



(19) **United States**

(12) **Patent Application Publication**

Ruiz et al.

(10) **Pub. No.: US 2019/0228290 A1**

(43) **Pub. Date: Jul. 25, 2019**

(54) **METHOD AND SYSTEM FOR INTERACTIVE, INTERPRETABLE, AND IMPROVED MATCH AND PLAYER PERFORMANCE PREDICTIONS IN TEAM SPORTS**

(52) **U.S. Cl.**
CPC *G06N 3/0427* (2013.01); *A63B 71/0616* (2013.01); *G06N 20/20* (2019.01); *G06N 3/08* (2013.01)

(71) Applicant: **STATS LLC**, Chicago, IL (US)

(57) **ABSTRACT**

(72) Inventors: **Hector Ruiz**, Barcelona (ES); **Sujoy Ganguly**, Chicago, IL (US); **Nathan Frank**, Chicago, IL (US); **Patrick Lucey**, Chicago, IL (US)

(73) Assignee: **STATS LLC**, Chicago, IL (US)

(21) Appl. No.: **16/254,108**

(22) Filed: **Jan. 22, 2019**

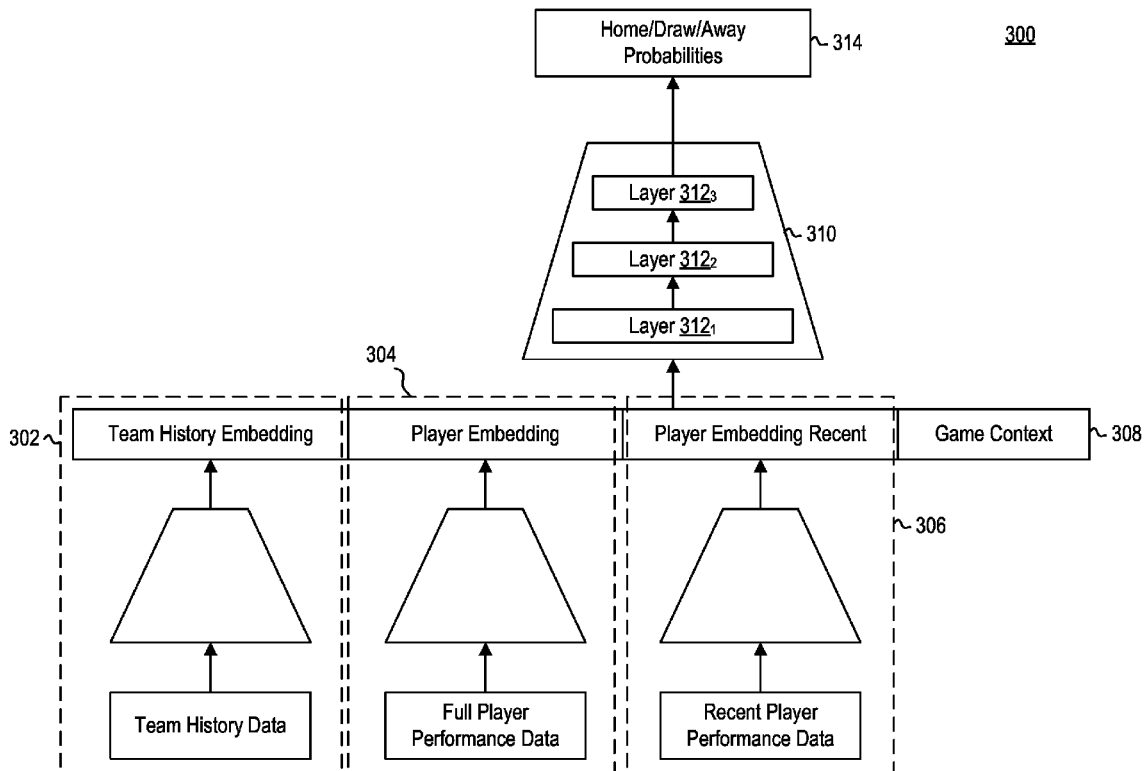
Related U.S. Application Data

(60) Provisional application No. 62/619,894, filed on Jan. 21, 2018.

Publication Classification

(51) **Int. Cl.**
G06N 3/04 (2006.01)
G06N 3/08 (2006.01)
G06N 20/20 (2006.01)

A method of generating an outcome for a sporting event is disclosed herein. A computing system retrieves tracking data from a data store. The computing system generates a predictive model using a deep neural network. The one or more neural networks of the deep neural network generates one or more embeddings comprising team-specific information and agent-specific information based on the tracking data. The computing system selects, from the tracking data, one or more features related to a current context of the sporting event. The computing system learns, by the deep neural network, one or more likely outcomes of one or more sporting events. The computing system receives a pre-match lineup for the sporting event. The computing system generates, via the predictive model, a likely outcome of the sporting event based on historical information of each agent for the home team, each agent for the away team, and team-specific features.



100

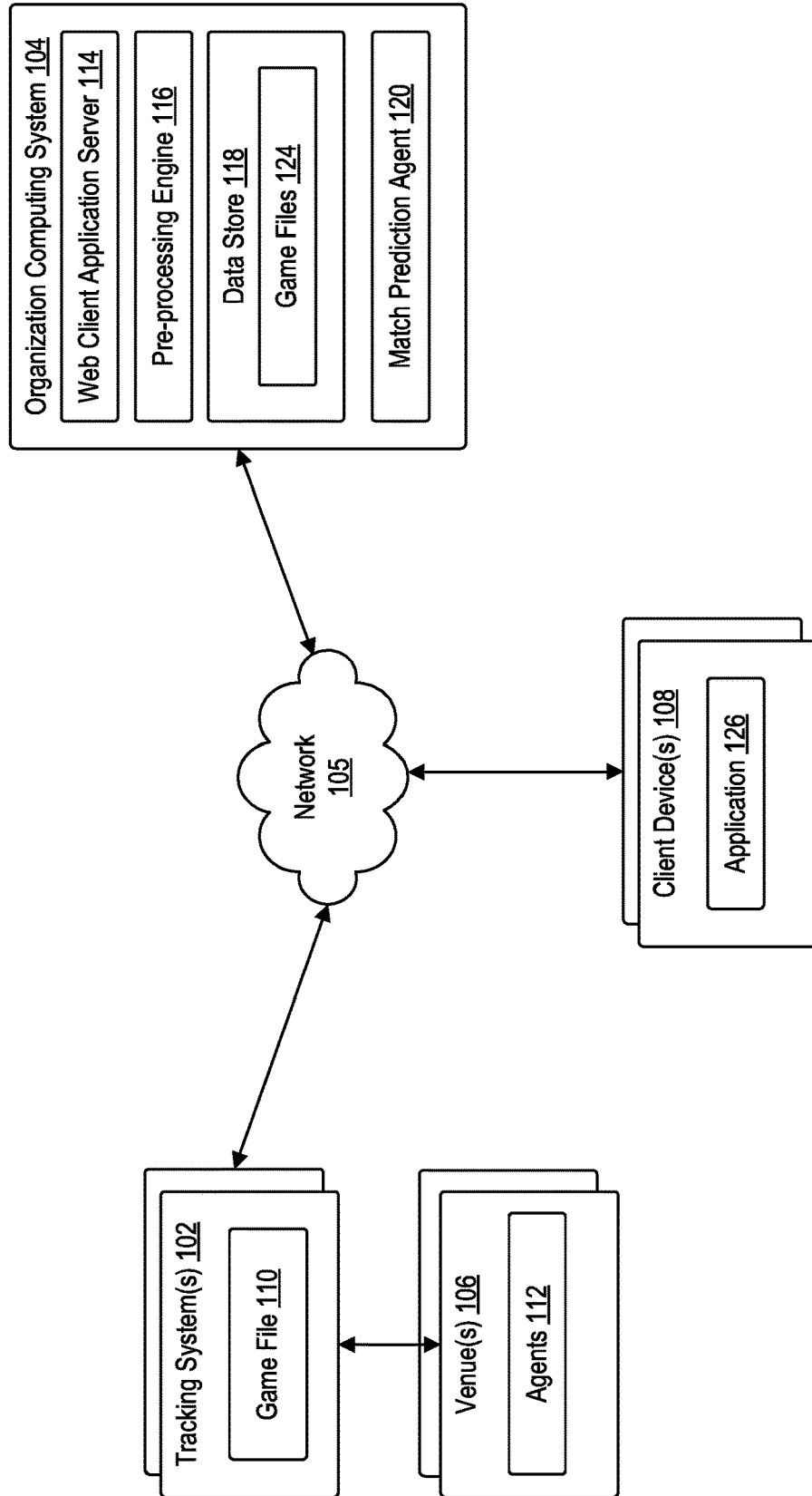


FIG. 1

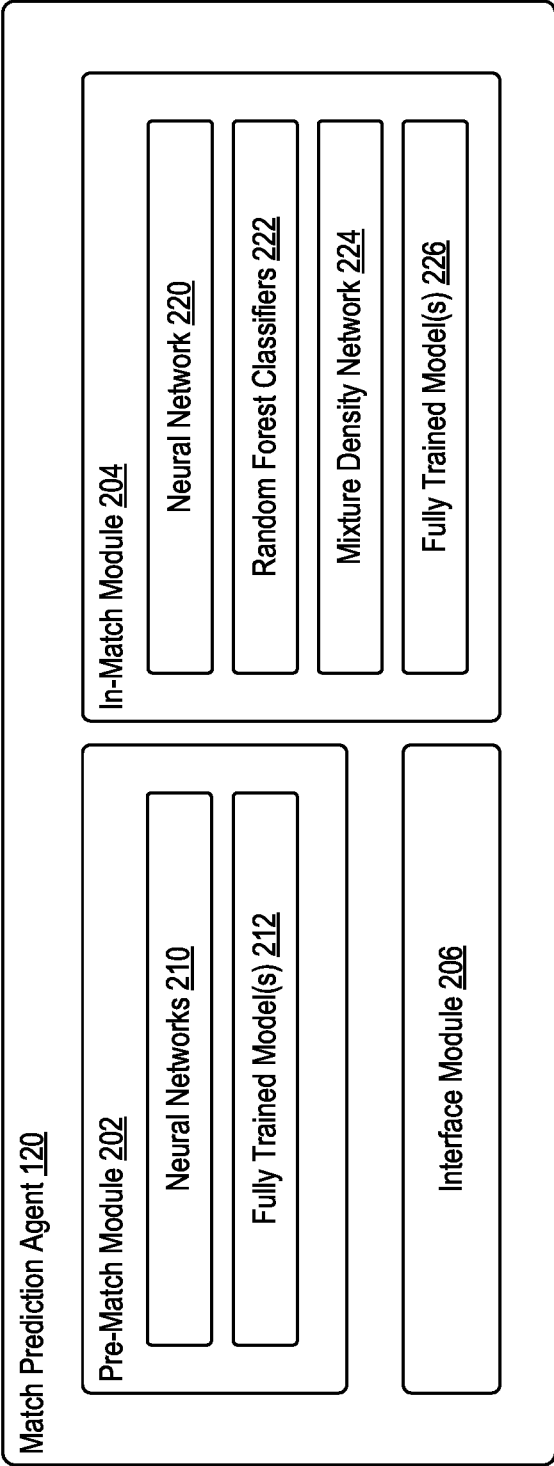


FIG. 2

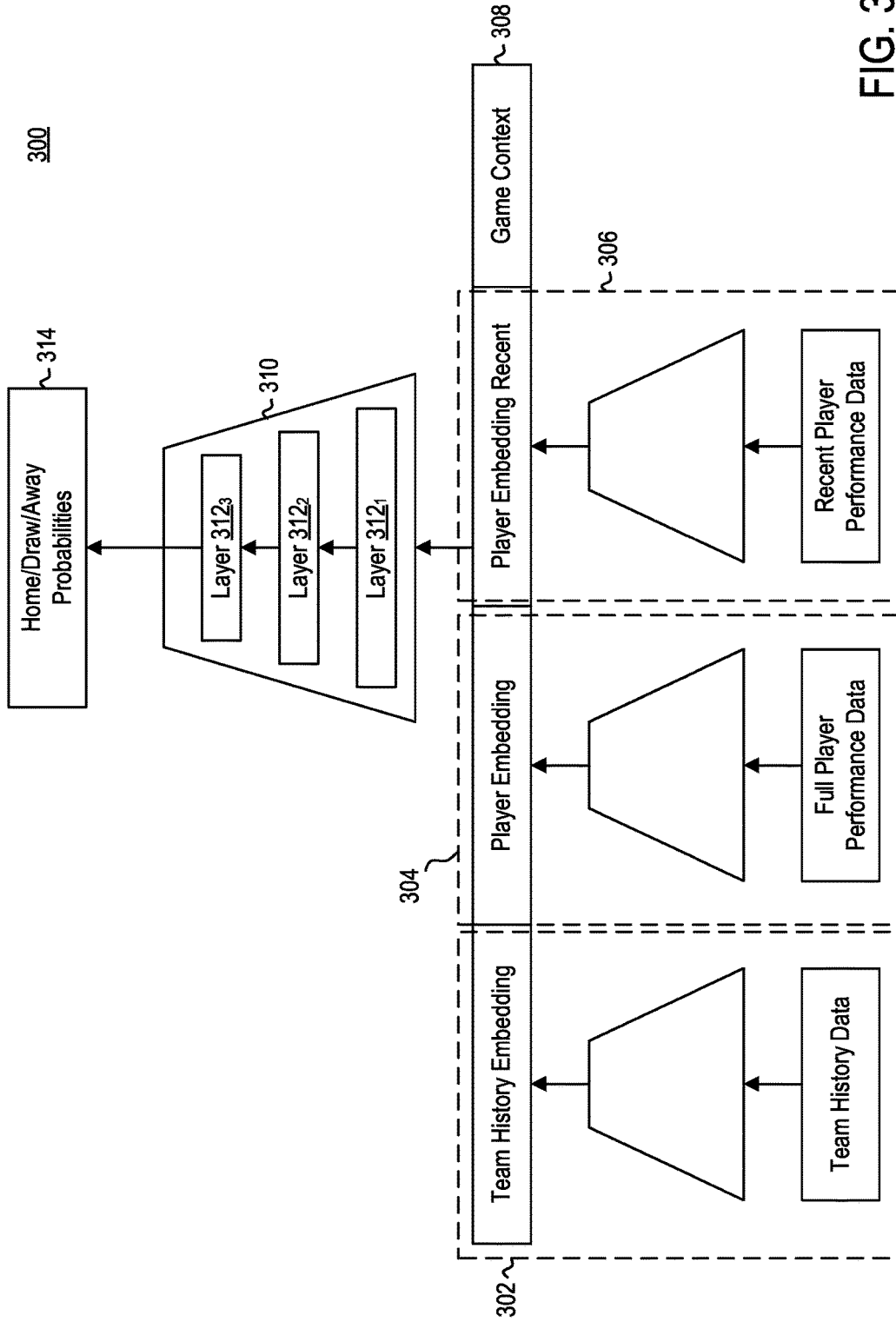


FIG. 3A

330

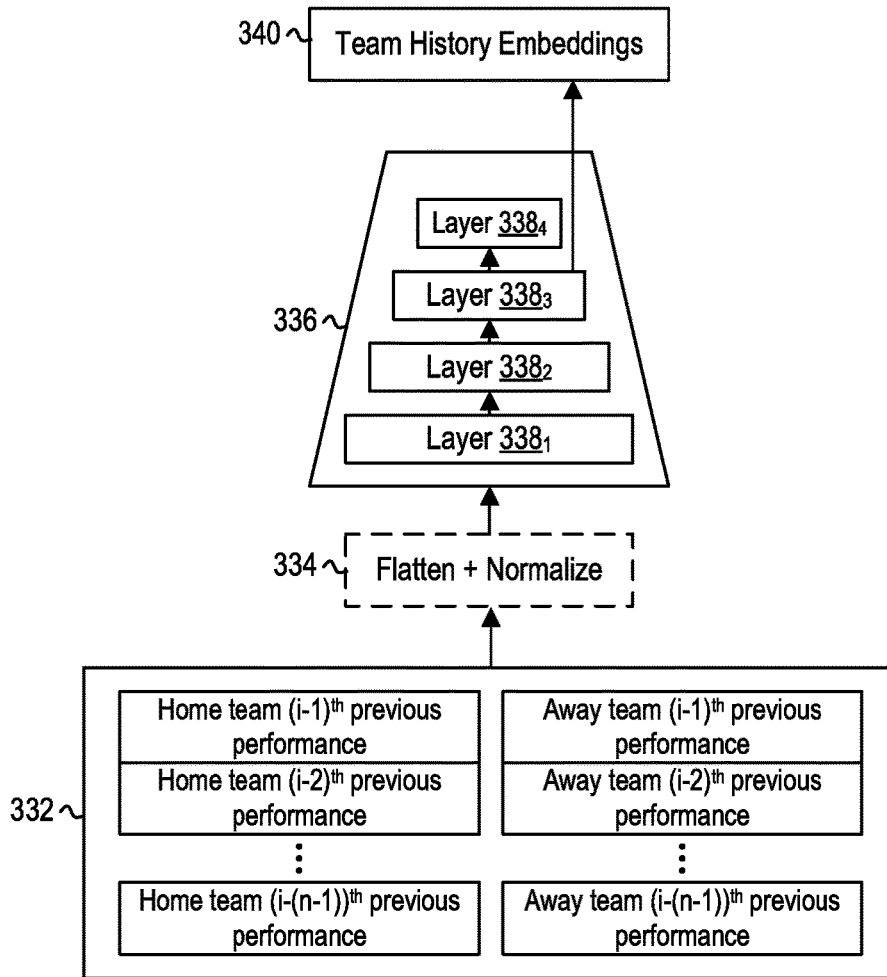


FIG. 3B

350

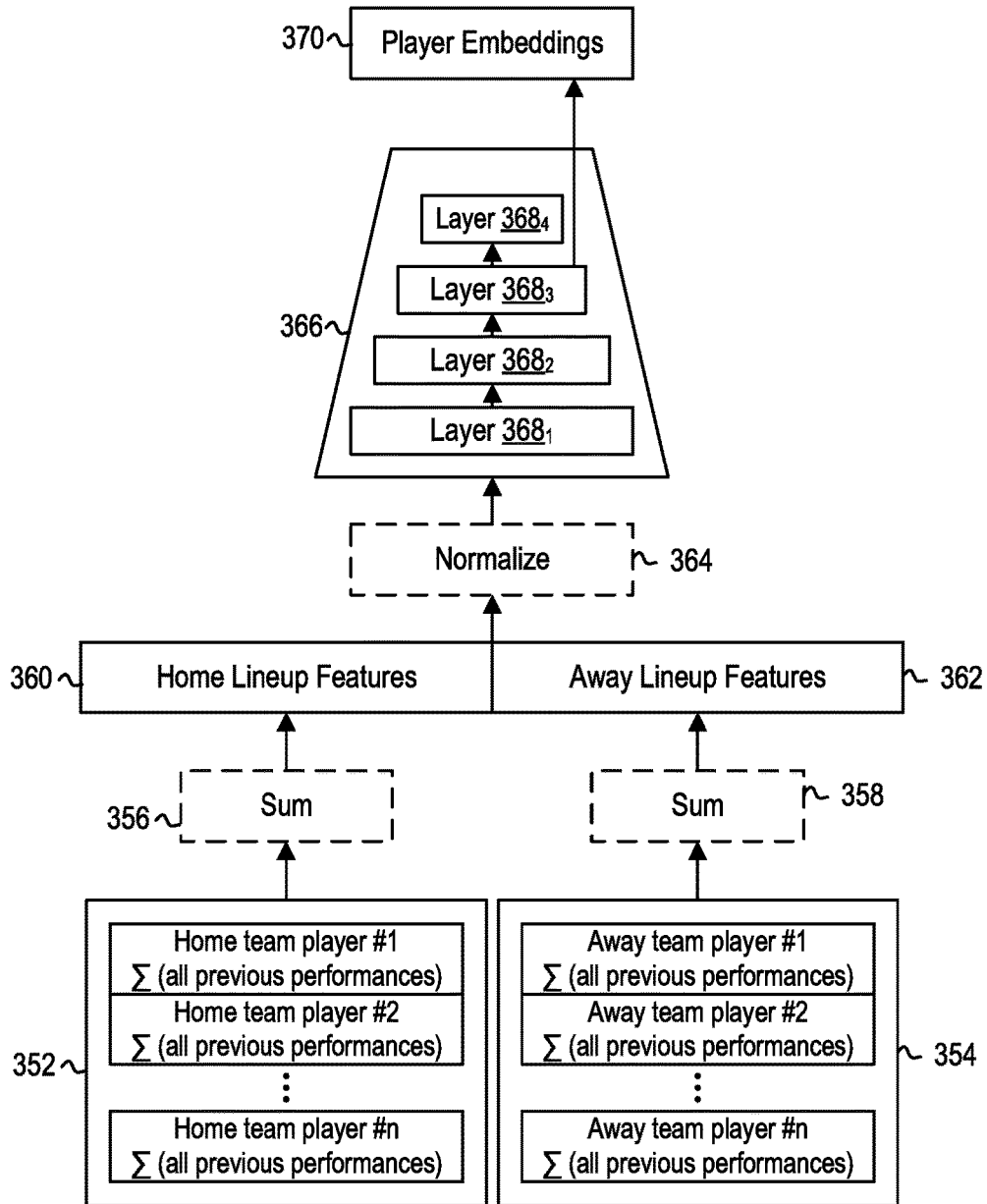


FIG. 3C

380

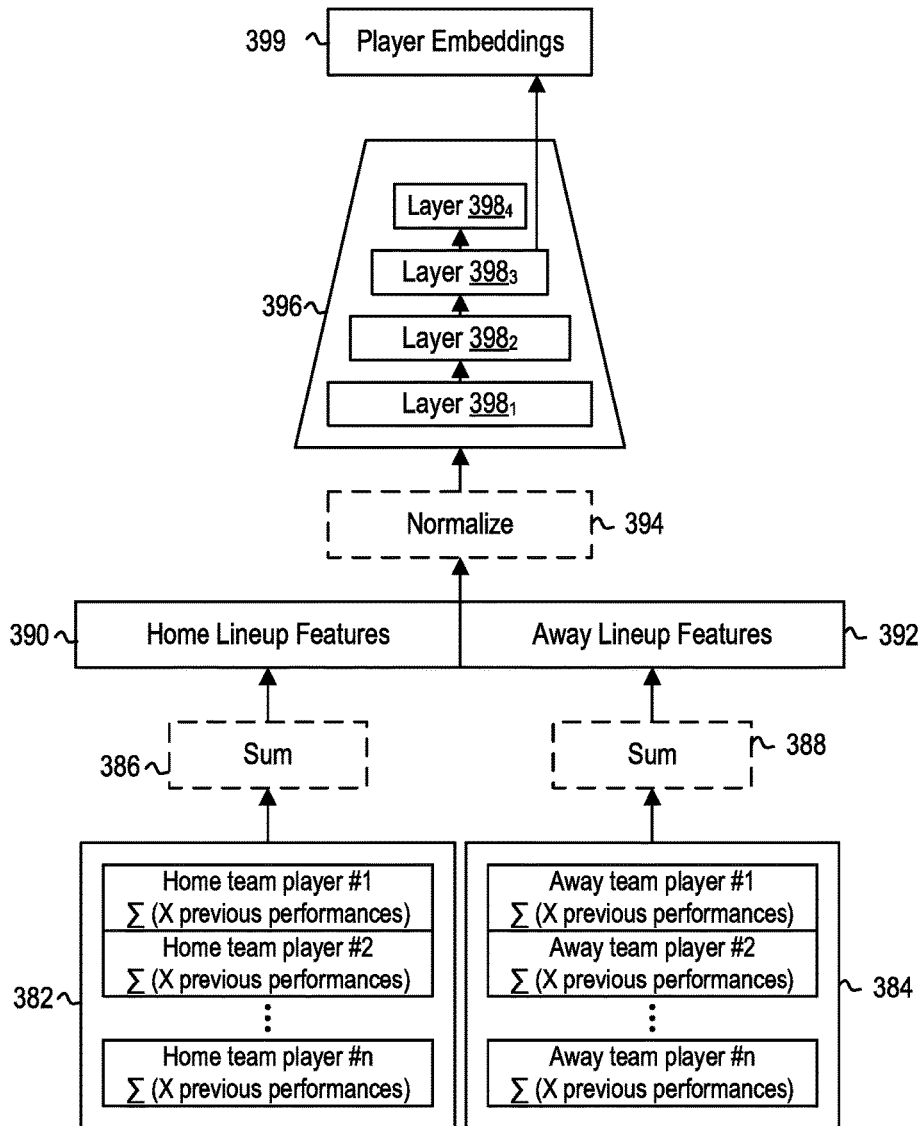


FIG. 3D

400

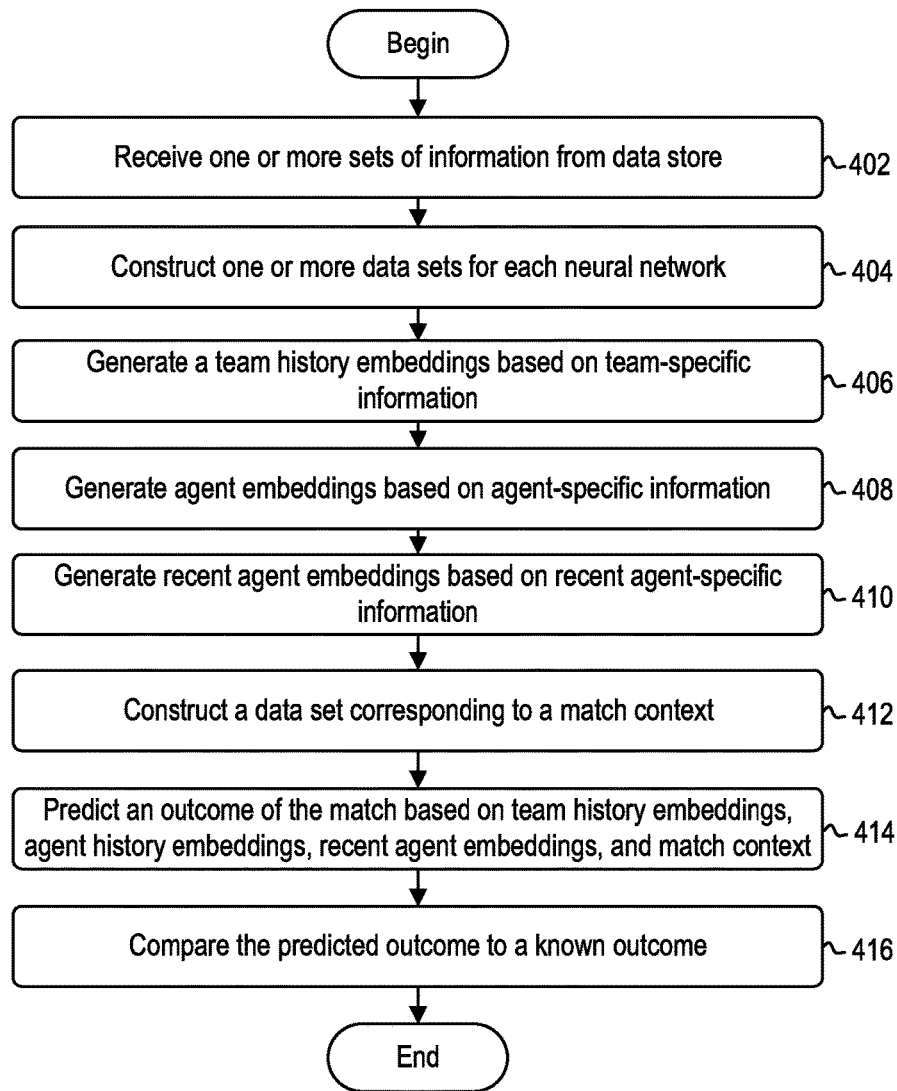


FIG. 4

600

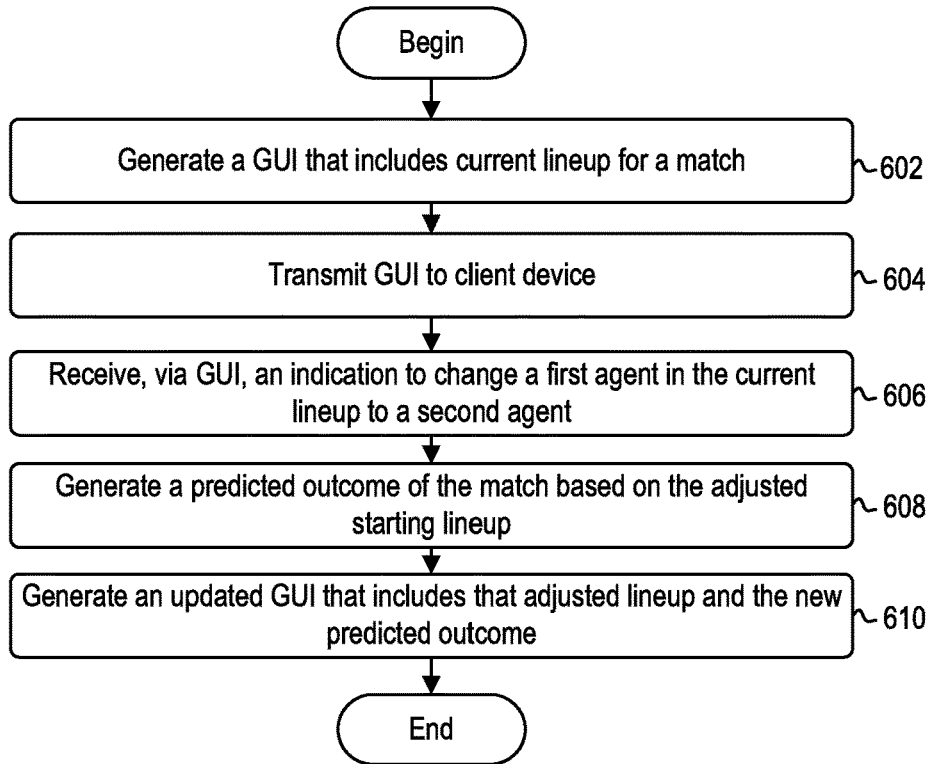


FIG. 6

700

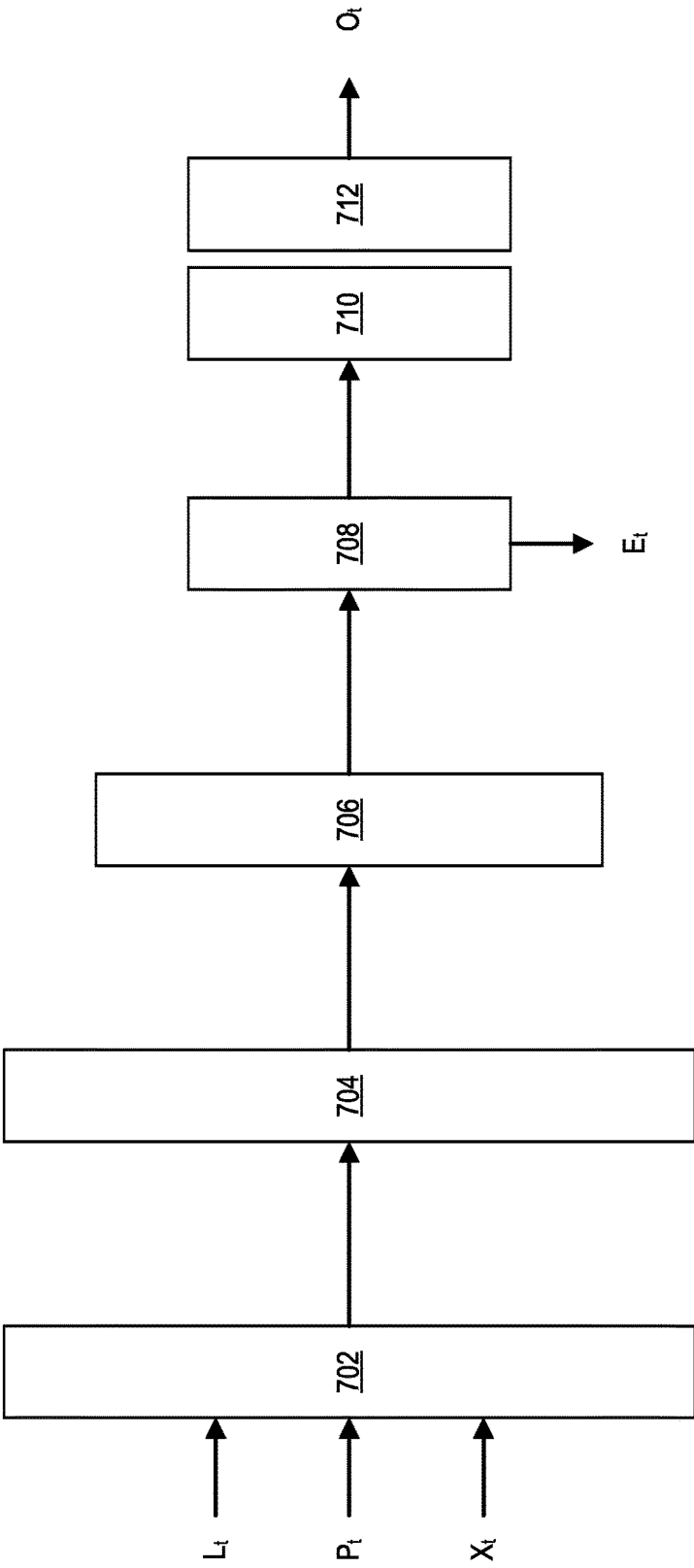


FIG. 7A

750

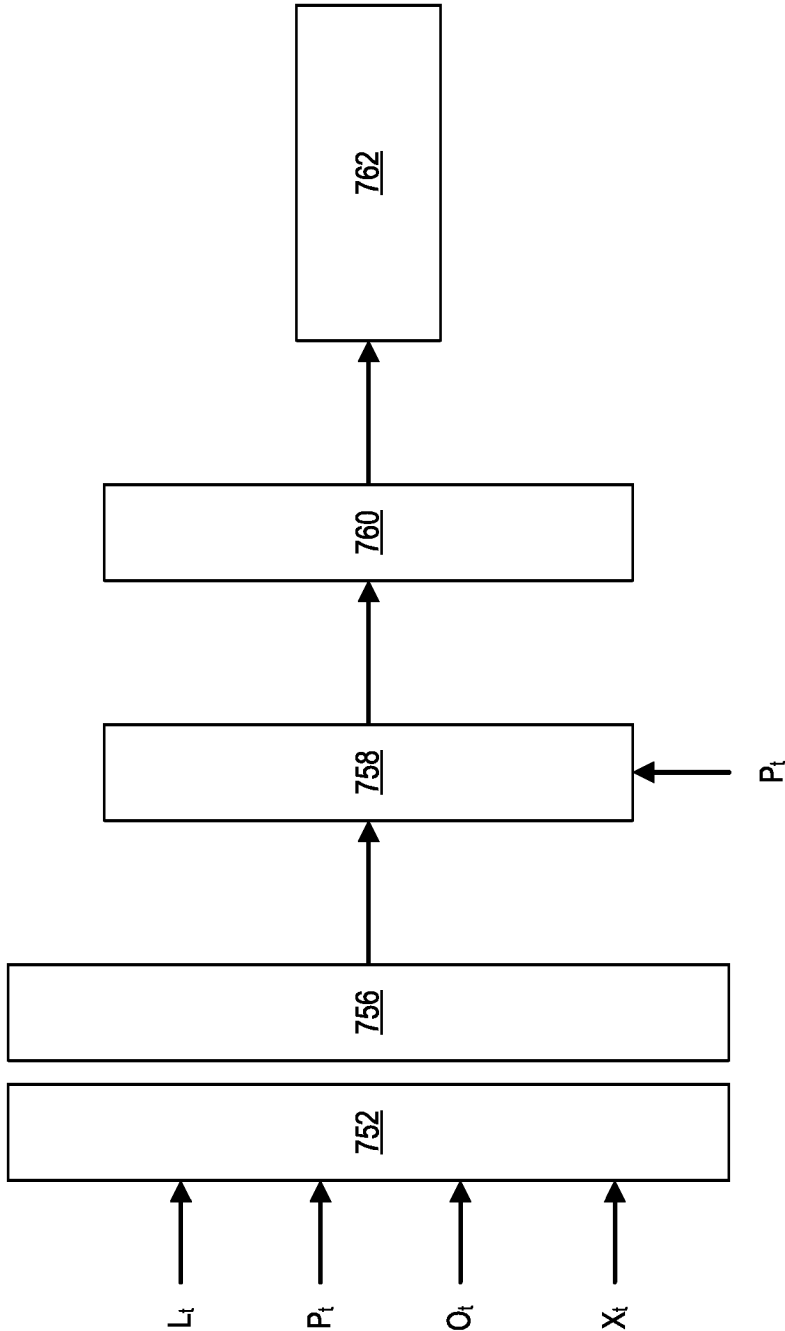


FIG. 7B

800

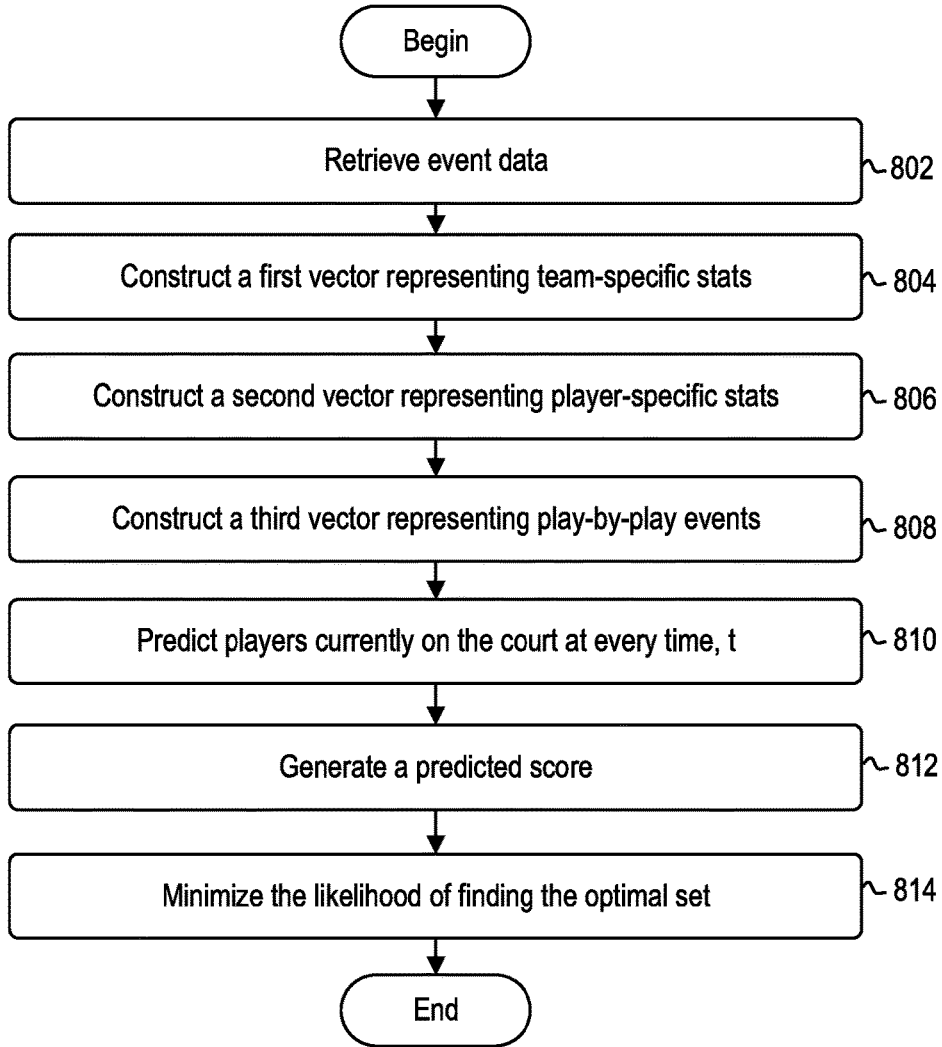


FIG. 8

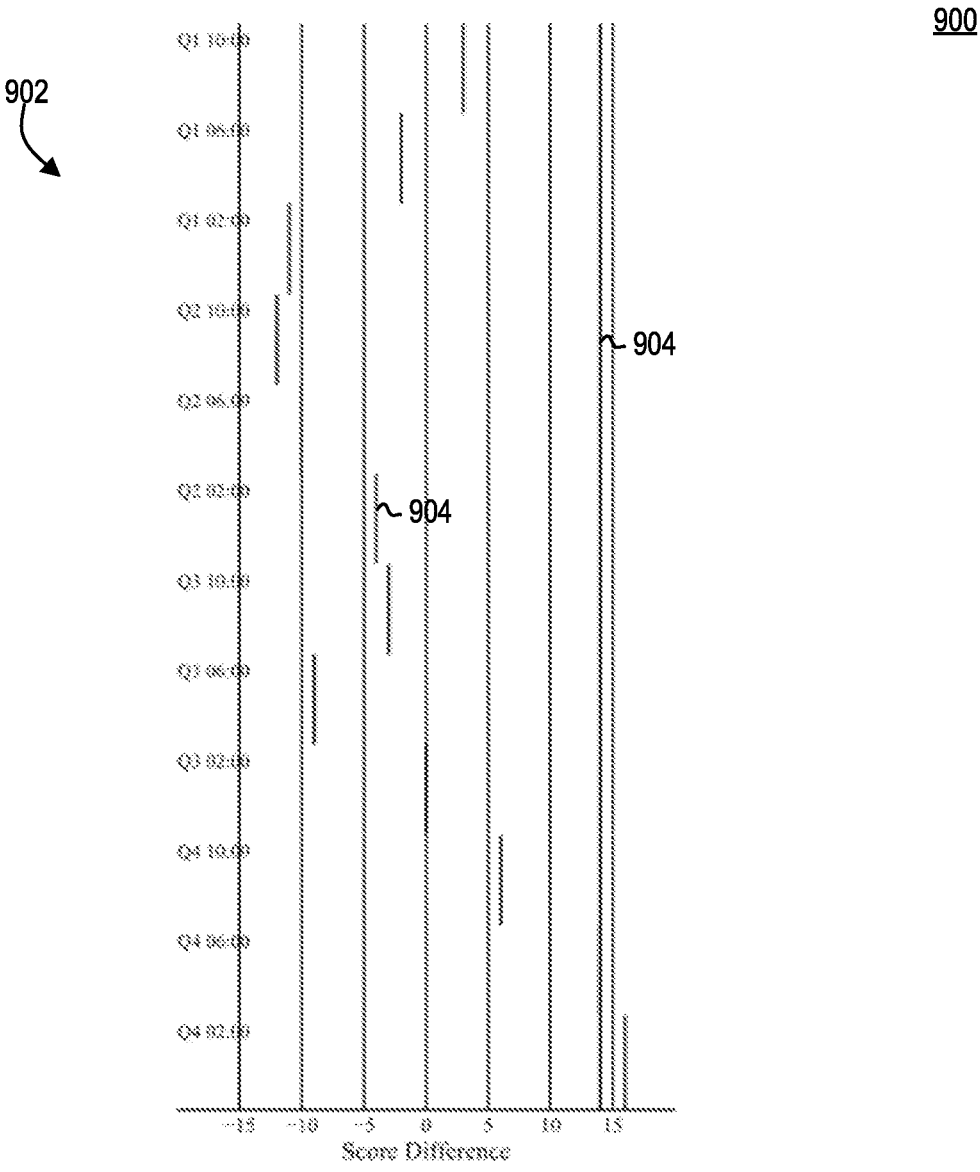


FIG. 9

1000

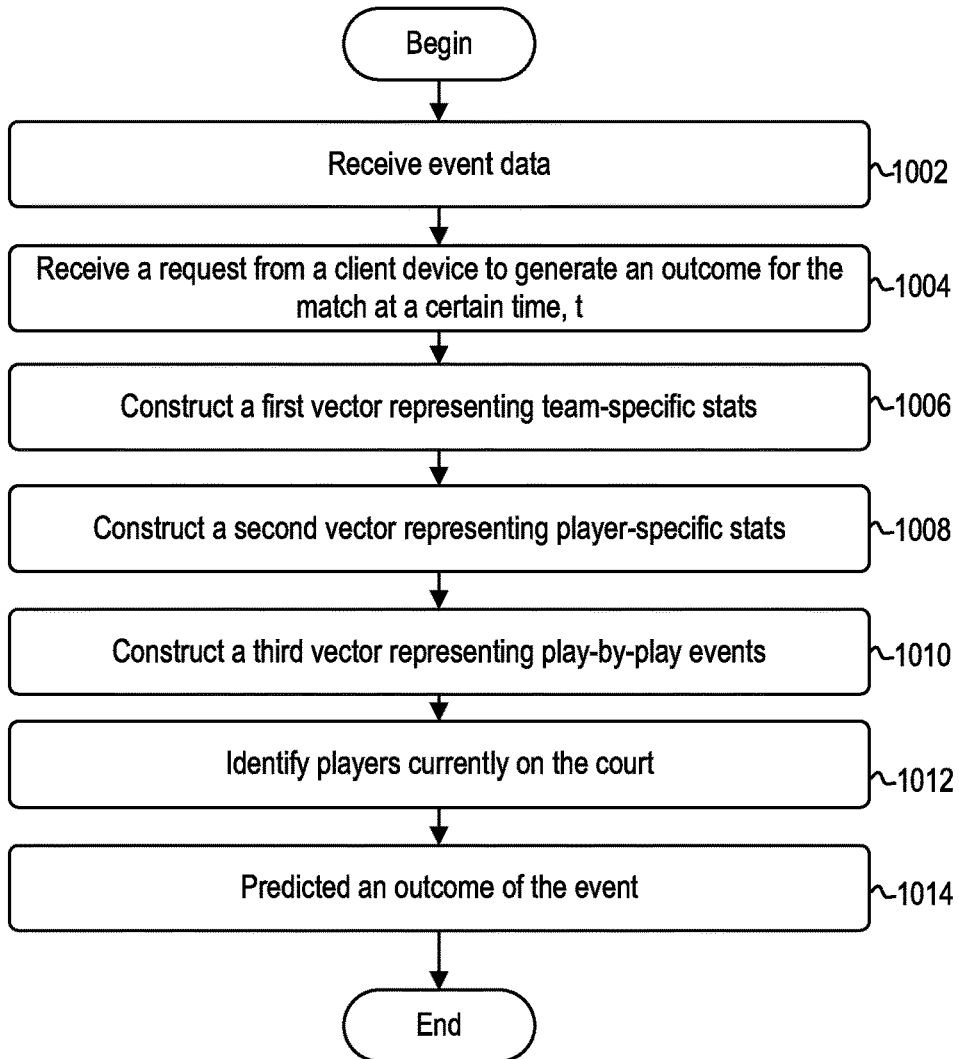
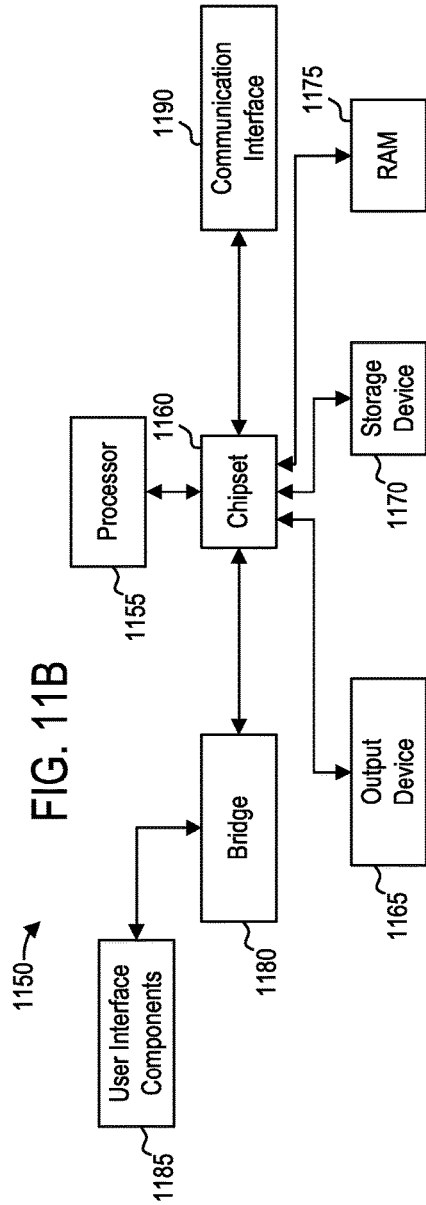
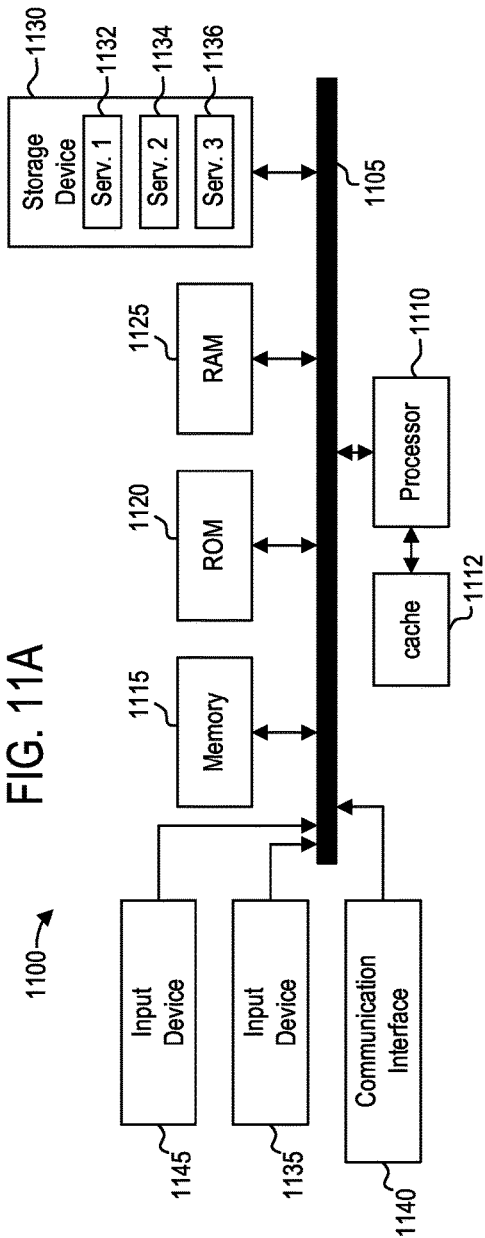


FIG. 10



**METHOD AND SYSTEM FOR
INTERACTIVE, INTERPRETABLE, AND
IMPROVED MATCH AND PLAYER
PERFORMANCE PREDICTIONS IN TEAM
SPORTS**

CROSS-REFERENCE TO RELATED
APPLICATIONS

[0001] This application claims reference to U.S. Provisional Application Ser. No. 62/619,894, filed Jan. 21, 2018, which is hereby incorporated by reference in its entirety.

FIELD OF THE DISCLOSURE

[0002] The present disclosure generally relates to system and method for generating a predictive model and, more specifically, a system and method for predicting pre-match and in-match outcomes.

BACKGROUND

[0003] Increasingly, sports fans and data analysts have become entrenched in sports analytics, particularly in predicting the outcome of a match both prior to initiation of the match and during match play. For example, leading up to a big match, sports networks continually host debate shows in which multiple parties argue over which team will win an upcoming match. Such analysis does not stop once the match ends, however. Typical “Monday Morning Quarterback” sportscasters argue over how the outcome of a match could have changed if, for example, the coach made one or more roster adjustments. Accordingly, there is a continual competition for developing a system that can more accurately predict an outcome of a sporting event.

SUMMARY

[0004] Embodiments disclosed herein generally relate to a system, method, and computer readable medium for generating an outcome for a sporting event is disclosed herein. A computing system retrieves event data from a data store. The event data includes play-by-play information for a plurality of events across a plurality of season. The computing system generates a predictive model using a deep neural network. The one or more neural networks of the deep neural network generate one or more embeddings comprising team-specific information and agent-specific information based on the event data. The computing system selects, from the event data, one or more features related to a current context of the sporting event. The computing system learns, by the deep neural network, one or more likely outcomes of one or more sporting events based at least on team-specific embeddings, agent-specific embeddings, and the current context of the sporting event. The computing system receives a pre-match lineup for the sporting event. The pre-match lineup includes a plurality of agents for a home team and a plurality of agents for an away team. The computing system generates, via the predictive model, a likely outcome of the sporting event based on historical information of each agent for the home team, each agent for the away team, and team-specific features.

[0005] In another embodiment, a system, method, and computer readable medium for generating an outcome for a sporting event. A computing system retrieves event data from a data store. The event data includes play-by-play information for a plurality of events across a plurality of

season. The computing system generates a predictive model using a deep neural network. A neural network learns one or more players likely to be in each event at each time, t , given time based on lineup features of each team, current state of each event at each time t , and current box score at each time t . The computing system generates a data set that includes the one or more players likely to be in each event at each time t . A mixture density network learns a score difference at each time t , based on the lineup features of each team, the current state of each event at time t , the current box score at each time t , and the data set comprising the one or more players likely to be in each event at each time t . The computing system receives an indication to generate a predicted outcome of the sporting event at a current time, T . The computing system generates, via the predictive model, a final score differential for the sporting event based on lineup features of each team to the sporting event, current state of the sporting event at the time T , current box score at the time T , and current lineup in the sporting event at time T .

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] So that the manner in which the above recited features of the present disclosure can be understood in detail, a more particular description of the disclosure, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrated only typical embodiments of this disclosure and are therefore not to be considered limiting of its scope, for the disclosure may admit to other equally effective embodiments.

[0007] FIG. 1 is a block diagram illustrating a computing environment, according to example embodiments.

[0008] FIG. 2 is a block diagram illustrating a component of the computing environment of FIG. 1, according to example embodiments.

[0009] FIG. 3A is a block diagram illustrating a neural network architecture, according to example embodiments.

[0010] FIG. 3B is a block diagram illustrating a neural network architecture, according to example embodiments.

[0011] FIG. 3C is a block diagram illustrating a neural network architecture, according to example embodiments.

[0012] FIG. 3D is a block diagram illustrating a neural network architecture, according to example embodiments.

[0013] FIG. 4 is a flow diagram illustrating a method of training a deep neural network model, according to example embodiments.

[0014] FIG. 5A is a block diagram illustrating a graphical user interface, according to example embodiments.

[0015] FIG. 5B is a block diagram illustrating a graphical user interface, according to example embodiments.

[0016] FIG. 6 is a flow diagram illustrating a method of predicting a pre-match outcome, according to example embodiments.

[0017] FIG. 7A is a block diagram of a structure of a neural network model, according to example embodiments.

[0018] FIG. 7B is a block diagram illustrating a structure of a mixture density network, according to example embodiments.

[0019] FIG. 8 is a flow diagram illustrating a method of training a deep neural network model, according to example embodiments.

[0020] FIG. 9 is a block diagram illustrating an exemplary graphical user interface, according to example embodiments.

[0021] FIG. 10 is a flow diagram illustrating a method of generating an in-game prediction of a sports event, according to example embodiments.

[0022] FIG. 11A is a block diagram illustrating a computing device, according to example embodiments.

[0023] FIG. 11B is a block diagram illustrating a computing device, according to example embodiments.

[0024] To facilitate understanding, identical reference numerals have been used, where possible, to designate identical elements that are common to the figures. It is contemplated that elements disclosed in one embodiment may be beneficially utilized on other embodiments without specific recitation.

DETAILED DESCRIPTION

[0025] One or more techniques disclosed herein generally relate to a system and a method for generating predicted outcomes of a sporting event (e.g., “match”). For example, one or more techniques disclosed herein are directed to predicting the outcome of a match both prior to initiation of the match and during match play.

[0026] In some embodiments, one or more techniques disclosed herein generate a prediction model utilizing the power of personalized predictions to capture low-level nonlinearities that exist between player performances within a team to generate a predicted outcome, prior to the match initiation. The rationale behind this approach, for example, is that agents and teams may have varying levels of talent, but their talent level, which they can display, may be intrinsically linked to other players on the field, as well as their opponents. Although this approach may be intuitive, it has proved extremely challenging to do this in practice due to the vast array of permutations that exist between players within and between teams. The one or more techniques described herein, however, are able to take advantage of lower-dimensional embeddings that capture these same interactions.

[0027] The one or more techniques described herein utilize the power of deep neural networks (i.e., multiple hidden layers) to identify these embeddings. Identification of these embeddings allow for (1) more accurate match prediction over conventional techniques; (2) data-driven player influence rankings; and (3) interactive “what-if” analysis, which may leverage the interpretability of the input feature space to compare players in specific situations.

[0028] The one or more techniques described herein are drastically different from current match-prediction models. For example, one or more techniques described herein explicitly encode the team lineup information in the feature representation. Such techniques stand in stark contrast to most other match-prediction models, which either forecast the number of goals being scored via a Poisson distribution with variables, which model the offensive and defensive capabilities, or estimate the win, draw, or loss directly. Conventional approaches also focus on “player ratings” estimate the number of goals/points a player can expect, but no conventional system explicitly predicts the outcome of the match—which, ultimately, is a superior label to use to correlate player performance.

[0029] Further, in some embodiments, the one or more techniques disclosed herein relate to in-match game predic-

tion (i.e., after the match has started). Such techniques may allow the system described herein to dynamically generate match predictions, at any time during the match. As illustrated in the following example, conventional systems are unable to accurately predict match outcomes during the course of the game.

[0030] In Game 1 of the 2017 Western Conference Finals between the Golden State Warriors and the San Antonio Spurs, with 7:53 remaining in the third quarter and the Spurs up 78-55, Kawhi Leonard re-injured his left ankle while taking a jump-shot. Following Leonard’s departure from the game, the Warriors went on an 18-0 run, eventually winning the game.

[0031] Intuition states that with Leonard off the court the Spurs are less likely to win; however, conventional approaches are unable to account for the severity of losing Leonard as well as the dominant previous performances of the Warriors over the past several seasons. Win probability models should be responsive to in-game contextual features such as injuries and fouls. Additionally, win probability models should incorporate team identity, i.e. team strength. Conventional systems fails to account for the Warriors’ dominance in prior seasons and demonstrated ability to come back to win in similar situations should be captured by the model.

[0032] Further, conventional win probability models are limited in predicting the likelihood of a single, binary outcome given an in-game scenario. In reality, there are many paths to any one outcome, and the model should highlight this complexity. The fact that such issues are common in win probability estimates highlights a final problem. Still further, there simply remains no publicly available datasets or models against which researchers and analysts can compare information.

[0033] The one or more techniques described herein directly address the one or more limitations of conventional system by providing a system that addresses the issues of context and uncertainty through lineup encoding an explicit prediction of the score difference distribution (e.g., final score difference). For example, by using a mixture density network with lineup encoding, the one or more techniques described herein achieve levels of accuracy unattainable by conventional methods.

[0034] FIG. 1 is a block diagram illustrating a computing environment 100, according to example embodiments. Computing environment 100 may include tracking system 102, organization computing system 104, and one or more client devices 108 communicating via network 105.

[0035] Network 105 may be of any suitable type, including individual connections via the Internet, such as cellular or Wi-Fi networks. In some embodiments, network 105 may connect terminals, services, and mobile devices using direct connections, such as radio frequency identification (RFID), near-field communication (NFC), Bluetooth™, low-energy Bluetooth™ (BLE), Wi-Fi™ ZigBee™, ambient backscatter communication (ABC) protocols, USB, WAN, or LAN. Because the information transmitted may be personal or confidential, security concerns may dictate one or more of these types of connection be encrypted or otherwise secured. In some embodiments, however, the information being transmitted may be less personal, and therefore, the network connections may be selected for convenience over security.

[0036] Network 105 may include any type of computer networking arrangement used to exchange data or informa-

tion. For example, network **105** may be the Internet, a private data network, virtual private network using a public network and/or other suitable connection(s) that enables components in computing environment **100** to send and receive information between the components of environment **100**.

[0037] Tracking system **102** may be positioned in a venue **106**. For example, venue **106** may be configured to host a sporting event that includes one or more agents **112**. Tracking system **102** may be configured to record the motions of all agents (i.e., players) on the playing surface, as well as one or more other objects of relevance (e.g., ball, referees, etc.). In some embodiments, tracking system **102** may be an optically-based system using, for example, a plurality of fixed cameras. For example, a system of six stationary, calibrated cameras, which project the three-dimensional locations of players and the ball onto a two-dimensional overhead view of the court may be used. In some embodiments, tracking system **102** may be a radio-based system using, for example, radio frequency identification (RFID) tags worn by players or embedded in objects to be tracked. Generally, tracking system **102** may be configured to sample and record, at a high frame rate (e.g., 25 Hz). Tracking system **102** may be configured to store at least player identity and positional information (e.g., (x, y) position) for all agents and objects on the playing surface for each frame in a game file **110**.

[0038] Game file **110** may be augmented with other event information corresponding to event data, such as, but not limited to, game event information (pass, made shot, turnover, etc.) and context information (current score, time remaining, etc.).

[0039] Tracking system **102** may be configured to communicate with organization computing system **104** via network **105**. Organization computing system **104** may be configured to manage and analyze the data captured by tracking system **102**. Organization computing system **104** may include at least a web client application server **114**, a pre-processing engine **116**, a data store **118**, and match prediction agent **120**. Each of pre-processing engine **116** and match prediction engine **120** may be comprised of one or more software modules. The one or more software modules may be collections of code or instructions stored on a media (e.g., memory of organization computing system **104**) that represent a series of machine instructions (e.g., program code) that implements one or more algorithmic steps. Such machine instructions may be the actual computer code the processor of organization computing system **104** interprets to implement the instructions or, alternatively, may be a higher level of coding of the instructions that is interpreted to obtain the actual computer code. The one or more software modules may also include one or more hardware components. One or more aspects of an example algorithm may be performed by the hardware components (e.g., circuitry) itself, rather as a result of the instructions.

[0040] Match prediction agent **120** may be configured to generate "personalized predictions" for the outcome of a given match. In some embodiments, match prediction agent **120** may be configured to generate a predicted outcome of a given match, prior to initiation of the match (i.e., the match has not yet started). Accordingly, match prediction agent **120** may generate the predicted outcome based on a projected starting lineup for each team. In some embodiments, match prediction agent **120** may be configured to generate a

predicted outcome of a given match, after initiation of the match (i.e., the match has started). Accordingly, match prediction agent **120** may be configured to predict, dynamically, the outcome of a match at any point *t* during the match. Match prediction agent **120** may utilize a deep learning framework that is trained to learn various team-specific attributes, player-specific attributes, and game context attributes to generate predicted outcomes.

[0041] Data store **118** may be configured to store one or more game files **124**. Each game file **124** may include spatial event data and non-spatial event data. For example, spatial event data may correspond to raw data captured from a particular game or event by tracking system **102**. Non-spatial event data may correspond to one or more variables describing the events occurring in a particular match without associated spatial information. For example, non-spatial event data may correspond to each play-by-play event in a particular match. In some embodiments, non-spatial event data may be derived from spatial event data. For example, pre-processing engine **116** may be configured to parse the spatial event data to derive play-by-play information. In some embodiments, non-spatial event data may be derived independently from spatial event data. For example, an administrator or entity associated with organization computing system may analyze each match to generate such non-spatial event data. As such, for purposes of this application, event data may correspond to spatial event data and non-spatial event data.

[0042] In some embodiments, each game file **124** may further include the home and away team box scores. For example, the home and away teams' box scores may include the number of team assists, fouls, rebounds (e.g., offensive, defensive, total), steals, and turnovers at each time, *t*, during gameplay. In some embodiments, each game file **124** may further include a player box score. For example, the player box score may include the number of player assists, fouls, rebounds, shot attempts, points, free-throw attempts, free-throws made, blocks, turnovers, minutes played, plus/minus metric, game started, and the like. Although the above metrics are discussed with respect to basketball, those skilled in the art readily understand that the specific metrics may change based on sport. For example, in soccer, the home and away teams' box scores may include shot attempts, assists, crosses, shots, and the like.

[0043] Pre-processing agent **116** may be configured to process data retrieved from data store **118**. For example, pre-processing agent **116** may be configured to generate one or more sets of information that may be used to train one or more neural networks associated with pre-match module **202** and/or in-match module **204**. Pre-processing agent **116** may scan each of the one or more game files stored in data store **118** to identify one or more statistics corresponding to each specified data set, and generate each data set accordingly. For example, pre-processing agent **116** may scan each of the one or more game files in data store **118** to identify a line-up of each team and the statistics associated with each player in the line-up.

[0044] Client device **108** may be in communication with organization computing system **104** via network **105**. Client device **108** may be operated by a user. For example, client device **108** may be a mobile device, a tablet, a desktop computer, or any computing system having the capabilities described herein. Users may include, but are not limited to, individuals such as, for example, subscribers, clients, pro-

spective clients, or customers of an entity associated with organization computing system 104, such as individuals who have obtained, will obtain, or may obtain a product, service, or consultation from an entity associated with organization computing system 104.

[0045] Client device 108 may include at least application 126. Application 126 may be representative of a web browser that allows access to a website or a stand-alone application. Client device 108 may access application 126 to access one or more functionalities of organization computing system 104. Client device 108 may communicate over network 105 to request a webpage, for example, from web client application server 114 of organization computing system 104. For example, client device 108 may be configured to execute application 126 to access content managed by web client application server 114. The content that is displayed to client device 108 may be transmitted from web client application server 114 to client device 108, and subsequently processed by application 126 for display through a graphical user interface (GUI) of client device 108.

[0046] FIG. 2 illustrates match prediction agent 120 in greater detail, according to example embodiments. Match prediction agent 120 may include pre-match module 202, in-match module 204, and interface module 206.

[0047] Pre-match module 202 may be configured to predict an outcome of a match, prior to initiation of the match. At a high-level, pre-match module 202 may be configured to predict an outcome of a match based on, for example, a proposed starting lineup of the match. More specifically, pre-match module 202 may be configured to predict an outcome of a match based on, at least, team-specific embeddings, agent-specific embeddings, and match context. Pre-match module 202 may include at least one or more neural networks 210 and one or more fully trained models 212. Each neural network 210 may be configured to learn a specific embedding. For example, pre-match module 202 may include a first neural network to learn team history embeddings, a second neural network to learn agent embeddings, and a third neural network to learn recent agent embeddings. Such embeddings may be provided, as input, to a fourth neural network, which learns how to predict a match outcome based on, for example, team-specific information, player-specific information, and the match context. One or more fully trained models 212 may be generated as a result of a training process via one or more neural networks 210. For example, one or more fully trained models 212 may be used to predict match outcome and perform “what-if” analysis as a result of the training process.

[0048] In-match module 204 may be configured to predict an outcome of a match, after initiation of the match. For example, in-match module 204 may be configured to predict the outcome of the match during any point of the match. In-match module 204 may be able to predict the outcome of a match based on, for example, current game context, team history, and agent history. In-match module 204 may include neural network module 220, random forest classifier 222, mixture density network module 224, and one or more fully trained models 226.

[0049] Neural network module 220 may be configured to predict which agents are in an event (e.g., on the court) at a given time. For example, neural network module 220 may be configured to learn how to predict which agents are in an event based on team-specific information, player-specific

information, and the current game state. In-match module 204 may train neural network module 220 with one or more sets of data from data store 118.

[0050] Random forest classifier 222 may be configured to aid in training neural network module 202. For example, in-match module 204 may train a set of random forest classifier models that use lineup encoding to report internal and external performance of the models. Random forest classifiers may be constructed using ensembles of decision tree classifiers. Decision trees may be configured to learn to make splits, i.e. infer rules, based upon values of the incoming observations in order to identify the value of target variables.

[0051] Mixture density network module 224 may be configured to generate a match prediction based on the predicted lineup from neural network module 202. For example, mixture density network may be configured to learn how to predict match outcomes based on the currently predicted lineup, agent-specific information, team specific information, and the current game state. In-match module 204 may train mixture density network module 224 with one or more sets of data from data store 118.

[0052] One or more fully trained models 212 may be generated as a result of a training process via one or more neural networks 220 and mixture density network 224. For example, one or more fully trained models 212 may be used to predict match outcome at any point during a match.

[0053] Pre-Match Prediction

[0054] FIG. 3A is a block diagram illustrating a neural network architecture 300, according to example embodiments. As illustrated in FIG. 3A, several types of input features may be implemented with neural network architecture: team context input feature 302, full agent performance input 304, recent player performance input 306, and game context input 308.

[0055] Team context input may be generated based on a detailed breakdown of the previous performances that preceded the current match, for both teams. For example, for every game in data store 118, pre-match module 202 may select a subset of the most recent matches (e.g., last twenty matches) played by each of the two teams, ordered by date, aligned in pairs (e.g., most recent game by team A with most recent game by team B, second most recent game by team A with second most recent game by team B, etc.), and characterized by one or more features. Such features may include, but are not limited to:

[0056] atHomeA, atHomeB: binary variable to indicate whether the teams were playing at home in the previous fixture.

[0057] dataA, dataB: days from this previous match to the match to be predicted.

[0058] rankAopp, promotedAopp, relegatedAopp, rankBopp, promotedBopp, relegatedBopp: the ranking, promotion, and relegation indications of the teams each team (team A and team B) faced in the subset of most recent matches.

[0059] strengthDiffA, strengthDiffB: league strength difference between the teams and their previous opponents in the subset of most recent matches.

[0060] winA, drawA, lossA, winB, drawB, lossB: the results of each teams in each match in the subset of the most recent matches.

[0061] scoreA, scoreAopp, egvA, egvAopp, scoreB, scoreBopp, egvB, egvBopp: the number of goals (ac-

tual and expected) for each team and their opponents in the subset of most recent matches.

[0062] CAstyleA, STstyleA, HPstyleA, FTstyleA, DPstyleA, CRstyleA, CAstyleB, STstyleB, HPstyleB, FTstyleB, DPstyleB, CRstyleB: the playing style of each team and their opponents in the subset of most recent matches.

[0063] In some embodiments, these features may be derived from the outputs of a separate STATS framework, e.g., a playing style analysis tool for soccer. Each output may be defined as:

[0064] CAstyleA: an indicator of how much Counter Attack (CA) the home team played in the subset of most recent matches.

[0065] STstyleA: an indicator of how much Sustained Threat (ST) the home team played in the subset of most recent matches.

[0066] PstyleA: an indicator of how much High Press (HP) the home team played in the subset of most recent matches.

[0067] FTstyleA: an indicator of how much Fast Tempo (FT) the home team played in the subset of most recent matches.

[0068] DPstyleA: an indicator of how much Direct Play (DP) the home team played in the subset of most recent matches.

[0069] CRstyleA: an indicator of how much Crossing (CR) the home team played in the subset of most recent matches.

[0070] CAstyleB, STstyleB, etc. the away team counterpart for each of the above.

[0071] Counter Attack, Sustained Threat, High Press, Fast Tempo, Direct Play and Crossing are exemplary playing styles that may be measured.

[0072] Player context input may be generated based on one or more features that describes the starting lineups of each team involved in the match to be predicted. Each agent in the lineup may be characterized using their performance in previous appearances, each of which may be measured using the following indicators:

[0073] minutesPlayed: number of minutes played.

[0074] shots, goals, EGV: number of shots, goals, and EGV produced.

[0075] oBMP+, oBMP-, dBMP, dBMP-: scores for ball distribution (offensive) and ball disruption (defensive) produced.

[0076] passesMade, passesCompleted, passPCT, crossesMade, crossesCompleted, CrossPCT:

[0077] number of passes and crosses made and completed, and completion percentage.

[0078] passesDefended, crossDefended, shotsBlocked: number of passes, crosses, and shots defended.

[0079] foulsMade, foulsReceived: number of fouls made and received.

[0080] saves, ESVFaced: (goalkeepers only) number of saves made, actual and expected.

[0081] on-pitchShotsOpp, on-pitchGoalsOPP, on-pitchEGVOpp: number of shots, gals, and EGV produced by the agent's opposition while on the pitch.

[0082] Recent player performance input may be representative of a subset of player performance input. For example, recent player performance input may be generated based on a subset of recent player matches (e.g., last five matches).

[0083] Game context input may be generated based on one or more features that capture the high-level context around a match to be predicted, thus providing a general overview of the recent (e.g., last five games) and more distant (e.g., previous season) team statistics. For example, game context input may include:

[0084] strengthDiff: the difference between the strength of the domestic league of the two teams. For example, this value may be non-zero in international competitions and zero in domestic competitions.

[0085] rankA, rankB: the final position of the teams in their previous season of the domestic league.

[0086] promotedA, promotedB, relegatedA, relegatedB: binary variables to indicate whether each team was promoted or relegated into the current season

[0087] AGGpointsA, AGGscoreA, AGGscoreAopp, AGGpointsB, AGGscoreB, AGGscoreBopp: average points won, goals scored, and goals conceded. In some embodiments, this metric may be limited to recent team statistics (e.g., last five games).

[0088] The groups of input features described above may represent a high dimensional space (e.g., about 900 inputs) for a model to be trained using thousands of examples. To simplify the learning process, pre-match module **202** may generate one or more embeddings having a lower dimension than the inputs alone. Each embedding may be generated using a separate fully-connected neural network, each of which is described below. For example, as illustrated team context embeddings may be generated via neural network **302**, agent embeddings may be generated via neural network **304**, and recent agent embeddings may be generated via neural network **306**. FIGS. **3B-3D** illustrate each neural network in greater detail below.

[0089] FIG. **3B** is a block diagram illustrating a neural network architecture **330**, according to example embodiments. Neural network architecture **330** may be used to generate team history embeddings. As illustrated, neural network architecture **330** may include input data **332**, a data pre-processing module **334**, a neural network **336**, and output **340**.

[0090] Input data **332** may include one or more sets of team history information. For example, input data **332** may include one or more sets of team history information that include the team context input data discussed above in conjunction with FIG. **3A**. As illustrated, input data **332** may include team history information for both the home team and away team. Such information may include a pre-defined amount of matches for each team. For example, the data illustrated in input data **332** may include data from each teams' **20** previous performances.

[0091] Data pre-processing module **330** may be configured to process input data **332** prior to input to neural network **336**. For example, data pre-processing module **330** may be configured to normalize input data **332**. In some embodiments, data pre-processing module **330** may further be configured to flatten input data **332**.

[0092] Pre-match module **202** may then train neural network **336** via the pre-processed team history information. Neural network **336** may include one or more layers **338₁-338₄**. One or more layers may include three hidden layers **338₁-338₃**. Layer **338₁** may include 200 nodes; layer **338₂** may include 40 nodes; and layer **338₃** may include 15 nodes. Each layer **338₁-338₃** may include rectified linear units as

the activation function. Layer 338₄ (i.e., the last layer in neural network 336) may be a linear layer.

[0093] Neural network 336 may be trained using a combination of mean square error and Adam optimization. The output from neural network 336 may be a set of embeddings directed to team history.

[0094] FIG. 3C is a block diagram illustrating a neural network architecture 350, according to example embodiments. Neural network architecture 350 may be used to generate agent embeddings. As illustrated, neural network architecture 350 may include input data 352, 354, data summation modules 356, 358 data pre-processing module 364, a neural network 366, and output 370.

[0095] Input data 352 may include one or more sets of agent information for each agent on the home team. For example, input data 352 may include one or more sets of agent information that include the agent context input data discussed above in conjunction with FIG. 3A. Such information may reflect all of the performances for each player. For example, input data 352 may include the summation of all performances for each home team agent 1-11.

[0096] Input data 354 may include one or more sets of agent information for each agent on the away team. For example, input data 354 may include one or more sets of agent information that include the agent context input data discussed above in conjunction with FIG. 3A. Such information may reflect all of the performances for each player. For example, input data 354 may include the summation of all performances for each away team agent 1-11.

[0097] Summation module 356 may be configured to sum (or aggregate) all of the values for each home team agent in input data 352. For example, summation module 356 may be configured to add together all of the values for each home team player, thus generating a single data set representing all agents of the home team. Accordingly, summation module 356 may output a set of home lineup features 360 representing one or more agents of the home team.

[0098] Summation module 358 may be configured to sum (or aggregate) all of the values for each away team agent in input data 354. For example, summation module 358 may be configured to add together all of the values for each away team player, thus generating a single data set representing all agents of the away team. Accordingly, summation module 358 may output a set of away lineup features 360 representing one or more agents of the away team.

[0099] Data pre-processing module 364 may be configured to process home lineup features 360 and away lineup features 362 prior to input to neural network 366. For example, data pre-processing module 360 may be configured to normalize the data sets corresponding to home lineup features 360 and away lineup features 362.

[0100] Pre-match module 202 may then train neural network 366 via the pre-processed home lineup features 360 and away lineup features 362. Neural network 366 may include one or more layers 368₁-368₄. One or more layers may include three hidden layers 368₁-368₃. Layer 368₁ may include 200 nodes; layer 368₂ may include 40 nodes; and layer 368₃ may include 15 nodes. Each layer 368₁-368₃ may include rectified linear units as the activation function. Layer 368₄ (i.e., the last layer in neural network 366) may be a linear layer.

[0101] Neural network 366 may be trained using a combination of mean square error and Adam optimization. The Adam optimization algorithm is a method of tuning the

parameters defining a neural network and is an extension of stochastic gradient descent. Optimization algorithms iteratively adjust the parameters of a neural network in order to generate predictions which best match the supplied examples. Adam optimization differs from classical gradient stochastic gradient descent in that it adapts the learning rate for each network parameter rather than using a fixed rate. Adam optimization is widely adopted in the machine learning community because it has demonstrated to achieve good results fast. The output from neural network 366 may be a set of embeddings directed to the one or more agents (i.e., 370).

[0102] FIG. 3D is a block diagram illustrating a neural network architecture 380, according to example embodiments. Neural network architecture 380 may be used to generate recent agent embeddings. As illustrated, neural network architecture 380 may include input data 382, 384, data summation modules 386, 388 data pre-processing module 394, a neural network 396, and output 399.

[0103] Input data 382 may include one or more sets of agent information for each agent on the home team. For example, input data 382 may include one or more sets of agent information that include the agent context input data discussed above in conjunction with FIG. 3A. Such information may reflect a subset of the most recent performances for each player on the home team. For example, input data 382 may include the summation of all performances for each home team agent 1-11 in the last five matches.

[0104] Input data 384 may include one or more sets of agent information for each agent on the away team. For example, input data 384 may include one or more sets of agent information that include the agent context input data discussed above in conjunction with FIG. 3A. Such information may reflect a subset of the most recent performances for each player on the away team. For example, input data 384 may include the summation of all performances for each away team agent 1-11 in the last five matches.

[0105] Summation module 386 may be configured to sum (or aggregate) all of the values for each home team agent in input data 382. For example, summation module 386 may be configured to add together all of the values for each home team player, thus generating a single data set representing all agents of the home team. Accordingly, summation module 386 may output a set of home lineup features 390 representing one or more agents of the home team.

[0106] Summation module 388 may be configured to sum (or aggregate) all of the values for each away team agent in input data 384. For example, summation module 388 may be configured to add together all of the values for each away team player, thus generating a single data set representing all agents of the away team. Accordingly, summation module 388 may output a set of away lineup features 390 representing one or more agents of the away team.

[0107] Data pre-processing module 394 may be configured to process home lineup features 390 and away lineup features 392 prior to input to neural network 396. For example, data pre-processing module 390 may be configured to normalize the data sets corresponding to home lineup features 390 and away lineup features 392.

[0108] Pre-match module 202 may then train neural network 396 via the pre-processed home lineup features 390 and away lineup features 392. Neural network 396 may include one or more layers 398₁-398₄. One or more layers may include three hidden layers 398₁-398₃. Layer 398₁ may

include 200 nodes; layer 398₂ may include 40 nodes; and layer 398₃ may include 15 nodes. Each layer 398₁-398₃ may include rectified linear units as the activation function. Layer 398₄ (i.e., the last layer in neural network 396) may be a linear layer.

[0109] Neural network 396 may be trained using a combination of mean square error and Adam optimization. The output from neural network 396 may be a set of embeddings directed to the recent performance of the one or more agents (i.e., 399).

[0110] The one or more embeddings may be able to perform nonlinear dimensionality reductions of the inputs, but supervised by an external target to ensure that the combination of inputs may be guided by a relevant criterion to the task at interest.

[0111] Referring back to FIG. 3A, as shown, the team embeddings 340, the agent embeddings 370, the recent agent embeddings 399, and game context 308 may be used as input to neural network 310. Neural network 310 may be a fully-connected neural network. Neural network 310 may include two hidden layers 312₁ and 312₂, followed by a softmax function 312₃. Hidden layer 312₁ may be of size 30; hidden layer 312₂ may be of size 10. Each of hidden layer 312₁, 312₂ may use a rectifier linear unit function as its activation function. The final layer, 312₃, may have three outputs, one per possible match outcome (home win, draw, away win), and one or more weights that were calculated by minimizing the average cross-entropy loss over the training data set. The output 314 may be a prediction for a given match (e.g., home win, draw, away win).

[0112] FIG. 4 is a flow diagram illustrating a method 400 of generating a prediction model for predicting pre-match outcomes, according to example embodiments. Method 400 may begin at step 402.

[0113] At step 402, pre-match module 202 may receive one or more sets of information from data store 118. Data store 118 may include spatial event data that captures every touch of the ball, with XY coordinates and time stamps as well as non-spatial event data, i.e., one or more variables describing the events occurring without associated spatial information. Accordingly, pre-match module 202 may be able to reconstruct games both in space and time, and therefore the ability to construct performance indicators of verifying complexity. Such performance indicators may include:

[0114] Traditional statistics: counts of passes, crosses and shots, as well as their outcome (completed/missed, saved/scored, etc.).

[0115] Playing styles: how a team play when in a particular possession. This may be assessed along a number of predefined styles, such as, but not limited to, direct play, counter attack, build up, fast tempo, and high press.

[0116] Expected metrics: the probability that a given shot is scored based on its characteristics. For example, to generate an expected goal value (EGV), the location of a shot, its type, and the characteristic of play preceding the shot may be used to obtain its (EGV).

[0117] Ball movement points: indicators of player performance from the point of view of ball distribution and disruption. Every player possession may be analyzed (e.g., start location, end location, and outcome) and a value may be assigned that estimates the value that it had for their teams. In some embodiments, this value

may be positive for beneficial contributions, or negative for successful ones, and it is evaluated both offensively (oBMP) and defensively (dBMP).

[0118] At step 404, pre-match module 202 may construct one or more data sets for each neural network architecture. Pre-match module 202 may parse the received one or more sets of information to construct a first data set directed to team history features for neural network 336. Pre-match module 202 may parse the received one or more sets of information to construct a second data set directed to agent performance features for neural network 366. Pre-match module 202 may parse the received one or more sets of information to construct a third data set directed to agent performance features for neural network 396.

[0119] At step 406, pre-match module 202 may generate one or more team history embeddings based on team history features. For example, pre-match module 202 may implement neural network 336 to generate the one or more team history embeddings. Neural network 336 may take, as input, one or more metrics related to team history features. Such metrics may include, but are not limited to home team performance in a pre-defined number of events and away team performance in a pre-defined number of events. Such information may be used to train neural network 336 to generate the one or more team history embeddings.

[0120] At step 408, pre-match module 202 may generate one or more team agent embeddings based on agent performance features. For example, pre-match module 202 may implement neural network 366 to generate the one or more agent embeddings. Neural network 366 may take, as input, one or more metrics related to agent performance across all previous matches for each agent on each team. Such information may be used to train neural network 366 to generate the one or more team history embeddings.

[0121] At step 410, pre-match module 202 may generate one or more team recent agent embeddings based on recent agent performance features. For example, pre-match module 202 may implement neural network 396 to generate the one or more agent embeddings. Neural network 396 may take, as input, one or more metrics related to recent agent performance across a subset of the most recent matches for each agent on each team. For example, neural network 396 may receive, as input, one or more metrics related to agent performance in the last five matches. Such information may be used to train neural network 396 to generate the one or more recent team history embeddings.

[0122] At step 412, pre-match module 202 may generate a fourth data set related to game context features. For example, pre-match module 202 may parse the received one or more sets of information to construct to identify one or more features related to game context for use with neural network 310. Such features may include high-level context around each match to be predicted, thus providing a general idea of the recent (e.g., last five matches) and more distant (e.g., previous season) performances.

[0123] At step 414, pre-match module 202 may learn how to predict the outcome of a match based on team history embeddings, agent history embeddings, recent agent embeddings, and match context. For example, neural network 310 may be trained using a combination of team history embeddings, player embeddings, recent player embeddings, and game context as input. The weights of neural network 310 may be calculated by, for example, reducing (e.g., minimizing) the average cross-entropy loss over the data set.

[0124] At step 416, pre-match module 202 may compare the predicted outcome of each match to the actual outcome of each match. For example, trajectory agent 120 may utilize a cross-entropy loss and Adam optimizer to minimize the error between the inferred outcome (i.e., output from neural network 310) and the actual outcome (e.g., from data store 118).

[0125] FIG. 5A is a block diagram illustrating a graphical user interface (GUI) 500, according to example embodiments. GUI 500 may be generated by interface module 206. In some embodiments, GUI 500 may be made available to one or more end users through application 126. In some embodiments, match prediction agent 120 may transmit GUI 500 to one or more client devices 106, via application 126, such that each client device 106 may render and display GUI 500.

[0126] GUI 500 may include a visual representation of a current starting lineup 501 for a team for a match. As illustrated, GUI 500 may include a visual representation of each agent 502₁-502₁₁ (generally “agent 502”) in the current starting lineup, as well as a predicted outcome 504 of an upcoming match based on this current starting lineup. Via GUI 500, end users may substitute an agent in the current starting lineup to generate a proposed starting lineup. For example, as illustrated, an end user may replace agent 502₁ with a new agent 506.

[0127] FIG. 5B is a block diagram illustrating a graphical user interface (GUI) 550, according to example embodiments. GUI 550 may be generated by interface module 206 after a user substitutes an agent 502 in FIG. 5A with a new agent. In some embodiments, GUI 550 may be made available to one or more end users through application 126. In some embodiments, match prediction agent 120 may transmit GUI 550 to one or more client devices 106, via application 126, such that each client device 106 may render and display GUI 550.

[0128] When user replaces agent 502₁ with new agent 506, match prediction agent 120 may generate a new predicted outcome based on an adjusted starting lineup 551. For example, match prediction agent 120 may parse the received input and may generate a new pre-match outcome prediction 554 based on the adjusted started lineup. Accordingly, interface agent 156 may generate an updated GUI 530 that reflects the adjusted starting lineup with an updated predicted outcome.

[0129] FIG. 6 is a flow diagram illustrating a method 600 of generating a pre-match prediction, according to example embodiments. Method 600 may begin at step 602.

[0130] At step 602, match prediction agent 120 may generate a graphical user interface that includes a current lineup for a team for a given match. For example, interface module 206 may generate GUI 500 that illustrates a current starting lineup for a particular match. Current starting lineup may include one or more agents that are selected to start the particular match.

[0131] At step 604, match prediction agent 120 may transmit GUI 500 to client device 106. For example, match prediction agent 120 may transmit GUI 500 to client device 106 via application 126 executing thereon. Client device 106 may receive GUI 500, render GUI 500, and display GUI 500 for one or more end users.

[0132] At step 606, match prediction agent 120 may receive, via GUI 500, an indication to change a first agent in

the current lineup to a second agent. For example, match input agent may receive an indication to substitute agent 502₁ for agent 506.

[0133] At step 608, in response to receiving the indication, match prediction agent 120 may generate a predicted outcome of the match, based on the adjusted starting lineup. For example, pre-match module 202 may leverage a prediction model to generate the predicted outcome of the event. In some embodiments, pre-match module 202 may gather team-specific information, agent specific information, and game context, and provide such information to a neural network represented by neural network architecture 300.

[0134] At step 610, match prediction agent 120 may update GUI 500 to include the generated predicted outcome. For example, interface module 206 may generate GUI 550 that includes graphical representations of the adjusted lineup and the new predicted outcome.

[0135] In-Match Prediction

[0136] As recited above, in-match module 204 may be configured to generalize personalized predictions for the outcome of a given match, during gameplay. For example, in-match module 204 may be configured to predict the outcome of a match, based on a combination of current game context, team context (both home and away), agent context (both home and away), and current agents in the match.

[0137] FIG. 7A is a block diagram illustrating a neural network architecture 700 of neural network 220, according to example embodiments. Neural network architecture 700 may be used to predict the line-ups of each team that are currently in a match.

[0138] To train neural network 220, a data set may be generated. The data set may include data from over 8 million basketball play-by-play events over the course of several seasons. Each play-by-play event, P_t , may be described by game time, ball possession, and score differences, representing the base feature set. In some embodiments, the base feature set may be enhanced by the inclusion of home and away team identity, as well as event code (i.e., P_t^+).

[0139] As illustrated in FIG. 7A, P_t , X_t , and L_t may be used as input to neural network 220. X_t may represent the box scores for each of the home and away teams. For example, X_t may represent each team’s box score aggregated in the game up to the time t . Such information may include, but is not limited to, team assists, team blocks, team fouls, team rebounds (e.g., offensive, defensive, and total), team steals, team turnovers, and the like.

[0140] L_t may represent each team’s pre-game lineup for each game. For example, a lineup vector for each game ($L_{i \in \{H,A\}}^{j \in \{0 \dots n\}}$) may be constructed for each player (j) on each team (i) and includes player identity, starter and availability flags, season to date games played, games started, minutes played, plus-minus, minutes per game, plus-minus per game, fouls per game, and the like. Team lineups may be formed from the union of n -such vectors, with padding of empty vectors (e.g., zeros) for rosters of less than n -players. L_t may be the concatenation of home and away lineups.

[0141] Neural network 220 may be set with a task of predicting which players are on the court (e.g., O_t) at every game time, t , given the lineup features L_t , current game state P_t , and box score X_t .

[0142] Neural network 220 may include four fully connected encoding layers with ReLu activation. For example, neural network 220 may include a first layer 702 having 256 units, a second layer 704 having 256 units, a third layer 706

having 128 units, and a fourth layer **708** having 10 units. Although not shown, neural network **220** may include a dropout layer following each layer **704-708**. The last encoding layer (i.e., fourth layer **708**) may give the encoding features E_r to be used by other models. Random forest classifiers **222** may be trained to use the encoding features, E_r , and report the internal and external performance of these models.

[0143] To predict O_r , neural network **220** may decode the encoded data with two fully connected layers with 15 units and ReLU activation (i.e., layer **210**) and 30 units with sigmoid activation (e.g., layer **212**), respectively. To train neural network **220**, in-match module **204** may reduce (e.g., minimize) the cross-entropy via backpropagation using Adam optimization.

[0144] Because, in some embodiments (e.g., basketball), there are only ten valid on-court flags, in-match module may weight the cross-entropy to avoid the trivial solution of all agents being off the court.

[0145] FIG. 7B is a block diagram illustrating mixture density network architecture **750** of mixture density network **224**, according to example embodiments. Mixture density network **224** may be trained to predict the score difference distribution (e.g., final score difference distribution) in a given match. Use of mixture density network **224** allows for use of a combination of neural network architectures and back propagation to find an optimal set of mixture parameters.

[0146] As shown, mixture density network **224** may receive, as input, L_r , X_r , O_r , and P_r . Mixture density network **224** may include a fully connected layer **752**, a batch normalization layer **756**, a dense layer **758**, and an output layer **760**. Fully connected layer **752** may have a tan h activation function and 128 units. Batch normalization layer **756** may be positioned at the output of fully connected layer **752**. In-match module **204** may re-inject P_r into mixture density network via dense layer **758**. Dense layer **758** may be positioned at an output of batch normalization layer **756**. Dense layer **758** may include a tan h activation function and 64 units. Following dense layer **758** may be output layer **760**. Output layer **760** may include one or more linear activations.

[0147] Outputs from mixture density network **224** may be the parameters of a mixture of $N=10$ Gaussian distributions, such that the end of game score difference, S_{t_f} , may have a distribution of:

$$Q(s = S_{t_f}) = \sum_i \pi_i \mathcal{N}(\mu_i, \sigma_i)$$

where π_i may be the weight of the i^{th} distribution, and μ_i and σ_i may be the mean and standard deviation of the i^{th} Gaussian \mathcal{N} , respectively, and $s = \text{Home Team Score} - \text{Away Team Score}$. During training, in-match module **204** may reduce (e.g., minimize) the negative log likelihood, $-\log \{ Q(s = S_{t_f}) \}$, to find the optimal set of mixture parameters $\{ \pi_i, \mu_i, \sigma_i \}$, $i \in [0, N]$.

[0148] Accordingly, rather than generate an output that provides a result that is home win, draw, away win, mixture density network **224** may be configured to generate a projected score difference between the away team and the

home team. By predicting the score difference distribution, in-match module **204** may allow for the potential of various outcomes and measures of uncertainty. In other words, in-match module **204** may formulate the task of outcome prediction as a one (game state) to many (possible score differences) problem.

[0149] FIG. 8 is a flow diagram illustrating a method **800** of generating a prediction model for predicting in-match outcomes, according to example embodiments. Method **800** may begin at step **802**.

[0150] At step **802**, in-match module **204** may receive one or more sets of information from data store **118**. Data store **118** may include spatial event data that captures every touch of the ball, with XY coordinates and time stamps and non-spatial event data that captures one or more variables describing the events occurring without associated spatial information. For example, in-match module **204** may identify a data set for training the in-match prediction model. For example, in-match module **204** may compile a data set that includes over 8.7 million basketball play-by-play events from the 2002-03 through 2016-17 seasons.

[0151] At step **804**, in-match module **204** may construct a first vector representing team-specific statistics. In-match module **204** may generate the first vector by parsing the one or more sets of information received from data store **118**, and identifying those sets of information that are directed to team metrics. For example, in-match module **204** may construct a first vector, X_r , that includes each team's box score. Each team's box score may be aggregated up to a time, t , and includes team assists, team fouls, team rebounds (e.g., offensive, defensive, and total), team steals, and team turnovers.

[0152] At step **806**, in-match module **204** may construct a second vector representing agent-specific statistics. In-match module **204** may generate the second vector by parsing the one or more sets of information received from data store **118**, and identifying those sets of information that are directed to individual, agent metrics. In-match module **204** may construct a second vector, L_r , which may represent each team's in-game lineup for each game. For example, a lineup vector for each game ($l_{i=\{H,A\}}^{j=\{0 \dots n\}}$) may be constructed for each player (j) on each team (i) and includes player identity, starter and availability flags, season to date games played, games started, minutes played, plus-minus, minutes per game, plus-minus per game, fouls per game, and the like. Team lineups may be formed from the union of n -such vectors, with padding of empty vectors (e.g., zeros) for rosters of less than n -players. L_r may be the concatenation of home and away lineups.

[0153] At step **808**, in-match module **204** may construct a third vector representing one or more play-by-play events across all matches in the one or more sets of information. For example, in-match module **204** may generate a third vector, P_r , which includes one or more play-by-play events that may be described by game time, ball possession, and score differences, representing the base feature set. In some embodiments, the base feature set may be enhanced by the inclusion of home and away team identity, as well as event code (i.e., P_r^+).

[0154] At step **810**, in-match module **204** may predict one or more players current in the match (e.g., on the court) at each time, t . In-match module **204** may train neural network **220** to predict the one or more players currently in the match. For example, neural network **220** may be set with a task of

predicting which players are on the court (e.g., O_t) at every game time, t , given the lineup features L_t , current game state P_t , and box score X_t . To train neural network **220**, in-match module **204** may reduce (e.g., minimize) the cross-entropy via backpropagation using Adam optimization. Accordingly, the output from neural network **220** may be the vector, O_t , that represents the one or more agents in the match, for both teams, at each time, t .

[0155] At step **812**, In-match module **204** may generate a predicted final score for each match at each time, t . In-match module **204** may train mixture density network **224** to predict the score difference distribution in a given match. Use of mixture density network **224** allows for use of a combination of neural network architectures and back propagation to find an optimal set of mixture parameters. Mixture density network **224** may receive, as input, L_t , X_t , O_t , and P_t . Accordingly, the output from mixture density network **224** may be one or more sets of score differentials during each time, t , of each match.

[0156] At step **814**, in-match module **204** may reduce (or minimize) the likelihood of finding the optimal set. Outputs from mixture density network **224** may be the parameters of a mixture of $N=10$ Gaussian distributions, such that the end of game score difference, S_{t_f} , may have a distribution of:

$$Q(s = S_{t_f}) = \sum_i^N \pi_i \mathcal{N}(\mu_i, \sigma_i)$$

where π_i may be the weight of the i^{th} distribution, and μ_i and σ_i may be the mean and standard deviation of the i^{th} Gaussian \mathcal{N} , respectively, and $s=Home\ Team\ Score-Away\ Team\ Score$. During training, in-match module **204** may reduce (e.g., minimize) the negative log likelihood, $-\log \{Q(s=S_{t_f})\}$, to find the optimal set of mixture parameters $\{\pi_i, \mu_i, \sigma_i\}$, $i \in [0, N]$.

[0157] FIG. 9 is a block diagram illustrating a graphical user interface (GUI) **900**, according to example embodiments. GUI **900** may be generated by interface module **206**. In some embodiments, GUI **900** may be made available to one or more end users through application **126**. In some embodiments, match prediction agent **120** may transmit GUI **900** to one or more client devices **106**, via application **126**, such that each client device **106** may render and display GUI **900**.

[0158] GUI **900** may include graph **902**. Graph **902** may depict a plot $Q_t(s=S_{t_f})$ for twelve different game times, t . As illustrated, t may be plotted along the y-axis and the predicted score difference may be plotted along the x-axis. As illustrated, the predictions generated by in-match module **204** may be multi-modal, mirroring the myriad possible outcomes a game may have. As illustrated, initial predictions may be matchup specific. As further illustrated, as the match progresses and the match state changes (i.e., one or more lines **904**), the model distributions may evolve as well, often showing state switching and mean shifting over time. As such, the evolution of $Q_t(s=S_{t_f})$ may be context and matchup specific. The one or more predictions may oscillate between two initial states as the score difference oscillates. The actual score may be represented by solid line **906**.

[0159] As shown, the distributions may not collapse or narrow with time. The apparent insensitivity of the distri-

bution variance to game time may be a function of the Markovian nature of the current prediction architecture. The one or more models implemented by in-match module **204** may not have a sense of how much time remains in a match, only that some amount of time has passed.

[0160] FIG. 10 is a flow diagram illustrating a method **1000** of generating an in-match predicted score of a match using the prediction model generated in FIG. 8, according to example embodiments. Method **1000** may begin at step **1002**.

[0161] At step **1002**, in-match module **204** may receive, from event data. For example, in-match module **204** may receive, from tracking system **102**, in real-time, near real-time, or periodically one or more sets of event data of a match currently in progress. Another example, in-match module **204** may receive, from one or more computing systems, in real-time, near real-time, or periodically one or more sets of event data derived from an entity associated with organization computing system **104**. Such event data may include one or more features of match play (e.g., play-by-play events). Although method **1000** discusses use of live (or near-live) data, those skilled in the art understand that such operations may be performed with historical data.

[0162] At step **1004**, in-match module **204** may receive a request from a client device **106** to generate a predicted score for the match at a certain time, t , such as, but not limited to, predicting a final score of the match at each time, t , during the match. For example, in-match module **204** may receive the request from client device **106** via application **126** executing thereon. Although method **1000** includes operation **1004**, those skilled in the art may understand that in-match module **204** need not wait for an explicit request from an end-user to generate a predicted score of a match; rather, in-match module **204** may dynamically generate one or more predicted scores on-demand, as the match progresses.

[0163] At step **1006**, in-match module **204** may construct a first vector representing team-specific statistics. In-match module **204** may generate the first vector by parsing the event data, and identifying those sets of information that are directed to team metrics. For example, in-match module **204** may construct a first vector, X_t , that includes each team's box score up to time t . Each team's box score may include, but is not limited to, team assists, team fouls, team rebounds (e.g., offensive, defensive, and total), team steals, and team turnovers.

[0164] At step **1008**, in-match module **204** may construct a second vector representing player specific statistics. In-match module **204** may generate the second vector by parsing the event data received to identify one or more agents in each team's lineup. In-match module **204** may retrieve from data store **118** one or more agent-specific statistics associated with each agent. In-match module **204** may construct a second vector, L_t , which may represent each team's pre-game lineup for each game. For example, a lineup vector for each game ($L_{i=\{H,A\}}^{j=\{0 \dots n\}}$) may be constructed for each player (j) on each team (i) and includes player identity, starter and availability flags, season to date games played, games started, minutes played, plus-minus, minutes per game, plus-minus per game, fouls per game, and the like. Team lineups may be formed from the union of n -such vectors, with padding of empty vectors (e.g., zeros) for rosters of less than n -players. L_t may be the concatenation of home and away lineups.

[0165] At step **1010**, in match-module **204** may construct a third vector representing all play-by-play events up to time, t . For example, in-match module **204** may generate a third vector, P_t , which includes one or more play-by-play events that may be described by game time, ball possession, and score differences, representing the base feature set. In some embodiments, the base feature set may be enhanced by the inclusion of home and away team identity, as well as event code (i.e., P_t^+).

[0166] At step **1012**, in-match module **204** may identify one or more agents currently in the match (e.g., one or more agents currently on the court). In some embodiments, in-match module **204** may identify one or more agents currently in the match by parsing the play-by-play events up to time, t . In some embodiments, in-match module **204** may predict which agents are currently in the match utilizing trained neural network **220** to predict the one or more players currently in the match. For example, neural network **220** may predict which players are on the court (e.g., O_t) at time, t , given the lineup features L_t , current game state P_t , and box score X_t . The output from neural network **220** may be the vector, O_t , that represents the one or more agents in the match, for both teams, at each time, t .

[0167] At step **1014**, in-match module **204** may generate a predicted outcome of the match. For example, in-match module **204** may generate a predicted final score based on the information generated at time, t . In-match module **204** may provide, as input, to trained mixture density network **224** L_t , X_t , O_t , and P_t . The output from mixture density network **224** may be one or more sets of score differentials during each time, t , of each match.

[0168] FIG. 11A illustrates a system bus computing system architecture **1100**, according to example embodiments. System **1100** may be representative of at least a portion of organization computing system **104**. One or more components of system **1100** may be in electrical communication with each other using a bus **1105**. System **1100** may include a processing unit (CPU or processor) **1110** and a system bus **1105** that couples various system components including the system memory **1115**, such as read only memory (ROM) **1120** and random access memory (RAM) **1125**, to processor **1110**. System **1100** may include a cache of high-speed memory connected directly with, in close proximity to, or integrated as part of processor **1110**. System **1100** may copy data from memory **1115** and/or storage device **1130** to cache **1112** for quick access by processor **1110**. In this way, cache **1112** may provide a performance boost that avoids processor **1110** delays while waiting for data. These and other modules may control or be configured to control processor **1110** to perform various actions. Other system memory **1115** may be available for use as well. Memory **1115** may include multiple different types of memory with different performance characteristics. Processor **1110** may include any general purpose processor and a hardware module or software module, such as service **1** **1132**, service **2** **1134**, and service **3** **1136** stored in storage device **1130**, configured to control processor **1110** as well as a special-purpose processor where software instructions are incorporated into the actual processor design. Processor **1110** may essentially be a completely self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor may be symmetric or asymmetric.

[0169] To enable user interaction with the computing device **1100**, an input device **1145** may represent any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output device **1135** may also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems may enable a user to provide multiple types of input to communicate with computing device **1100**. Communications interface **1140** may generally govern and manage the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

[0170] Storage device **1130** may be a non-volatile memory and may be a hard disk or other types of computer readable media which may store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, solid state memory devices, digital versatile disks, cartridges, random access memories (RAMs) **1125**, read only memory (ROM) **1120**, and hybrids thereof.

[0171] Storage device **1130** may include services **1132**, **1134**, and **1136** for controlling the processor **1110**. Other hardware or software modules are contemplated. Storage device **1130** may be connected to system bus **1105**. In one aspect, a hardware module that performs a particular function may include the software component stored in a computer-readable medium in connection with the necessary hardware components, such as processor **1110**, bus **1105**, display **1135**, and so forth, to carry out the function.

[0172] FIG. 11B illustrates a computer system **1150** having a chipset architecture that may represent at least a portion of organization computing system **104**. Computer system **1150** may be an example of computer hardware, software, and firmware that may be used to implement the disclosed technology. System **1150** may include a processor **1155**, representative of any number of physically and/or logically distinct resources capable of executing software, firmware, and hardware configured to perform identified computations. Processor **1155** may communicate with a chipset **1160** that may control input to and output from processor **1155**. In this example, chipset **1160** outputs information to output **1165**, such as a display, and may read and write information to storage device **1170**, which may include magnetic media, and solid state media, for example. Chipset **1160** may also read data from and write data to RAM **1175**. A bridge **1180** for interfacing with a variety of user interface components **1185** may be provided for interfacing with chipset **1160**. Such user interface components **1185** may include a keyboard, a microphone, touch detection and processing circuitry, a pointing device, such as a mouse, and so on. In general, inputs to system **1150** may come from any of a variety of sources, machine generated and/or human generated.

[0173] Chipset **1160** may also interface with one or more communication interfaces **1190** that may have different physical interfaces. Such communication interfaces may include interfaces for wired and wireless local area networks, for broadband wireless networks, as well as personal area networks. Some applications of the methods for generating, displaying, and using the GUI disclosed herein may include receiving ordered datasets over the physical interface or be generated by the machine itself by processor **1155**

analyzing data stored in storage **1170** or **1175**. Further, the machine may receive inputs from a user through user interface components **1185** and execute appropriate functions, such as browsing functions by interpreting these inputs using processor **1155**.

[0174] It may be appreciated that example systems **1100** and **1150** may have more than one processor **1110** or be part of a group or cluster of computing devices networked together to provide greater processing capability.

[0175] While the foregoing is directed to embodiments described herein, other and further embodiments may be devised without departing from the basic scope thereof. For example, aspects of the present disclosure may be implemented in hardware or software or a combination of hardware and software. One embodiment described herein may be implemented as a program product for use with a computer system. The program(s) of the program product define functions of the embodiments (including the methods described herein) and can be contained on a variety of computer-readable storage media. Illustrative computer-readable storage media include, but are not limited to: (i) non-writable storage media (e.g., read-only memory (ROM) devices within a computer, such as CD-ROM disks readably by a CD-ROM drive, flash memory, ROM chips, or any type of solid-state non-volatile memory) on which information is permanently stored; and (ii) writable storage media (e.g., floppy disks within a diskette drive or hard-disk drive or any type of solid state random-access memory) on which alterable information is stored. Such computer-readable storage media, when carrying computer-readable instructions that direct the functions of the disclosed embodiments, are embodiments of the present disclosure.

[0176] It will be appreciated to those skilled in the art that the preceding examples are exemplary and not limiting. It is intended that all permutations, enhancements, equivalents, and improvements thereto are apparent to those skilled in the art upon a reading of the specification and a study of the drawings are included within the true spirit and scope of the present disclosure. It is therefore intended that the following appended claims include all such modifications, permutations, and equivalents as fall within the true spirit and scope of these teachings.

What is claimed:

1. A method of generating an outcome for a sporting event, comprising:

retrieving, by a computing system, tracking data from a data store, the tracking data comprising event data for a plurality of events across a plurality of season;

generating, by the computing system, a predictive model using a deep neural network, by:

learning, by a neural network, one or more players likely to be in each event at each time, t , given time based on lineup features of each team, current state of each event at each time t , and current box score at each time t ;

generating a data set comprising the one or more players likely to be in each event at each time t ;

learning, by a mixture density network, a score difference at each time t , based on the lineup features of each team, the current state of each event at time t , the current box score at each time t , and the data set comprising the one or more players likely to be in each event at each time t ;

receiving, by the computing system, an indication to generate a predicted outcome of the sporting event at a time, T ; and

generating, by the computing system via the predictive model, a final score differential for the sporting event based on lineup features of each team to the sporting event, current state of the sporting event at the time T , current box score at the time T , and current lineup in the sporting event at time T .

2. The method of claim **1**, wherein receiving, by the computing system, the indication to generate the predicted outcome of the sporting event at the time, T comprises:

receiving, from a client device, a request to predict the outcome of the sporting event at the time T .

3. The method of claim **1**, wherein learning, by the mixture density network, the score difference at each time t , comprises:

comparing the score difference at each time t to the actual score difference at each time t ; and

minimizing an error between the score difference and the actual score difference using a negative log likelihood of finding an optimal set of parameters.

4. The method of claim **1**, wherein learning, by the neural network, the one or more players likely to be in each event at each time, t , given time based on the lineup features of each team, the current state of each event at each time t , and the current box score at each time t comprises:

learning one or more lineup encoding features.

5. The method of claim **4**, further comprising:

training a set of random forest classifiers with the one or more lineup encoding features.

6. The method of claim **1**, wherein the lineup features of each team are represented as a union of a plurality of lineup vectors constructed for each player on each team.

7. The method of claim **1**, wherein the current state of each event at each time t is represented by a vector comprising each play-by-play event up to the time t and at each time t .

8. A system for predicting the outcome of a sporting event, comprising:

a processor; and

a memory having programming instructions stored thereon, which, when executed by the processor, performs one or more operations comprising:

retrieving tracking data from a data store, the tracking data comprising event data for a plurality of events across a plurality of season;

generating a predictive model using a deep neural network, by:

learning, by a neural network, one or more players likely to be in each event at each time, t , given time based on lineup features of each team, current state of each event at each time t , and current box score at each time t ;

generating a data set comprising the one or more players likely to be in each event at each time t ; and

learning, by a mixture density network, a score difference at each time t , based on the lineup features of each team, the current state of each event at time t , the current box score at each time t , and the data set comprising the one or more players likely to be in each event at each time t ;

- receiving an indication to generate a predicted outcome of the sporting event at a time, T; and
generating, via the predictive model, a final score differential for the sporting event based on lineup features of each team to the sporting event, current state of the sporting event at the time T, current box score at the time T, and current lineup in the sporting event at time T.
- 9.** The system of claim **8**, wherein receiving the indication to generate the predicted outcome of the sporting event at the time, T comprises:
receiving, from a client device, a request to predict the outcome of the sporting event at the time T.
- 10.** The system of claim **8**, wherein learning, by the mixture density network, the score difference at each time t, comprises:
comparing the score difference at each time t to the actual score difference at each time t; and
minimizing an error between the score difference and the actual score difference using a negative log likelihood of finding an optimal set of parameters.
- 11.** The system of claim **8**, wherein learning, by the neural network, the one or more players likely to be in each event at each time, t, given time based on the lineup features of each team, the current state of each event at each time t, and the current box score at each time t comprises:
learning one or more lineup encoding features.
- 12.** The system of claim **8**, wherein the one or more operations further comprise:
training a set of random forest classifiers with the one or more lineup encoding features.
- 13.** The system of claim **8**, wherein the lineup features of each team are represented as a union of a plurality of lineup vectors constructed for each player on each team.
- 14.** The system of claim **8**, wherein the current state of each event at each time t is represented by a vector comprising each play-by-play event up to the time t and at each time t.
- 15.** A non-transitory computer readable medium including one or more sequences of instructions that, when executed by the one or more processors, causes:
retrieving, by a computing system, tracking data from a data store, the tracking data comprising event data for a plurality of events across a plurality of season;
generating, by the computing system, a predictive model using a deep neural network, by:
learning, by a neural network, one or more players likely to be in each event at each time, t, given time based on lineup features of each team, current state of each event at each time t, and current box score at each time t;
generating a data set comprising the one or more players likely to be in each event at each time t;
learning, by a mixture density network, a score difference at each time t, based on the lineup features of each team, the current state of each event at time t, the current box score at each time t, and the data set comprising the one or more players likely to be in each event at each time t;
receiving, by the computing system, an indication to generate a predicted outcome of the sporting event at a time, T; and
generating, by the computing system via the predictive model, a final score differential for the sporting event based on lineup features of each team to the sporting event, current state of the sporting event at the time T, current box score at the time T, and current lineup in the sporting event at time T.
- 16.** The non-transitory computer readable medium of claim **15**, wherein receiving, by the computing system, the indication to generate the predicted outcome of the sporting event at the time, T comprises:
receiving, from a client device, a request to predict the outcome of the sporting event at the time T.
- 17.** The non-transitory computer readable medium of claim **15**, wherein learning, by the mixture density network, the score difference at each time t, comprises:
comparing the score difference at each time t to the actual score difference at each time t; and
minimizing an error between the score difference and the actual score difference using a negative log likelihood of finding an optimal set of parameters.
- 18.** The non-transitory computer readable medium of claim **15**, wherein learning, by the neural network, the one or more players likely to be in each event at each time, t, given time based on the lineup features of each team, the current state of each event at each time t, and the current box score at each time t comprises:
learning one or more lineup encoding features.
- 19.** The non-transitory computer readable medium of claim **18**, further comprising:
training a set of random forest classifiers with the one or more lineup encoding features.
- 20.** The non-transitory computer readable medium of claim **15**, wherein the lineup features of each team are represented as a union of a plurality of lineup vectors constructed for each player on each team.

* * * * *