}) \quad \text{where}
\]

\[
\hat{\mu}^q_{c,t} = \frac{\hat{\mu}^q_{c,t} (\hat{\sigma}^q_{c,t})^{-2} + \mu^p_{c,t} (\sigma^p_{c,t})^{-2}}{(\hat{\sigma}^q_{c,t})^{-2} + (\sigma^p_{c,t})^{-2}}
\]

\[
\hat{\sigma}^q_{c,t} = \frac{1}{(\hat{\sigma}^q_{c,t})^{-2} + (\sigma^p_{c,t})^{-2}}
\]

Here we have \( c \in \{a, r\} \), and \( z_{(+),t} \) denotes latent variables from the upper layer. This parameterization has a probabilistic motivation by viewing \( \hat{\mu}^q_{c,t} \) and \( \hat{\sigma}^q_{c,t} \) as the approximate Gaussian likelihood that is combined with a Gaussian prior \( \mu^p_{c,t} \) and \( \sigma^p_{c,t} \) from the generative distribution. Together these form the approximate posterior distribution \( q(z|z_{(+)}, c) \) using the same top-down dependency structure for both inference and generation.

Based on [36], the timestep-wise variational lower bound for our model is:

\[
\sum_{t=1}^T \{ \sum_{c \in \{a, s\}} -\beta D_{KL}[q(z_{c,t} | ct, z_{(+),t}, pl_t) || p(z_{c,t} | ct, z_{(+),t})] + \mathbb{E}_{z_{r,t} \sim q(z_{r,t})} \left[ \log p(\pi_l | z_{r,t}) - \lambda \mathcal{L}^c(z_{r,t}, z_{s,t}, a_t, r_t) \right] \} = \mathcal{L}^c
\]

where \( \beta \) controls the scale of KLD regularization. To mitigate local optima caused by posterior collapse (\( D_{KL}() \) drops to 0) at the initial stage of training [18], we apply a warm-up from deterministic to variational encoder by scaling \( \beta \) from 0 to 1 [16]. The bottom layer latent variables \( z_{r,t} \) absorb context and player information from upper layer and form a contextualized player representation:

\[
q(z_t | s_t, a_t, r_t, pl_t) = q(z_{r,t} | \cdot)
\]

This real-valued vector can replace the one-hot player representation and facilitates downstream applications such as predicting expected goals or score differences (see Section 5.3). We also add an application loss \( \mathcal{L}^c \) with a parameter \( \lambda \) to control its scale. This loss combines the gradient of the application models with the embedding inference. Co-training the embedding model and the application model significantly accelerates training and dynamically incorporates player information into different downstream applications.
5 Empirical Evaluation

We evaluate the generative performance of the embedding models for player identification and study the usefulness of embeddings for application tasks [40, 41]. Our application tasks include 1) estimating expected goals and 2) predicting the final score differences, which are among the most challenging tasks in sports analytics [42, 8].

5.1 Experiment Settings

We introduce our ice-hockey dataset and comparison methods following an ablation design. The Appendix gives further details about experimental settings and implementations.

Dataset: We utilize a dataset constructed by Sportlogiq. The data provides information about game events and player actions for the entire 2018-2019 National Hockey League (NHL) season, which contains 4,534,017 events, covering 31 teams, 1,196 games and 1,003 players. The dataset consists of events around the puck. Each event includes the identity and action of the player possessing the puck, with time stamps and features of the game context. (We provide a complete list of game features in Appendix.) The dataset records which unique player possesses the puck. In this paper, we refer to the acting player as the on-puck player. We randomly divide the dataset containing 1,196 games into a training set (80%), a validation set (10%), and a testing set (10%) and implement 5 independent runs. The resulting means and variances are reported.

Comparison models: We employ an ablation design that removes different components from our full VaRLAE system. We first remove the hierarchical dependency of latent variables and train a Conditional Variational Recurrent Neural Network (CVRNN) [35]. CVRNN concatenates the context variables ($s_t, a_t$ and $r_t$) and applies a single layer of latent variables to embed players with variational inference. We then replace the variational encoder with a Conditional Auto-Encoder at each RNN cell (CAERN) that learns a deterministic player representation. The third model is a Conditional Variational Auto-Encoder (CVAE) [33] that discards the play history and conditions only on the current observations with no recurrence. Replacing the variational model in CVAE with a Deterministic Encoder yields (DE) player embedding [8]. DE is a regressor that directly maps the current observations to the acting player. We also compare our player representation framework to traditional policy embedding. The implementation follows a state-of-the-art Multi-Agent Behavior Encoder (MA-BE) [9]. We present a more detailed summary of the comparison models in the Appendix. In application tasks (Section 5.3), we also compare the options of 1) applying one-hot player identities (Pids) directly to 2) adding no player information (N/A) to application models.

5.2 Generative Performance of Embedding Models: On-Puck Player Identification

This experiment studies the generative performance of embedding models: predict which player is acting given the current match context. We compare our VaRLAE model to 6 baselines: 1) identifying player with embedding models: DE, CVAE, MA-BE, CAERN and CVRNN 2) applying a RNN to model the game history and predict the acting player without a player embedding. The large player space (over 1k players) undermines the performance of encoders that do not utilize the recent play history. To make a fair comparison, our constrained setting limits the predictions to recently acting players: the current on-puck player (the correct answer) and the players that have possessed the puck in the previous 10 steps during testing. (10 is the trace length of our RNNs.) To study the identification performance for the players with sparse participation, we select the players (a total of 51 players) with fewer than 100 observations (out of a total of 4M events) and report the results.

Table 1: Results for acting players identification. We report both Accuracy and Log-Likelihood (LL).

<table>
<thead>
<tr>
<th>Prediction Method</th>
<th>Standard</th>
<th>Constraining</th>
<th>Sparse participation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>LL</td>
<td>Accuracy</td>
</tr>
<tr>
<td>VaRLAE</td>
<td>50.01% ± 2.56E-6</td>
<td>-1.76 ± 1.29E-3</td>
<td>78.54% ± 3.62E-6</td>
</tr>
<tr>
<td>CAERNN</td>
<td>43.64% ± 1.27E-5</td>
<td>-2.11 ± 1.55E-3</td>
<td>67.43% ± 5.21E-6</td>
</tr>
<tr>
<td>RNN</td>
<td>36.49% ± 2.21E-6</td>
<td>-3.10 ± 2.80E-4</td>
<td>54.21% ± 2.80E-6</td>
</tr>
<tr>
<td>MA-BE</td>
<td>19.74% ± 2.47E-4</td>
<td>-3.08 ± 1.75E-3</td>
<td>51.75% ± 7.58E-5</td>
</tr>
<tr>
<td>CVAE</td>
<td>11.94% ± 2.80E-5</td>
<td>-4.90 ± 2.84E-5</td>
<td>28.33% ± 2.96E-4</td>
</tr>
<tr>
<td>DE</td>
<td>9.40% ± 3.06E-5</td>
<td>-17.42 ± 2.23E-1</td>
<td>26.14% ± 6.43E-5</td>
</tr>
<tr>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 1 shows the results for acting player identification. VaRLAE achieves leading performance on both the prediction accuracy and the log-likelihood. The improvement is most apparent (> 10%) for the players with sparse participation, which demonstrates VaRLAE is more robust to unbalanced player participation. For the variational encoders, they perform better than the deterministic encoders. This is because the shrinkage regularizer prevents overfitting to the distribution of popular players in the training dataset. Game history is another crucial aspect for player identification, allowing the RNN models to outperform the memoryless models with observations only from the current time step. Constraining the candidate players to the group of recent on-puck players significantly improves the identification performance. The difference is most apparent for MA-BE (improves > 30%), which indicates that policy embeddings do not distinguish individual players sufficiently.

**Embedding Visualization and Case Study:** We visualize the generated player embeddings as follows. 1) Randomly select 5 games from the testing set. 2) Compute the contextualized embeddings for the acting player at each event. 3) Visualize the high-dimensional embedding with the unsupervised T-distributed Stochastic Neighbor Embedding (T-SNE) [43]. Figure 2 illustrates the scatter plots. The embeddings computed by our VaRLAE (top plots) show a shrinkage effect.

**Player Positions.** VaRLAE embeddings are similar for players in the same position (left column), as they are more likely to perform similarly. Although the position information is masked during training, our model manages to infer it from players’ behavior and assigns closer embeddings to players in the same position. For a case study on defense men (middle column), we select 5 players who are well-known and perform a similar number of actions in our 5 selected games. The plot shows that while defence men tend to be mapped to similar embeddings, our encoder also learns which contexts distinguish them from each other.

**Action Locations.** Action locations are part of the state variable $s_t$ conditioning player embeddings (right column). VaRLAE embeddings are similar for players when they act in the same zone. The similarity is weaker for the CAERNN embeddings (bottom plots). Without shrinkage, CAERNN overfits; it over-emphasizes outliers in the training data, which prevents generalizing to player behavior in the test data. The Appendix illustrates the influence of action types on embeddings.

![Figure 2: Embedding visualization. Each data point corresponds to a player embedding conditioning on the game context at the current time step $t$. The player embeddings are labelled 1) player positions (on the left column, including Center (C), Defense (D), Left-Wing (LW) and Right-Wing (RW) and Goalie (G)) 2) 5 selected defence men (on the middle column) and 3) player locations (on the right column, including Defence Zone (DZ), Neutral Zone (NZ) and Offensive Zone (OZ)). The embeddings are computed by VaRLAE (top plots) and CAERNN (bottom plots) respectively.](image)

### 5.3 Predictive Performance On Application Tasks

We validate our VaRLAE model by feeding the generated embeddings to other application models and studying their influence on the model performance in practical application tasks.
we apply reinforcement learning; specifically the temporal difference method Sarsa prediction [45]. At time \( t \), we input \( s_t, \text{shot}_t, z_t \) to the application model \( \zeta \) (implemented as a RNN), which is trained to output \( p(\text{goal}_{t+1}|s_t, \text{shot}_t, z_t) \): the probability that a goal is scored after the player \( p_l \) makes the shot. Our dataset provides ground-truth labels for whether a given shot led to a goal. Since only a few shot attempts lead to a goal (<3.9%), the training data is highly imbalanced. We employ a resampling method [44] so that successful and failed shots occur equally often during training.

### Expected Goals Estimation

We validate the player embeddings for the practical task of estimating the Expected Goal (XG). An XG metric is a shot quality model that weights each shot by its chance of leading to a goal [42]. We incorporate \( z_t \) from the learned player representations into XG prediction. For each testing game \( m \), we input \( s_t, \text{shot}_t, z_t \) to the application model \( \zeta \) (implemented as a RNN), which is trained to output \( p(\text{goal}_{t+1}|s_t, \text{shot}_t, z_t) \): the probability that a goal is scored after the player \( p_l \) makes the shot. Our dataset provides ground-truth labels for whether a given shot led to a goal. Since only a few shot attempts lead to a goal (<3.9%), the training data is highly imbalanced. We employ a resampling method [44] so that successful and failed shots occur equally often during training.

### Score Difference Prediction

Dynamic Score Difference Prediction (DSDP) is a recently introduced task [8]: predict the final score difference \( SD(T) \) under a game context \((s_t, a_t)\) where \( T \) runs from 0 to \( T \) (game ends). In preliminary experiments, we observed that traditional supervised learning methods suffer a large training variance (especially early in the game when many outcomes are equally likely). To exploit the temporal dependencies between score differences at successive times, we apply reinforcement learning: specifically the temporal difference method Sarsa prediction [45]. Sarsa learns a Q-function for a generic home/away team to estimate the expected cumulative goal scoring: 
\[
Q_{\text{team}}(s_t, a_t) = E\left[\sum_{\tau=t}^{T} g_{\text{team},\tau}\right] \text{ where } g_{\text{team}} = \text{Home} / \text{Away} \text{ and } \text{goal}_{\text{team}} = 1 \text{ if the team scores at } t \text{ and } 0 \text{ otherwise. From Q-functions, the Predicted Score Difference (PSD) at } t \text{ is given by } PSD(t) = Q_{\text{Home}}(\cdot) - Q_{\text{Away}}(\cdot) + SD(t). \text{ Our application model } \zeta \text{ is a DRQNN [57] that computes the Q-functions. The inputs are state } s_t, \text{ action } a_t, \text{ and the embedding } z_t \text{ for the acting player } p_l. \text{ For each testing game } m \text{ and time } t, \text{ the absolute error is given by } |PSD(t_m) - SD(T_m)|. \text{ For each game time } t, \text{ Figure 3 plots the mean and the variance of the absolute error over all testing games } m = 1, \ldots, M. \text{ The plot shows a larger difference between real and predicted SDs at the beginning of a game, but the mean and variance of the difference become smaller towards the game end. Among the evaluated embedding methods, our VaRLAE (the black line) manages to generate the player representations that lead to the most accurate predictions. We also find that the accuracy advantage is strongest towards the early game, especially compared to the N/A and pids. This indicates that an informative player representation significantly alleviates the difficulty of predicting multiple outcomes early in the game. Averaging over game times } t \text{ defines the game prediction error for each tested game } m. \text{ Table 5 reports the mean and standard deviation for the game prediction error. The VaRLAE embeddings yield the lowest Mean Absolute Error (MAE).}

\section{Conclusion}

Capturing what team sports players have in common and how they differ is one of the main concerns of sports analytics. This work introduces a \textit{player representation via player generation} framework that
learns deep contextualized representations for ice hockey players. We described a VaRLLAE model for sports data, based on a Markov Game model representation. The ELBo loss (Equation (9)) induces a shrinkage effect such that similar players are mapped to similar representations in similar match contexts. We validate the player representation on two downstream applications that are important in sports analytics: predicting expected goals and final match score differences. While our evaluation focuses on ice hockey, our approach is general and can be applied to other team sports, or indeed any strategic setting that can be modelled as a Markov game with many participants. A direction of future work is to combine player representations with modelling player interactions. Another is learning representations for different lineups, given a dataset with full observations of on-court players.

7 Broader Impact

We expect the main impact outside of the machine learning community to be in professional sports. As an entertainment industry, professional sports increases the quality of life for many people. With respect to the broad challenges of our society (polarization, inequality, bias), entertainment is in our view neutral. The main stakeholders impacted will be sports teams, managers, and athletes. For sports stakeholders, we expect mainly positive and some negative outcomes.

Positive Outcomes. Our work significantly enhances the reliability of the machine learning algorithm on modeling complex sport dynamics. This allows teams, coaches, and fans to better understand a player’s style, influence, and contributions. The result will be better and even more enjoyable sports. For sports stakeholders, the main positive outcome will be that coaches can deploy players more effectively, which will help them display and improve their considerable skills. This will create winners and losers among the players, but on the whole, a data-driven approach will increase fairness and objectivity in player performance ranking, while decreasing bias. Bias towards underrepresented groups is known to hurt the perception and career chances of underrepresented groups, such as indigenous ice hockey players; Fred Sasakamoose is a prominent example.

Negative Outcomes. Putting players’ performance and contribution under intense study may lead to more pressure on the professional players. We believe that for most players this Taylorist pressure [46] is outweighed by the guidance for how to improve both their play and their market value. To apply our model, sports teams should invest in technical analytic resources, which might not be affordable for small clubs. It potentially increases the inequality between top-ranked teams and lower-ranked teams. To help level the analytics playing field, we have placed our code in the public domain at github.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>1.55 ± 0.35</td>
</tr>
<tr>
<td>Pid</td>
<td>1.56 ± 0.32</td>
</tr>
<tr>
<td>DE</td>
<td>1.48 ± 0.24</td>
</tr>
<tr>
<td>CVAE</td>
<td>1.49 ± 0.29</td>
</tr>
<tr>
<td>MA-BE</td>
<td>1.45 ± 0.25</td>
</tr>
<tr>
<td>CAERNN</td>
<td>1.31 ± 0.23</td>
</tr>
<tr>
<td>CVRNN</td>
<td>1.32 ± 0.27</td>
</tr>
<tr>
<td>VaRLLAE</td>
<td><strong>1.28 ± 0.29</strong></td>
</tr>
</tbody>
</table>

Table 3: The test set game prediction error between predicted and final score differences for the entire game.
Acknowledgments and Disclosure of Funding

This project was supported by a Strategic Project Grant from the Natural Sciences and Engineering Research Council of Canada. We are grateful for helpful comments from Jesse Davis, Tim Swartz, and the reviewers and participants in the 2019 AAAI Team Sports workshop. Our computations were facilitated by a GPU donation from NVIDIA.

References


