US 20170032245A1

(19) **United States**

(12) **Patent Application Publication**

Osband et al.

(10) **Pub. No.: US 2017/0032245 A1**

(43) **Pub. Date: Feb. 2, 2017**

(54) **SYSTEMS AND METHODS FOR PROVIDING REINFORCEMENT LEARNING IN A DEEP LEARNING SYSTEM**

(71) Applicant: **The Board of Trustees of the Leland Stanford Junior University**, Stanford, CA (US)

(72) Inventors: **Ian David Moffat Osband**, Stanford, CA (US); **Benjamin Van Roy**, Stanford, CA (US)

(73) Assignee: **The Board of Trustees of the Leland Stanford Junior University**, Stanford, CA (US)

(21) Appl. No.: **15/212,042**

(22) Filed: **Jul. 15, 2016**

Related U.S. Application Data

(63) Continuation-in-part of application No. 15/201,284, filed on Jul. 1, 2016.

(60) Provisional application No. 62/187,681, filed on Jul. 1, 2015.

Publication Classification

(51) **Int. Cl.**

G06N 3/08 (2006.01)

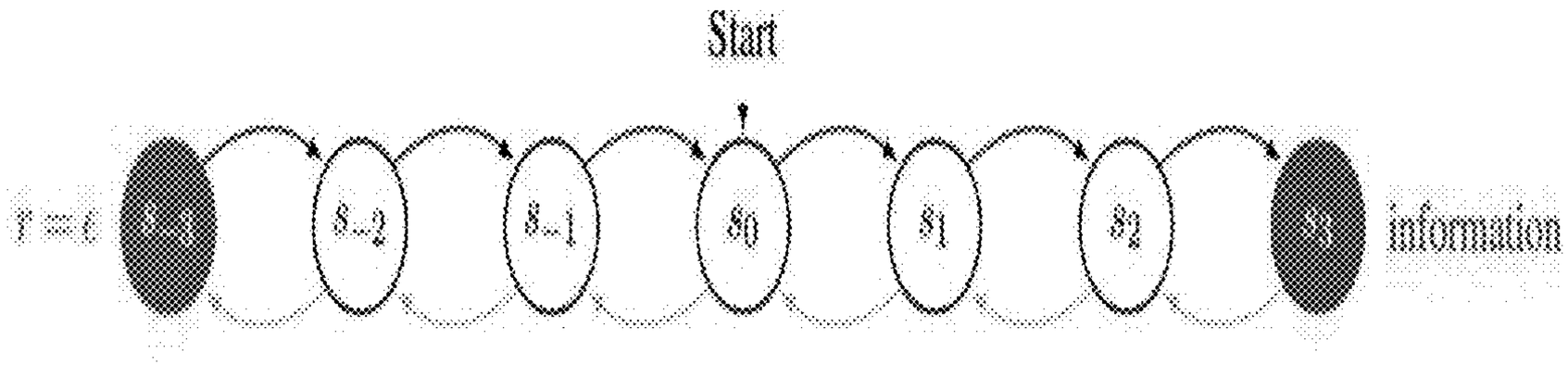
G06N 99/00 (2006.01)

(52) **U.S. Cl.**

CPC ***G06N 3/08*** (2013.01); ***G06N 99/005*** (2013.01)

(57) **ABSTRACT**

Systems and methods for providing reinforcement learning for a deep learning network are disclosed. A reinforcement learning process that provides deep exploration is provided by a bootstrap that applied to a sample of observed and artificial data to facilitate deep exploration via a Thompson sampling approach.



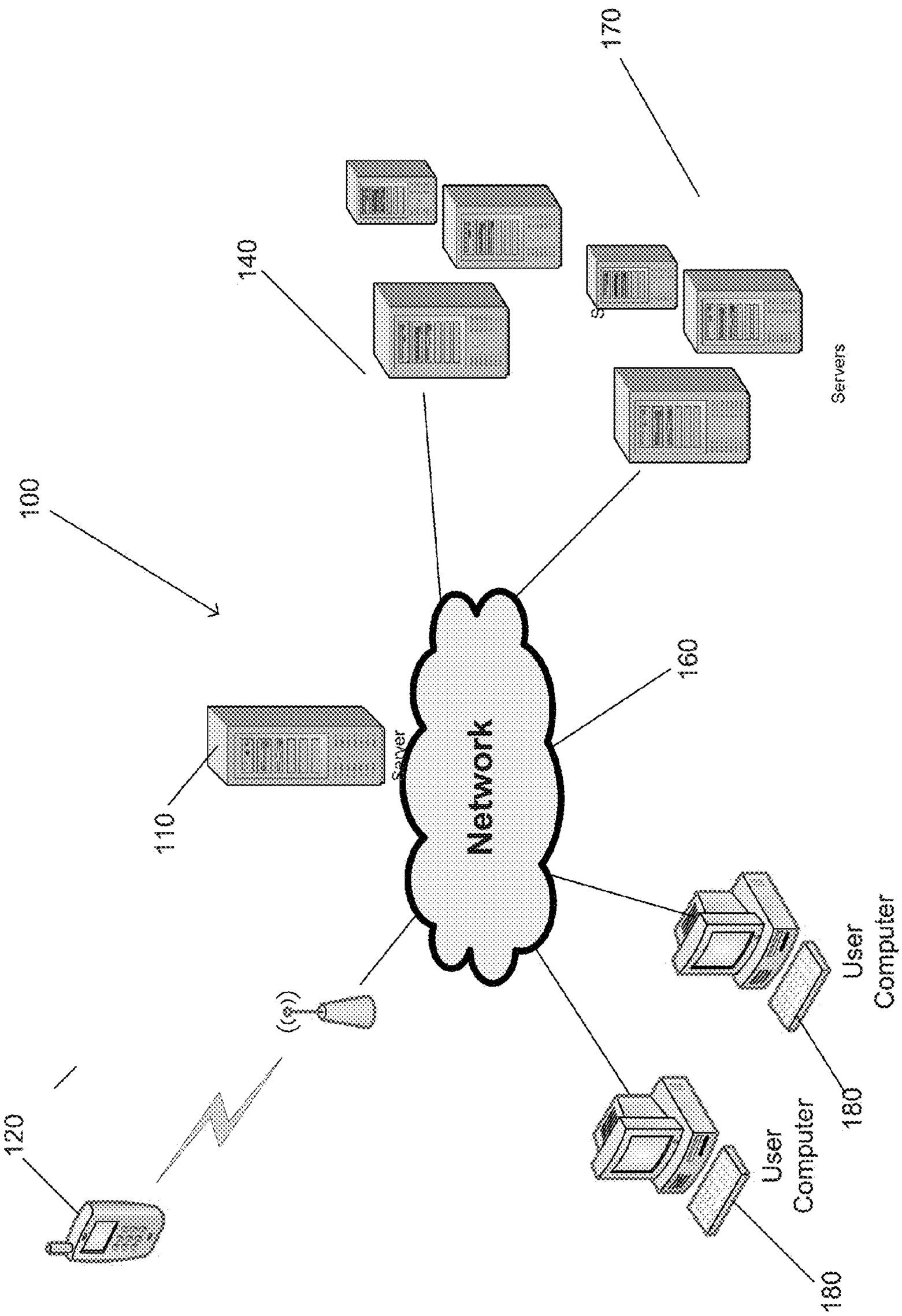

100
110
Server
120
130
140
Servers
150
Network
160
170
180
User Computer
180
User Computer

Figure 1

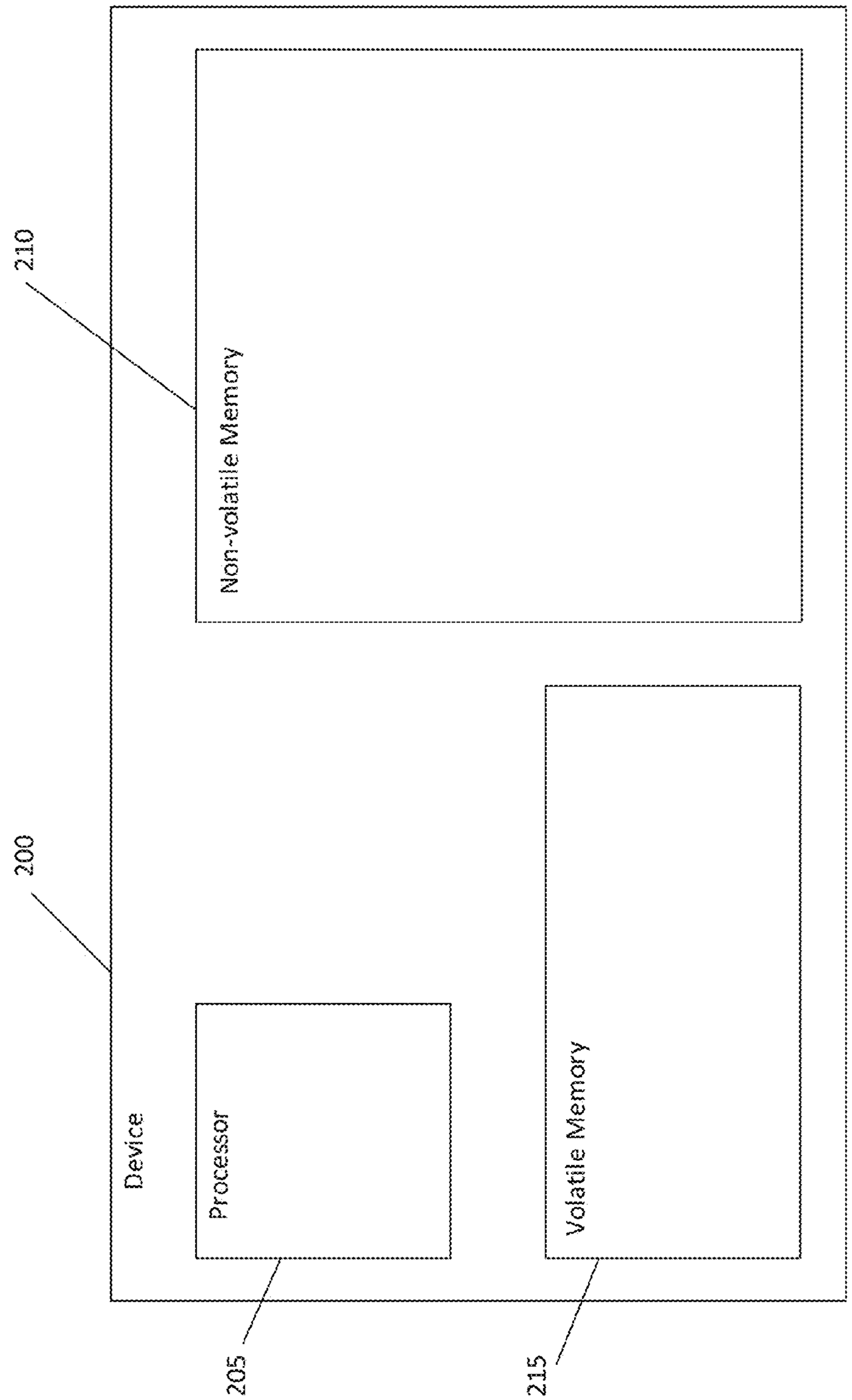

Device
200
Processor
205
Volatile Memory
215
Non-volatile Memory
210

Figure 2

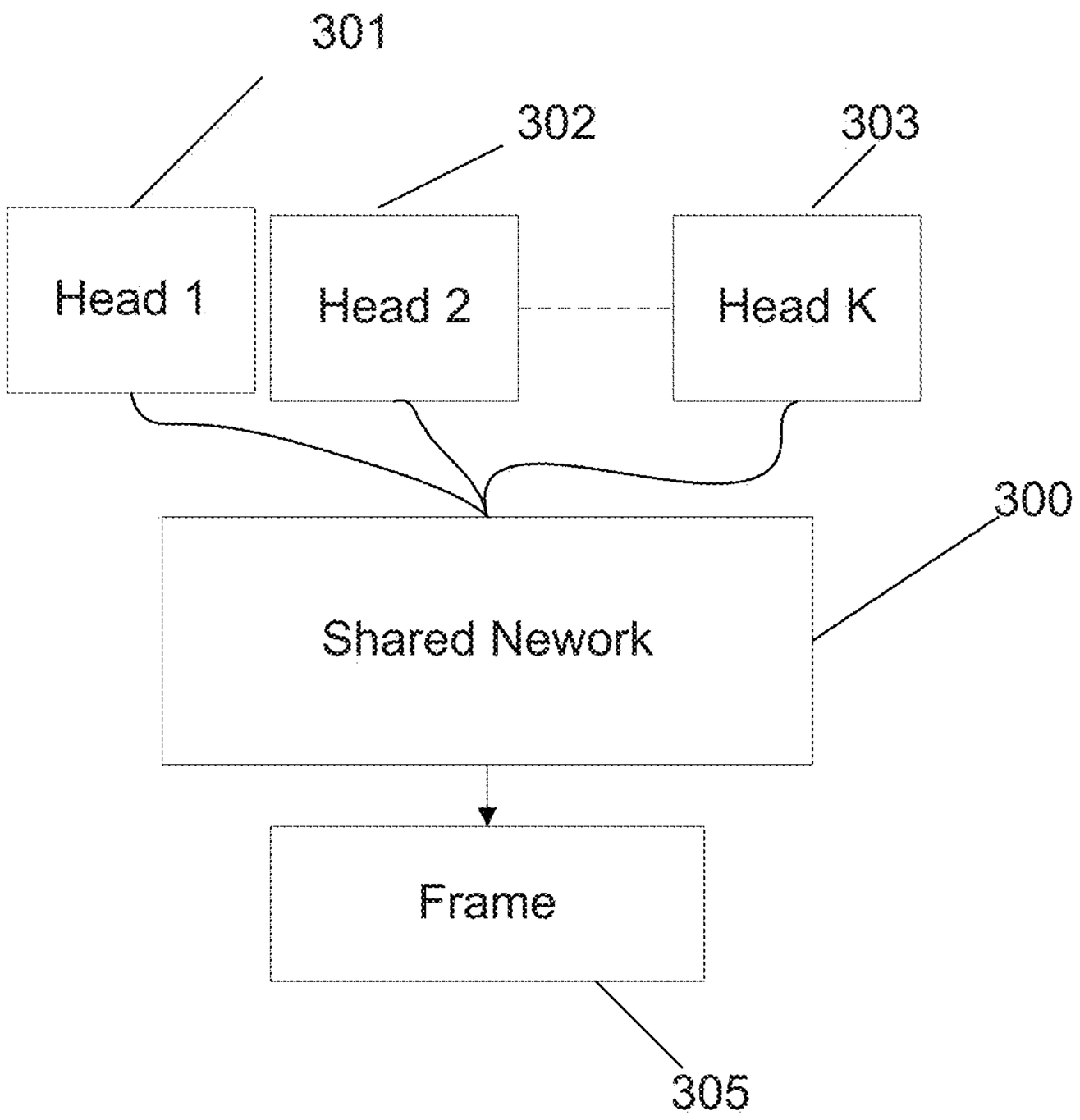


Figure 3

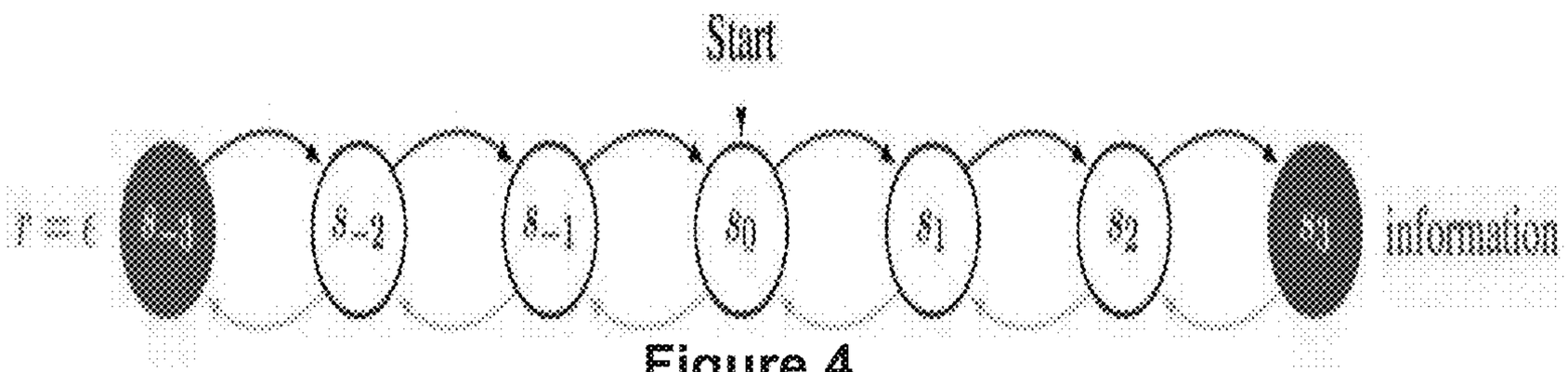


Figure 4

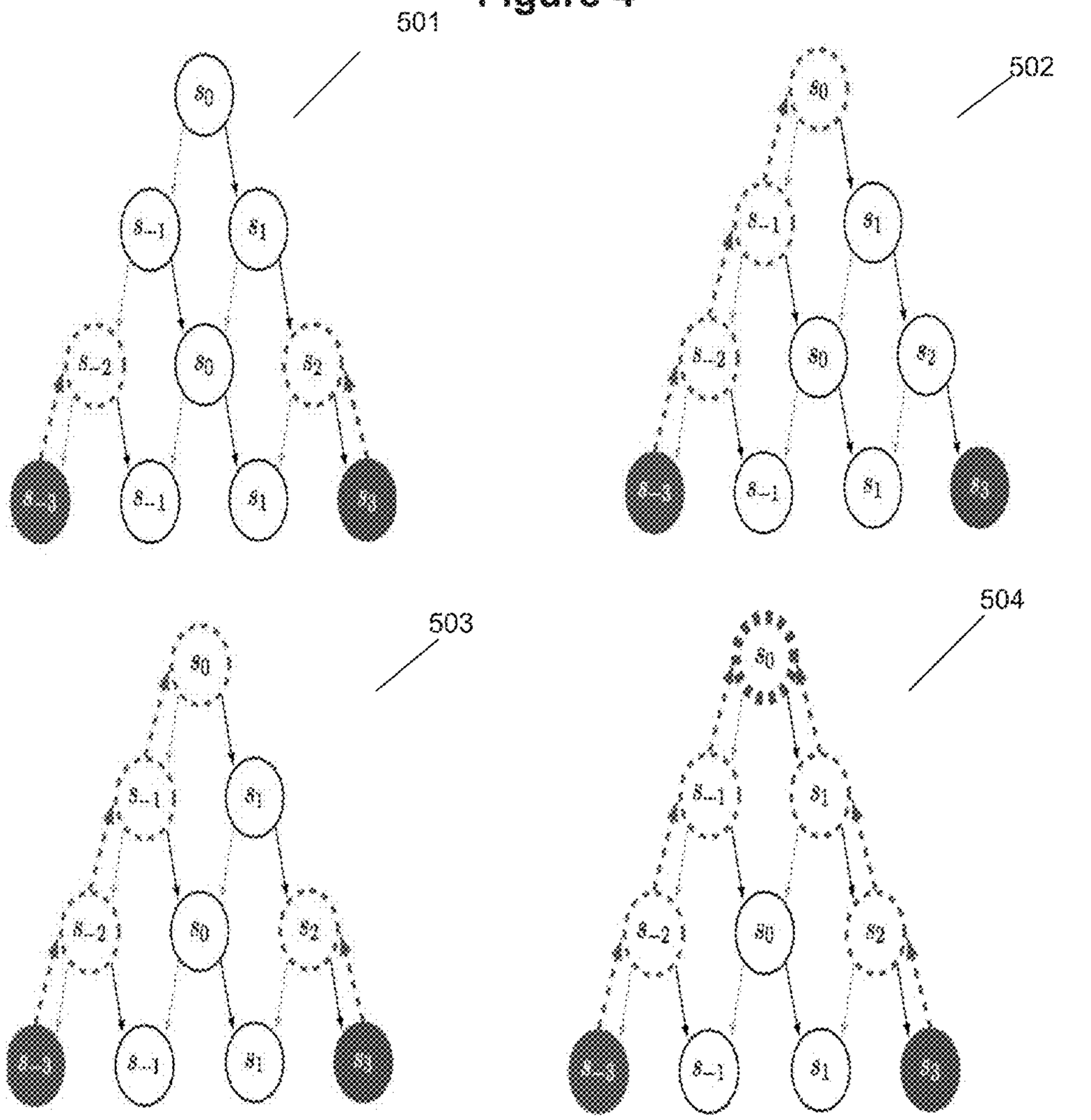


Figure 5

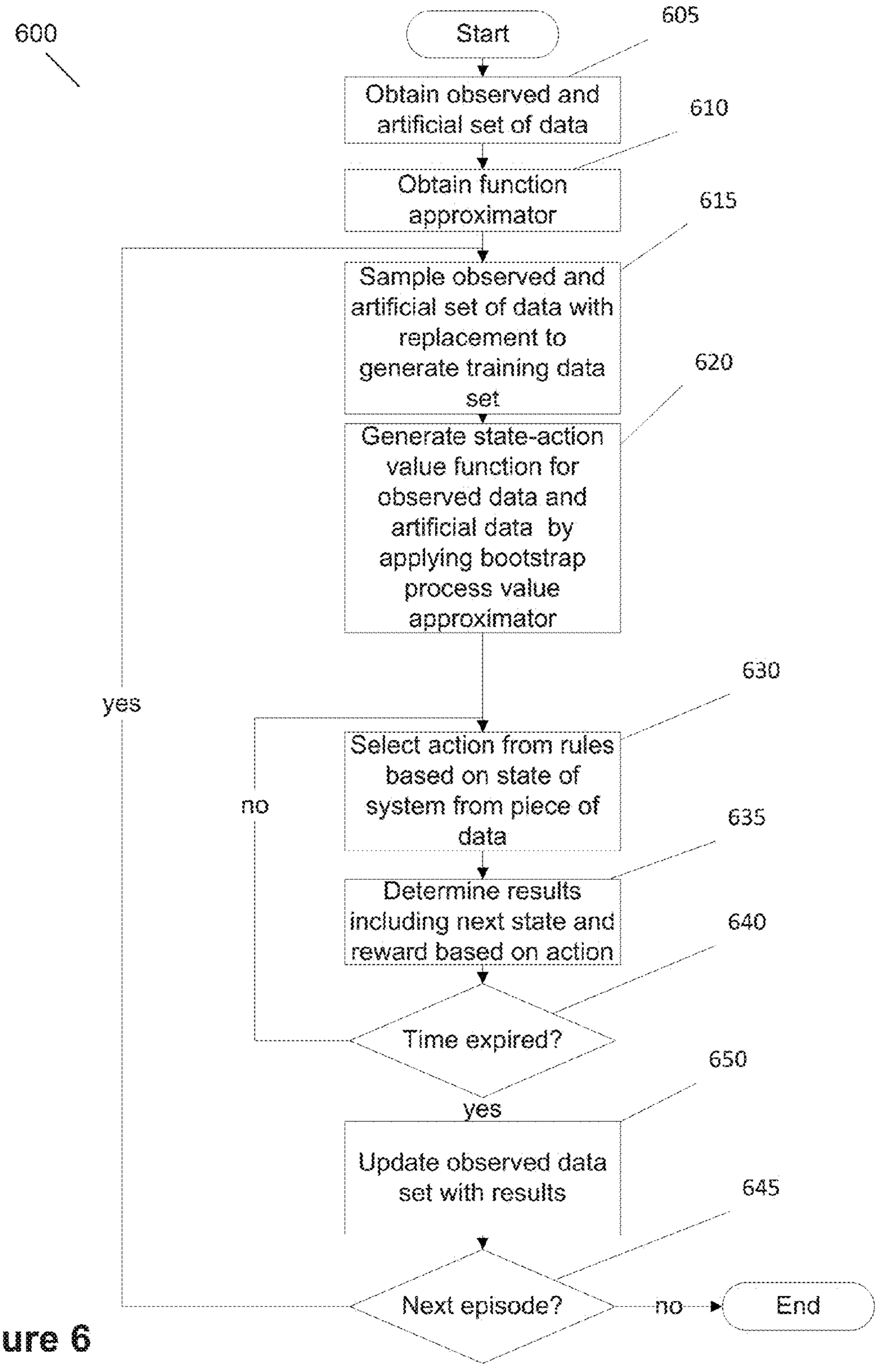

600
Start
605
Obtain observed and artificial set of data
610
Obtain function approximator
615
Sample observed and artificial set of data with replacement to generate training data set
620
Generate state-action value function for observed data and artificial data by applying bootstrap process value approximator
630
Select action from rules based on state of system from piece of data
635
Determine results including next state and reward based on action
640
Time expired?
yes
no
650
Update observed data set with results
645
Next episode?
yes
no
End

Figure 6

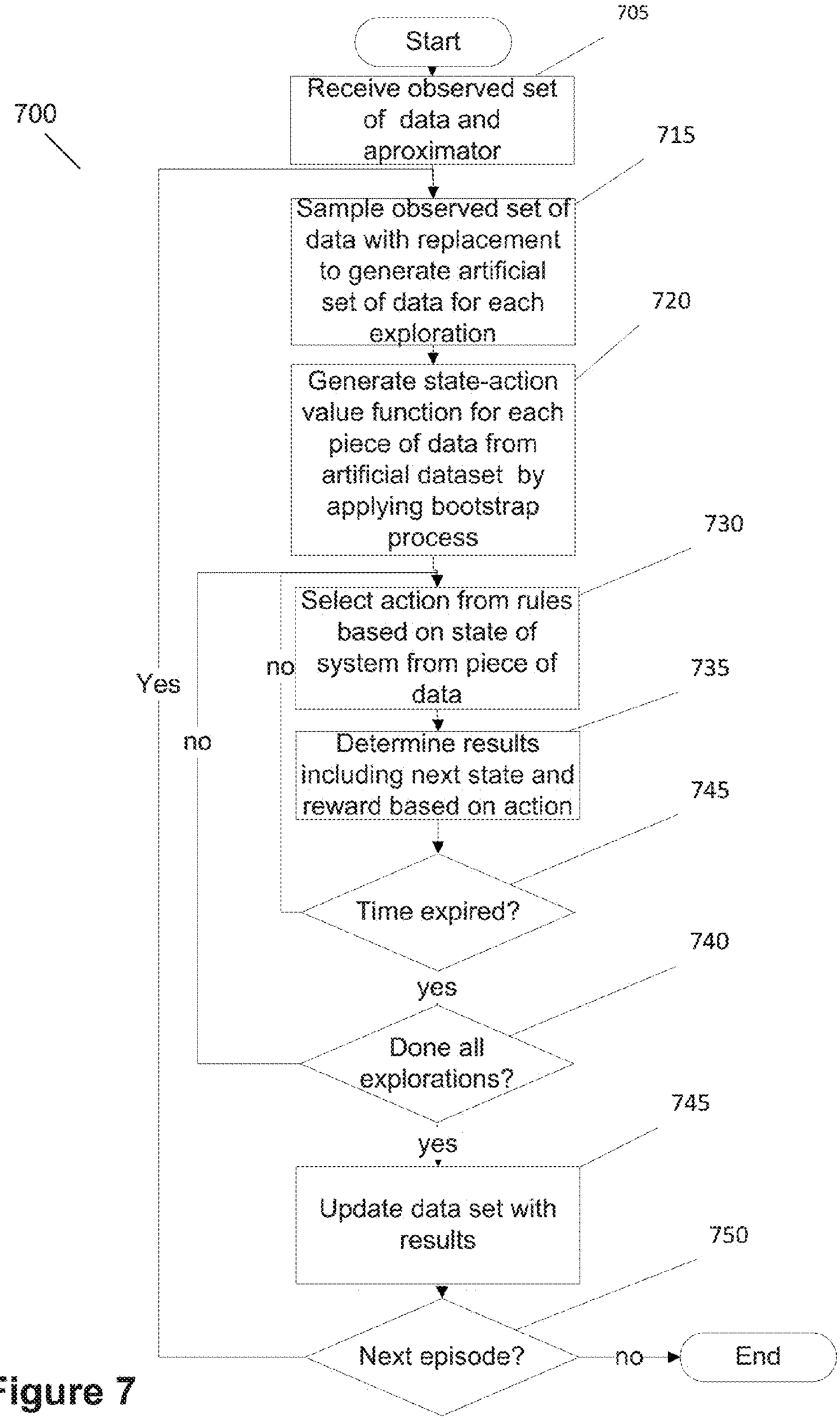

700
705
Start
Receive observed set of data and aproximator
715
Sample observed set of data with replacement to generate artificial set of data for each exploration
720
Generate state-action value function for each piece of data from artificial dataset by applying bootstrap process
730
Select action from rules based on state of system from piece of data
no
735
Determine results including next state and reward based on action
745
Time expired?
740
yes
Done all explorations?
Yes
no
745
yes
Update data set with results
750
Next episode?
no
End

Figure 7

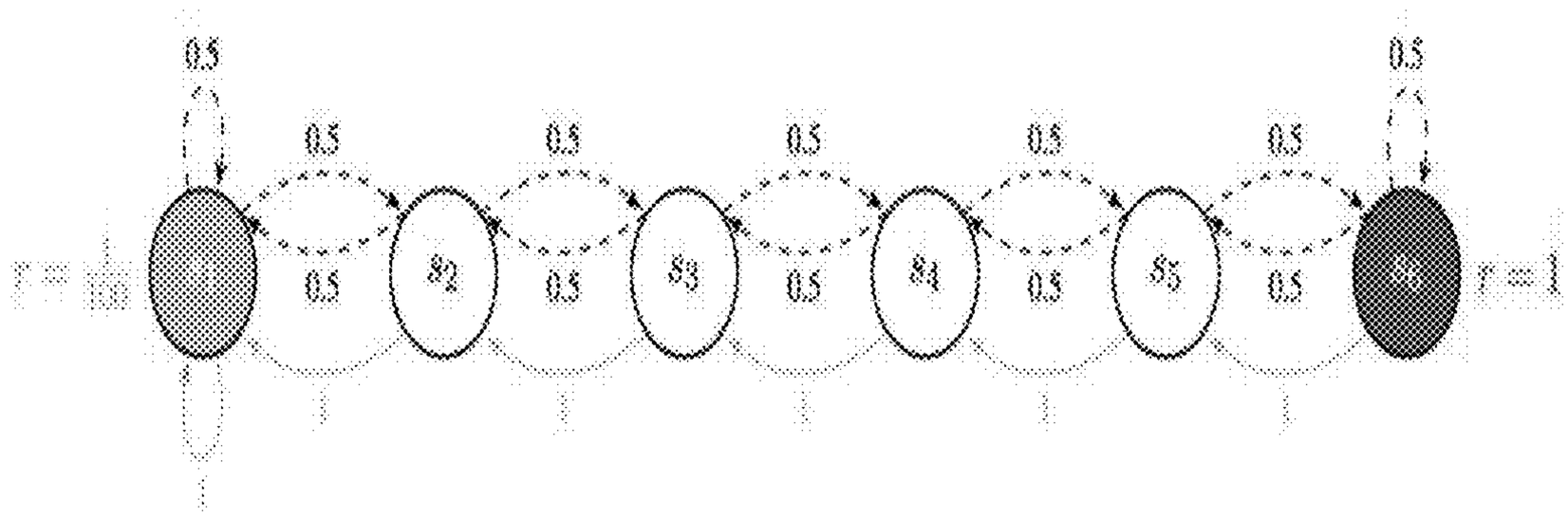


Figure 8

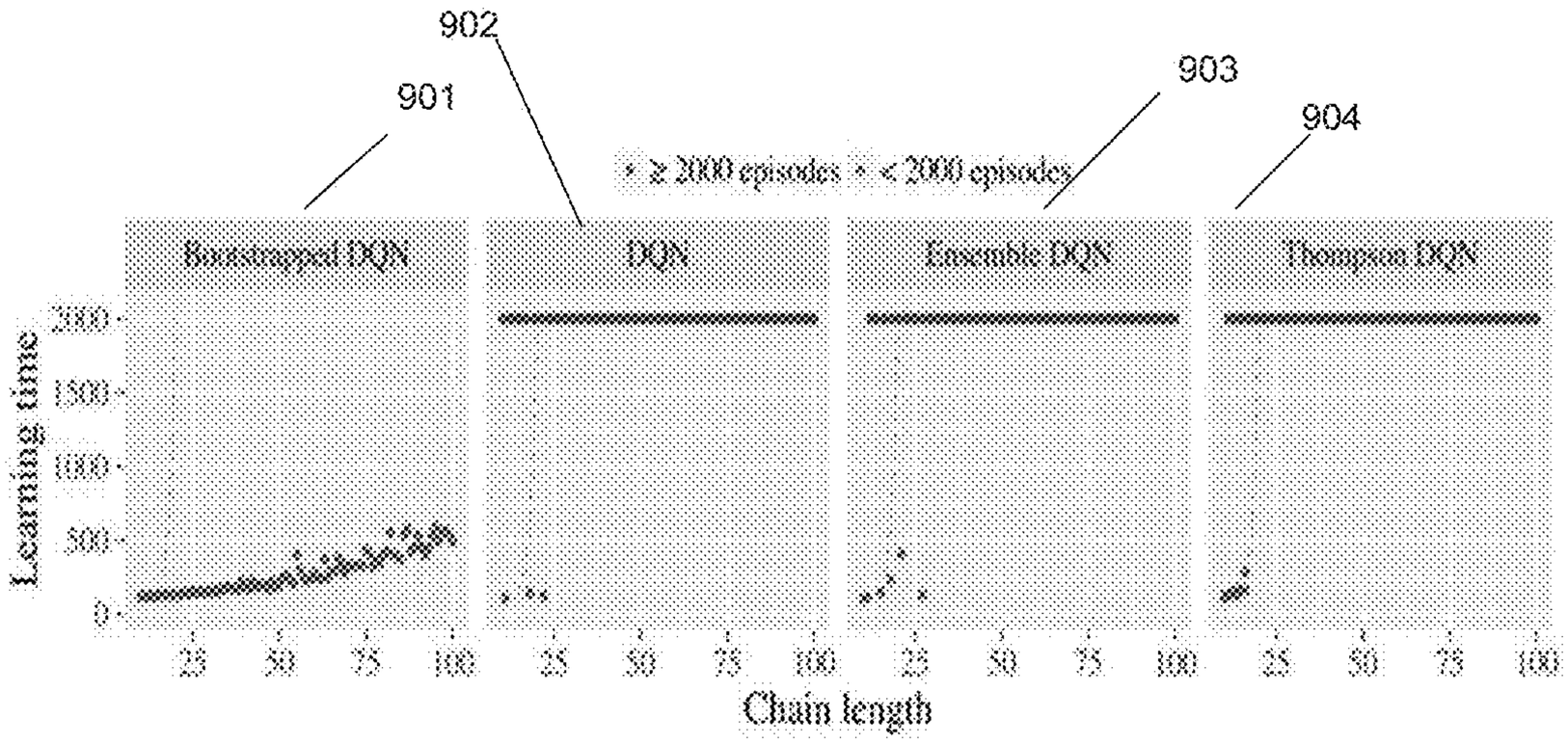


Figure 9

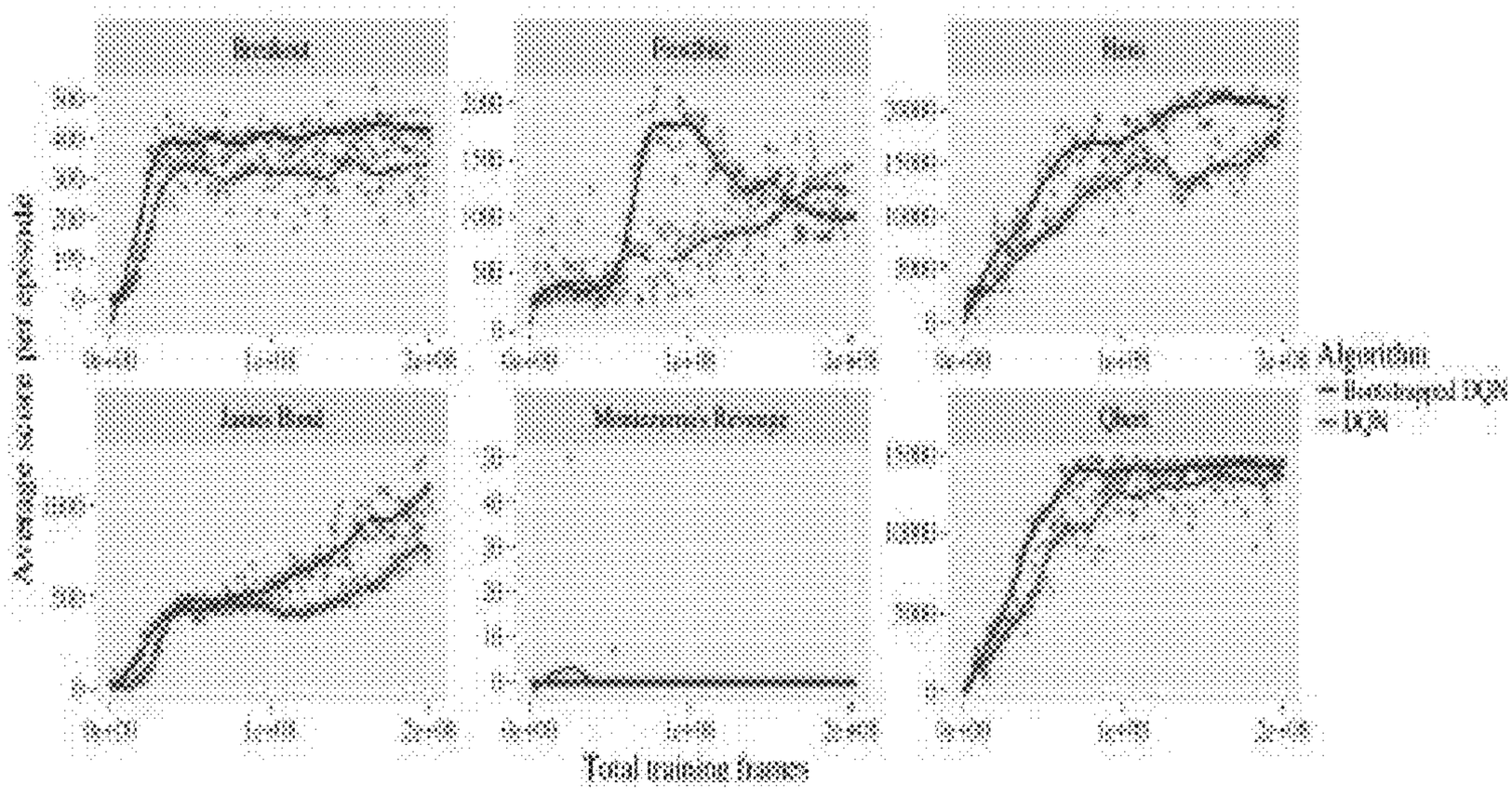

Breakout
Freeway
Ms. Pacman
Space Invaders
Seaquest
Algorithm
Bootstrapped DQN
DQN
Average score (last episode)
Total training frames

Figure 10

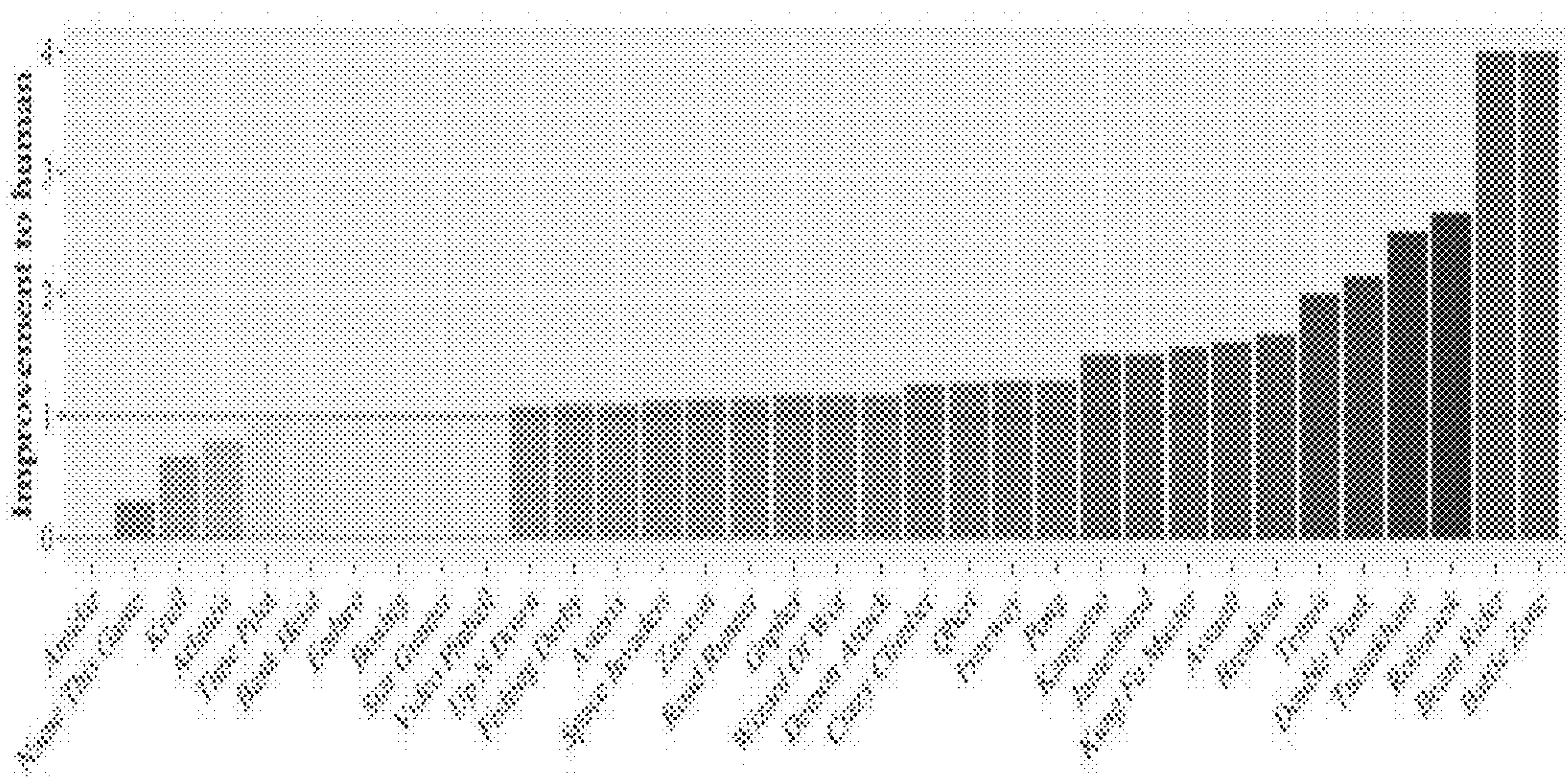

Improvement to human
Amidar
Name This Game
Krull
Atlantis
Time Pilot
Bank Heist
Enduro
Bowling
Star Gunner
Video Pinball
Up N Down
Fishing Derby
Asterix
Space Invaders
Zaxxon
Road Runner
Gopher
Wizard Of Wor
Demon Attack
Crazy Climber
Q*bert
Freeway
Pong
Kangaroo
Jamesbond
Kung Fu Master
Assault
Breakout
Tennis
Double Dunk
Tutankham
Robotank
Beam Rider
Battle Zone

Figure 11

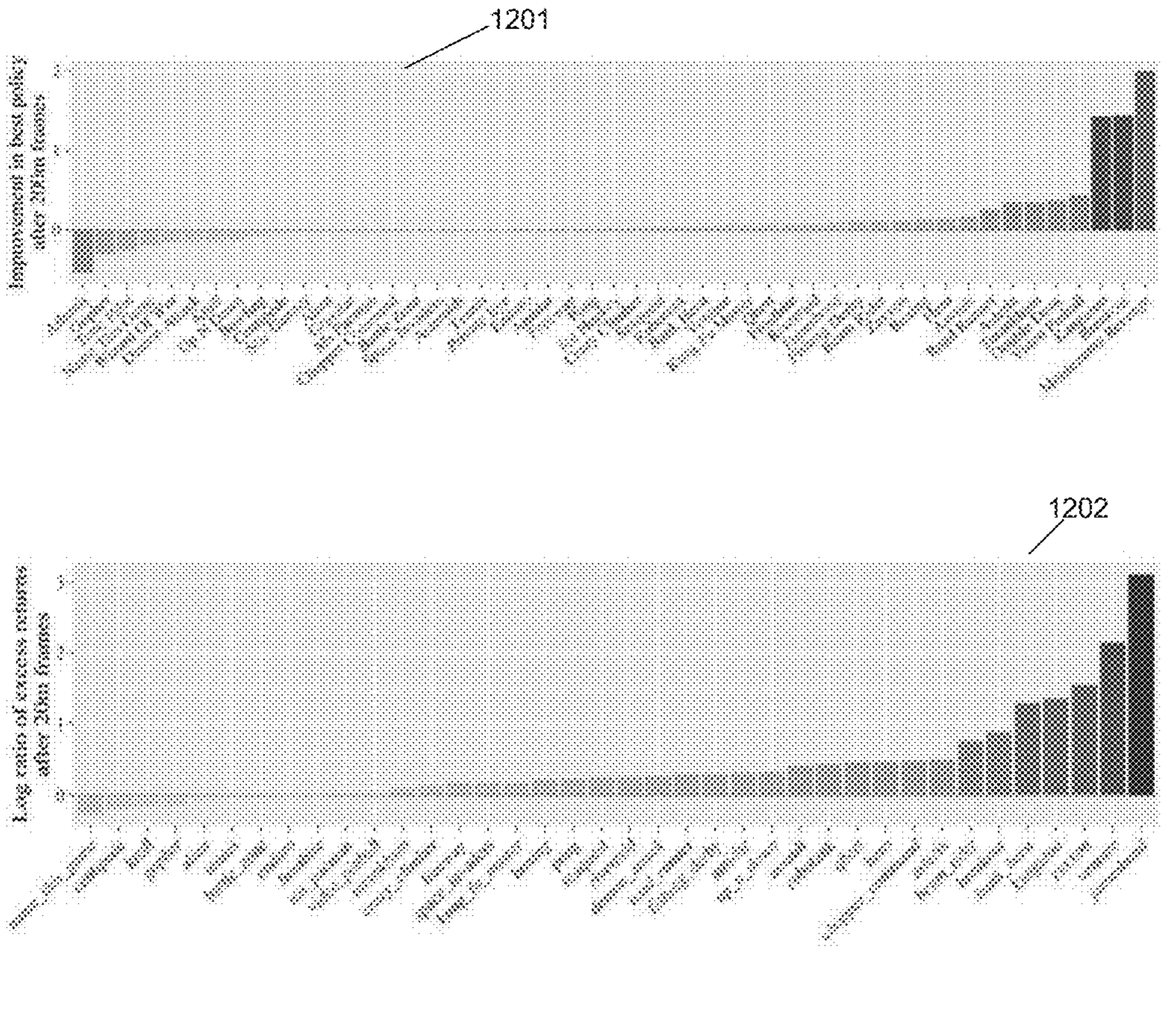


Figure 12

	Random	Human	Bootstrapped DQN	DDQN	Nature
Alien	227.8	7127.7	2436.6	**4007.7**	3069
Amidar	5.8	1719.5	1272.5	**2138.3**	739.5
Assault	222.4	742.0	**8047.1**	6997.9	3359
Asterix	210.0	8503.3	**19713.2**	17306.4	6012
Asteroids	719.1	47388.7	1032.0	**1981.4**	1629
Atlantis	12850.0	29028.1	**994500.0**	767850.0	85641
Bank Heist	14.2	753.1	**1208.0**	1106.0	429.7
Battle Zone	2360.0	37187.5	**38666.7**	34620.7	26300
Beam Rider	363.9	16926.5	**23429.8**	16650.7	6846
Bowling	23.1	160.7	68.2	**77.3**	42.4
Boxing	0.1	12.1	**93.2**	90.2	71.8
Breakout	1.7	30.5	**855.0**	437.0	401.2
Centipede	2090.9	12017.0	4553.5	4855.4	**8309**
Chopper Command	811.0	7387.8	4160.0	5049.0	**6687**
Crazy Climber	10780.5	35829.4	**137925.9**	137344.4	114103
Demon Attack	152.1	1971.0	82810.0	**98450.0**	9711
Double Dunk	-18.6	-16.4	**3.0**	-1.8	-18.1
Enduro	0.0	860.5	**1591.0**	1486.7	301.8
Fishing Derby	-91.7	-38.7	**26.0**	19.8	-0.8
Freeway	0.0	29.6	**33.9**	33.4	30.3
Frostbite	65.2	4334.7	2181.4	**2766.8**	328.3
Gopher	257.6	2412.5	**17438.4**	13815.9	8520
Gravitar	173.0	3351.4	286.1	**708.6**	306.7
Hero	1027.0	30826.4	**21021.3**	20674.2	19950
Ice Hockey	-11.2	0.9	**-1.3**	-1.7	-1.6
Jamesbond	29.0	302.8	**1663.5**	1128.2	576.7
Kangaroo	52.0	3035.0	**14862.5**	14717.6	6740
Krull	1598.0	2665.5	8627.9	**9690.9**	3805
Kung Fu Master	258.5	22736.3	**36733.3**	36365.7	23270
Montezuma Revenge	0.0	4753.3	**100.0**	0.0	0
Ms Pacman	307.3	6951.6	2983.3	**3424.6**	2311
Name This Game	2292.3	8049.0	11501.1	**11744.4**	7257
Pong	-20.7	14.6	**20.9**	**20.9**	18.9
Private Eye	24.9	69571.3	**1812.5**	158.4	1788
Qbert	163.9	13455.0	15092.7	**15209.7**	10596
Riverraid	1338.5	17118.0	12845.0	**14555.1**	8316
Road Runner	11.5	7845.0	**51500.0**	49518.4	18257
Robotank	2.2	11.9	66.8	**70.6**	51.6
Seaquest	68.4	42054.7	8083.1	**19183.9**	5286
Space Invaders	148.0	1668.7	2883.0	**4715.8**	1976
Star Gunner	664.0	10250.0	55725.0	**66091.2**	57997
Tennis	-23.8	-8.3	0.0	**11.8**	-2.5
Time Pilot	3568.0	5229.2	8079.4	**10075.8**	5947
Tutankham	11.4	167.6	214.8	**268.0**	186.7
Up N Down	533.4	11693.2	**26231.0**	19743.5	8456
Venture	0.0	1187.5	212.5	239.7	**380**
Video Pinball	0.0	17667.9	**811610.0**	685911.0	42684
Wizard Of Wor	563.5	4756.5	6884.7	**7855.7**	3393
Zaxxon	32.5	9173.3	11491.7	**12947.6**	4977

Table 1: Maximal evaluation Scores achieved by agents

Figure 13

SYSTEMS AND METHODS FOR PROVIDING REINFORCEMENT LEARNING IN A DEEP LEARNING SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The current application is a Continuation-In-Part Application of U.S. patent application Ser. No. 15/201,284 filed Jul. 1, 2016 that in turn claims priority to U.S. Provisional Application No. 62/187,681, filed Jul. 1, 2015, the disclosures of which are incorporated herein by reference as if set forth herewith.

FIELD OF THE INVENTION

[0002] This invention relates to deep learning networks including, but not limited to, artificial neural networks. More particularly, this invention relates to systems and methods for training deep learning networks from a set of training data using reinforcement learning.

BACKGROUND OF THE INVENTION

[0003] Deep learning networks including, but not limited to, artificial neural networks are machine learning systems that receive data, extract statistics and classify results. These systems use a training set of data to generate a model in order to make data driven decisions to provide a desired output.

[0004] Deep learning networks are often used to solve problems in an unknown environment where the training dataset is used to ascertain the extent of the environment. One manner of training a deep learning network is referred to as reinforcement learning in which the system takes a sequence of actions in order to maximize cumulative rewards. In reinforcement learning, the system begins with an imperfect knowledge of the environment and learns through experience. As such, there is a fundamental trade-off between exploration and exploitation in that the system may improve its future rewards by exploring poorly understood states and actions and sacrificing immediate rewards.

[0005] Many approaches to reinforcement learning have been put forth. Most of the proposed approaches are designed based upon Markov Decision Processes (MDPs) with small finite state spaces. Some other approaches require solving computationally intractable planning tasks. These approaches are not practical in complex environments that require the system to generalize in order to operate properly. Thus, these reinforcement learning approaches in large-scale application have relied upon either statistically inefficient exploration strategies or include no exploration at all.

[0006] Some other common exploration approaches are dithering strategies. An example of a dithering strategy is a ϵ-greedy. In common dithering strategies, the approximated value of an action is a single value and the system picks the action with the highest value. In some strategies, the system may also choose some actions at random. Another common exploration strategy is inspired by Thompson sampling. In a Thompson sampling strategy there is some notion of uncertainty. However, a distribution of the maintained over the possible values from the dataset and the system is explored by randomly selecting a policy according to the probability that the selected policy is the optimal policy.

SUMMARY

[0007] The above and other problems are solved and an advance in the art is made by systems and methods for providing reinforcement learning in deep learning networks in accordance with some embodiments of the invention. Reinforcement learning with deep exploration is provided in the following manner in accordance with some embodiments of the invention. A deep neural network is maintained. A reinforcement learning process is applied to the deep neural network.

[0008] The reinforcement learning process is performed in the following manner in accordance with some embodiments. A set of observed data and a set artificial data is received. For each of a number of episodes, the process samples a set of data that is a union of the set of observed data and the set of artificial data to generate set of training data. A state-action value function is then determined for the set of training data using a bootstrap process and an approximator. The approximator estimates a state-action function for a dataset. For each time step in each of the one or more episodes, the process determines a state of the system for a current time step from the set of training data. An action based on the determined state of the system and a policy mapping actions to the state of the system is selected by the process and results for the action including a reward and a transition state that result from the selected action are determined. Result data from the current time step that includes the state, the action, the transition state are stored. The set of the observed data is then updated with the result data from at least one time step of an episode at the conclusion of an episode.

[0009] In accordance with some embodiments, the reinforcement learning process generates the set of artificial data from the set of observed data. In accordance with many of these embodiments the artificial data is generated by sampling the set of observed data with replacement to generate the set of artificial data. In accordance with a number of other embodiments, the artificial data is generated by sampling state-action pairs from a diffusely mixed generative model and assigning each the sampled state-action pairs stochastically optimistic rewards and random state transitions.

[0010] In accordance with some embodiments, the reinforcement learning process maintains a training mask that indicates the result data from each of the time period in each episode to be used in training and updates the set of observed data by adding the result data from each time period of an episode indicated in the training mask.

[0011] In accordance with some embodiments, the approximator received as an input. In accordance with many embodiments, the approximator is read from memory. In accordance with a number of embodiments, the approximator is a neural network trained to fit a state-action value function to the data set via a least squared iteration.

[0012] In accordance with some embodiments, one or more reinforcement learning processes are applied to the deep neural network. In accordance with many of these embodiments, each of the reinforcement learning processes independently maintains a set of observed data. In accordance with some other embodiments, the reinforcement learning processes cooperatively maintain the set of observed data. In accordance with some of these embodiments, a bootstrap mask that indicates each element in the

set of observed data that is available to each of the reinforcement learning processes is maintained.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. **1** illustrates various devices in a network that perform processes that systems and methods for providing reinforcement learning in a deep learning network in accordance with various embodiments of the invention.

[0014] FIG. **2** illustrates a processing system in a device that performs processes that provide systems and methods for providing reinforcement learning in a deep learning network in accordance with various embodiments of the invention.

[0015] FIG. **3** illustrates a deep neural network that uses processes providing reinforcement learning in deep learning networks in accordance with some embodiments of the invention.

[0016] FIG. **4** illustrates a state diagram of a deterministic chain representing an environment.

[0017] FIG. **5** illustrates planning and look ahead trees for exploring the deterministic chain shown in FIG. **4** in accordance with various approaches.

[0018] FIG. **6** illustrates a process for providing reinforcement learning in a deep learning network in accordance with an embodiment of the invention.

[0019] FIG. **7** illustrates an incremental process for providing reinforcement learning in a deep learning network in accordance with an embodiment of the invention.

[0020] FIG. **8** illustrates a deterministic chain of states in an environment of a problem.

[0021] FIG. **9** illustrates the results of application of various reinforcement learning approaches.

[0022] FIG. **10** illustrates results of a deep learning network using processes that provide reinforcement learning in accordance with an embodiment of the invention and results from a DQN network.

[0023] FIG. **11** illustrates a graph showing results of a deep learning network using processes that provide reinforcement learning in accordance with an embodiment of the invention learning various Atari games compared to a human player playing the games.

[0024] FIG. **12** illustrates graphs showing improvements to policies and rewards of various Atari games by deep learning network using systems and methods for providing reinforcement learning in accordance with an embodiment of the invention.

[0025] FIG. **13** illustrates a table of results for various deep learning networks including a deep learning network that uses process providing reinforcement learning processes in accordance with an embodiment of the invention.

DETAILED DISCUSSION

[0026] Turning now to the drawings, systems and methods for providing reinforcement learning to a deep learning network in accordance with various embodiment of the invention are disclosed. For purposes of this discussion, deep learning networks are machine learning systems that use a dataset of observed data to learn how to solve a problem in a system where all of the states of the system, actions based upon states, and/or the resulting transitions are not fully known. Examples of deep learning networks include, but are not limited to, deep neural networks.

[0027] System and methods in accordance with some embodiments of this invention that provide reinforcement learning do so by providing an exploration process for a deep learning network to solve a problem in an environment. In reinforcement learning, actions taken by a system may impose delayed consequences. Thus, the design of exploration strategies is more difficult than systems that are action-response systems where there are no delayed consequences such as multi-armed bandit problems because the system must establish a context. An example of a system that has delayed consequences is a system that interacts with an environment over repeated episodes, l, of length τ. In each time step, $t=1, \ldots, \tau$, of an episode, the system observes a state of the environment, s_{lt}, and selects an action, a_{lt}, according to a policy π which maps the states to actions. A reward, r_{lt}, and a state transition to state, s_{lt+1}, are realized in response to the action. The goal of the system during exploration is to maximize the long-term sum of the expected rewards even though the system is unsure of the dynamics of the environment and the reward structure.

Deep Learning

[0028] To understand a system that may have delayed consequences, a deep learning network needs to explore as many states of the system to understand the state-action policy and the rewards associated with actions. Uncertainty estimates allow a system to direct an exploration process at potentially informative states and actions. In multi-arm bandit problems, directed exploration of the system rather than a dithering exploration generally categorizes efficient algorithms. However, directed exploration is not enough to guarantee efficiency in more complex systems with delayed consequences. Instead, the exploration must also be deep. Deep exploration means exploration that is directed over multiple time steps. Deep exploration can also be called "planning to learn" or "far-sighted" exploration. Unlike exploration of multi-arm bandit problems, deep exploration require planning over several time steps instead of the balancing of actions which are immediately rewarding or immediately informative in a directed exploration. For deep learning exploitation, an efficient agent should consider the future rewards over several time steps and not simply the myopic rewards. In exactly the same way, efficient exploration may require taking actions which are neither immediately rewarding, nor immediately informative.

[0029] To illustrate this distinction, consider a simple deterministic chain $\{s_{-3}, \ldots, s_{+3}\}$ with three step horizon starting from state s_0 is shown in FIG. **4**. The Markov Decision Processes (MDP) of the chain is known to a system a priori, with deterministic actions "left" and "right". All states have zero reward, except for the leftmost state s_{-3} which has known reward of $\epsilon>0$ and the rightmost state s_3 which is unknown. In order to reach either a rewarding state or an informative state within three steps from s_0 a system needs to plan a consistent strategy over several time steps.

[0030] Planning and look-ahead trees for several algorithmic exploration approaches to the MDP of this deterministic chain are shown in FIG. **5**. In FIG. **5**, tree **501** represents the possible decisions of a bandit algorithm, tree **502** represents the possible decisions of a dithering algorithm, tree **503** represents the possible decisions of a shallow exploration algorithm, and tree **504** represents the possible decisions of a deep exploration algorithm. In all of the trees **501-504**, Actions, including action "left", and action "right" are solid

lines; rewarding states are at the left and right most bottom nodes; and dashed lines indicate that the agent can plan ahead for either rewards or information. As can be seen from trees **501-504** only a system that employs a deep exploration strategy such as reinforcement learning Unlike bandit algorithms, an reinforcement learning agent can plan to exploit future rewards. The strategies that use direct exploration cannot plan ahead.

Reinforcement Learning

[0031] Reinforcement learning is a deep learning approach that differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). A common approach to reinforcement learning involves learning a state-action value function, Q, which for time, t, state, s, and action, a, provides an estimate, $Q_t(s, a)$, of expected rewards over the remainder of the episode: $r_{l,\ t}+r_{lt+1}+\ldots+r_{l\tau}$. Given a state-action value function, Q, the system selects an action that maximizes $Q_t(s, a)$ when at state s and time t. Most reinforcement learning systems provide an exploration strategy that balances exploration with exploitation. However, the vast majority of these processes operate in a "tabula rasa" setting which does not allow for generalization between state-action pairs which is needed in systems having a large number of states and actions.

[0032] A bootstrap principle is commonly used to approximate a population distribution be a sample distribution. A common bootstrap takes as input a data set D and an estimator, γ. The bootstrap generates a sample data set from the bootstrapped distribution that has a cardinality equal to D and is sampled uniformly with replacement from data set D. The bootstrap sample estimate is then taken to by γ (D). A network that is an efficient and scalable system for generating bootstrap samples from a large and deep neural network includes a shared architecture with K bootstrapped head (or exploration processes) branching off independently. Each head is trained only on a separate unique sub-sample of data that represents a single bootstrap sample γ (D'). The shared network learns via a joint feature representation across all of the data, which can provide significant computational advantages at the cost of lower diversity between heads. This type of bootstrap can be trained efficiently in a single forward/backward pass and can be thought of as data-dependent dropout, where the dropout mask for each head is fixed for each data point.

[0033] For a policy π, the value of an action a in state s may be expressed as $Q^{\pi}(s, a):=\mathbb{E}_{s,a,\pi}[\Sigma_{t=1}^{\infty}\gamma^t r_t]$, where $\gamma\epsilon(0,1)$ is a discount factor that balances immediate versus future rewards r_t. This expectation indicates that the initial state is s, the action is a, and thereafter actions are selected by the policy π. The optimal value is $Q^*(s, a):=\max_{\pi}Q^{\pi}(s, a)$. To scale to large problems, a parameterized estimate of the Q-value function Q(s, a; θ) is used rather than a tabular encoding. To estimate the parameterized value, a separate neural network is used as an approximator function to estimate the parameterized value.

[0034] In a deep Q learning network, a Q-learning update from state s_t, action a_t, reward r_t and new state s_{t+1} is given by

$$\theta_{t+1}\leftarrow\theta_t+\alpha(y_t^{Q}-Q(s_t,a_t;\theta_t))\nabla_{\theta}Q(s_t,a_t;\theta_t) \quad (1)$$

[0035] Where α is the scalar learning rate and y_t^{Q} is the target value $r_t+\gamma\ \max_a Q(s_{t+1}, a; \theta^-)$. θ^- are target network parameters fixed $\theta^-=\theta_t$.

[0036] Several important modifications to the updating process in Q-learning improve stability for a deep learning network using reinforcement learning provided in accordance with some embodiments of the invention. First, the system learns from sampled transitions from an experience buffer, rather than learning fully online. Second, the system uses a target network with parameters θ^- that are copied from the learning network $\theta^-\leftarrow\theta_t$ only every τ time steps and then kept fixed in between updates. A Double DQN system modifies the target y_t^{Q} and may help further as shown in Equation (2):

$$y_t^{Q}\leftarrow r_t+\gamma\ \max_a Q(s_{t+1},\ \arg\max_a Q(s_{t+1},a;\theta_t);\theta^-). \quad (2)$$

[0037] A modifies the learning process to approximate a distribution over Q-values via the bootstrap. At the start of each episode, a deep learning network that uses reinforcement learning as provided in accordance with some embodiments of the invention samples a single Q-value function from an approximate posterior maintained by the system. An exploration process then follows the policy which is optimal for that sample for the duration of the episode. This is a natural adaptation of the Thompson sampling heuristic to reinforcement learning that allows for temporally extended (or deep) exploration.

[0038] An exploration process for a deep learning network that uses reinforcement learning as provided in accordance with some embodiments of the invention may efficiently implemented by building up $K\epsilon\mathbb{N}$ bootstrapped estimates of the Q-value function in parallel. Each one of these value function heads $Q_k(s, a; \theta)$ is trained against a separate target network $Q_k(s, a; \theta^-)$ such that each $Q_1, \ldots, Q_K$ provides a temporally extended (and consistent) estimate of the value uncertainty via Thompson sampling distribution estimates. In order to keep track of which data belongs to which bootstrap head, flags $\omega_1, \ldots, \omega_K\epsilon\{0,1\}$ that indicate which heads are privy to which data may be used are maintained. A bootstrap sample is made by selecting $k\epsilon\{1, \ldots, K\}$ uniformly at random and following Q_k for the duration of that episode.

[0039] The observation that temporally extended exploration is necessary for efficient reinforcement learning is not new. For any prior distribution of MDPs, the optimal exploration strategy is available through dynamic programming in the Bayesian belief state space. However, the exact solution is intractable even for very simple systems. Many successful reinforcement learning applications focus on generalization and planning but address exploration only via an inefficient exploration strategy or not at all. However, such exploration strategies can be highly inefficient.

[0040] Many exploration strategies are guided by the principle of "optimism in the face of uncertainty" (OFU). These algorithms add an exploration bonus to values of state-action pairs that may lead to useful learning and select actions to maximize these adjusted values. This approach was first proposed for finite-armed bandits, but the principle has been extended successfully across various multi-armed bandits with generalization and/or tabular reinforcement learning. Except for particular deterministic contexts, OFU methods that lead to efficient reinforcement learning in complex domains have been computationally intractable. A

particular OFU system aims to add an effective bonus through a variation of a Deep Q-learning Network (DQN). The resulting system relies on a large number of hand-tuned parameters and is only suitable for application to deterministic problems.

[0041] Perhaps the oldest heuristic for balancing exploration with exploitation is given by Thompson sampling. Thompson sampling is often referred to a bandit algorithm and takes a single sample from the posterior at every time step and chooses the action which is optimal for that time step. To apply the Thompson sampling principle to reinforcement learning, a system samples a value function from its posterior. Naïve applications of Thompson sampling to reinforcement learning resample every time step can be extremely inefficient. Such a system agent would have to commit to a sample for several time steps in order to achieve deep exploration. One proposed system, PSRL does commit to a sample for several steps and provides state of the art guarantees. However, PSRL still requires solving a single known MDP, which will usually be intractable for large systems.

[0042] A deep learning network that use reinforcement learning provided in accordance with some embodiments of the invention approximates commits to a sample for several steps exploration via randomized value functions sampled from an approximate posterior. A deep learning network that use reinforcement learning provided in accordance with some embodiments of the invention recovers state of the art guarantees in the setting with tabular basis functions, but the performance of these systems is crucially dependent upon a suitable linear representation of the value function. A deep learning network that use reinforcement learning provided in accordance with some embodiments of the invention extends these ideas to produce a system that can simultaneously perform generalization and exploration with a flexible nonlinear value function representation. Our method is simple, general and compatible with almost all advances in deep exploration via reinforcement learning at low computational cost and with few tuning parameters.

[0043] A reinforcement learning system in accordance with embodiments of this invention overcomes these problems by providing an exploration strategy that combines efficient generalization and exploration via leveraging a bootstrap process and artificial data. In accordance with some embodiments of the invention, the system receives a set of training data that includes observed data and artificial data. In accordance with some embodiments, the artificial data is generated by sample state-action pairs from a diffusely mixed generative model and assign each state-action pair stochastically optimistic rewards and random state transitions. In accordance with some other embodiments, the artificial data is generated by sampling a set of observed data with replacement to obtain a set of data having a number of elements that is approximately equal to or greater than the number of elements as the set of observed data.

[0044] The observed and artificial data are sampled to obtain a training sample set of data. In accordance with some of these embodiments, the training sample dataset includes M samples of data. In accordance with a number of these embodiments, M is equal to or greater than a number of episodes to observe during an exploration process. In accordance with some embodiments, the sampling is performed in accordance with a known and/or a provided distribution. A bootstrap process is applied to the union of the observed data and the artificial data to obtain a new distribution for the sample of data. An approximator function is applied to the new distribution to generate a randomized state-value function.

[0045] For each episode, the process observes a state of the system, s_{lt}, for a particular time period from the training sample dataset and selects an action to perform based upon policy π. The results including a reward, r_{lt} realized and a resulting transition state, s_{lt+1} resulting from the action are observed. The state, action, reward and transition state are stored as result data for the episode. This is repeated for each time step in the episode. After the episode is completed, the observed dataset is updated with the result data stored during the episode. In accordance with some embodiments, a training mask may be maintained that includes flags indicating the result data from particular time steps is to be used for training. The training mask is then read to determine the result data to add to the observed data.

[0046] In accordance with many embodiments, multiple exploration processes may be performed concurrently. In accordance with a number of these embodiments, the result data observed from each exploration process is shared with other exploration processes. In accordance with a number of embodiments, the observed dataset for each process is updated independently. In accordance with some of these embodiments, a boot strap mask is maintained that indicates which elements of the observed dataset are available to each process.

[0047] In accordance with some embodiments, a replay buffer may be maintained and playback to update parameters of the value function network Q.

[0048] Systems and method for providing reinforcement learning in a deep learning network in accordance with some embodiments of the invention are set forth below with reference to the Figures.

Systems that Provide Deep Learning Networks

[0049] A system that provides a deep learning system that uses systems and methods that provide reinforcement learning in accordance with some embodiments of the invention is shown in FIG. **1**. Network **100** includes a communications network **160**. The communications network **160** is a network such as the Internet that allows devices connected to the network **160** to communicate with other connected devices. Server systems **110**, **140**, and **170** are connected to the network **160**. Each of the server system **110**, **140**, and **170** is a group of one or more servers communicatively connected to one another via internal networks that execute processes that provide cloud services to users over the network **160**. For purposes of this discussion, cloud services are one or more applications that are executed by one or more server systems to provide data and/or executable applications to devices over a network. The server systems **110**, **140**, and **170** are shown each having three servers in the internal network. However, the server systems **110**, **140** and **170** may include any number of servers and any additional number of server systems may be connected to the network **160** to provide cloud services. In accordance with various embodiments of this invention, a deep learning network that uses systems and methods that provide reinforcement learning in accordance with an embodiment of the invention may be provided by process being executed on a single server system and/or a group of server systems communicating over network **160**.

[0050] Users may use personal devices **180** and **120** that connect to the network **160** to perform processes for providing and/or interaction with a deep learning network that uses systems and methods that provide reinforcement learning in accordance with various embodiments of the invention. In the shown embodiment, the personal devices **180** are shown as desktop computers that are connected via a conventional "wired" connection to the network **160**. However, the personal device **180** may be a desktop computer, a laptop computer, a smart television, an entertainment gaming console, or any other device that connects to the network **160** via a "wired" connection. The mobile device **120** connects to network **160** using a wireless connection. A wireless connection is a connection that uses Radio Frequency (RF) signals, Infrared signals, or any other form of wireless signaling to connect to the network **160**. In FIG. **1**, the mobile device **120** is a mobile telephone. However, mobile device **120** may be a mobile phone, Personal Digital Assistant (PDA), a tablet, a smartphone, or any other type of device that connects to network **160** via wireless connection without departing from this invention.

Example of a Processing System

[0051] An example of a processing system in a device that executes instructions to perform processes that provide interact with other devices connected to the network as shown in FIG. **1** to provide a deep learning network that uses systems and methods that provide reinforcement learning in accordance with various embodiments of the invention in accordance with various embodiments of this invention is shown in FIG. **2**. One skilled in the art will recognize that a particular processing system may include other components that are omitted for brevity without departing from this invention. The processing device **200** includes a processor **205**, a non-volatile memory **210**, and a volatile memory **215**. The processor **205** is a processor, microprocessor, controller, or a combination of processors, microprocessor, and/or controllers that performs instructions stored in the volatile **215** or the non-volatile memory **210** to manipulate data stored in the memory. The non-volatile memory **210** can store the processor instructions utilized to configure the processing system **200** to perform processes including processes in accordance with embodiments of the invention and/or data for the processes being utilized. In other embodiments, the processing system software and/or firmware can be stored in any of a variety of non-transient computer readable media appropriate to a specific application. A network interface is a device that allows processing system **200** to transmit and receive data over a network based upon the instructions performed by processor **205**. Although a processing system **200** is illustrated in FIG. **2**, any of a variety of processing system in the various devices can configured to provide the methods and systems in accordance with embodiments of the invention can be utilized.

System that Provides Training by Multiple Concurrently Running Exploration Processes

[0052] In accordance with some embodiments of the invention, a deep learning network may currently run multiple exploration processes to achieve greater exploration of an environment. A conceptual diagram of a deep learning network that has multiple concurrently running exploration processes in accordance with an embodiment of this invention is shown in FIG. **3**. Deep learning network **300** operates on a frame **305**. K number of heads or exploration processes interact with network **300** to explore the environment.

[0053] The bootstrap principle is used to approximate a population distribution be a sample distribution. A common bootstrap takes as input a data set D and an estimator, γ. The bootstrap generates a sample data set from the bootstrapped distribution that has a cardinality equal to D and is sampled uniformly with replacement from data set D. The bootstrap sample estimate is then taken to by $\gamma(D)$. System **300** is an efficient and scalable system for generating bootstrap samples from a large and deep neural network. The network **300** includes a shared architecture with K bootstrapped heads (or exploration processes) branching off independently. Each head is trained only on a separate unique sub-sample of data that represents a single bootstrap sample γ (D'). The shared network learns via a joint feature representation across all of the data, which can provide significant computational advantages at the cost of lower diversity between heads. This type of bootstrap can be trained efficiently in a single forward/backward pass and can be thought of as data-dependent dropout, where the dropout mask for each head is fixed for each data point.

[0054] For a policy π, the value of an action a in state s may be expressed as $Q^{\pi}(s, a):=\mathbb{E}_{s,a,\pi}[\Sigma_{t=1}^{\infty}\gamma^{t}r_{t}]$, where $\gamma\in(0,1)$ is a discount factor that balances immediate versus future rewards r_t. This expectation indicates that the initial state is s, the action is a, and thereafter actions are selected by the policy π. The optimal value is $Q^{*}(s, a):=\max_{\pi}Q^{\pi}(s, a)$. To scale to large problems, a parameterized estimate of the Q-value function $Q(s, a; \theta)$ is used rather than a tabular encoding. In accordance with some embodiments, a neural network is used to estimate the parameterized value.

[0055] The Q-learning update from state s_t, action a_t, reward r_t and new state s_{t+1} is given by

$$\theta_{t+1}\leftarrow\theta_{t}+\alpha(y_{t}^{Q}-Q(s_{t},a_{t};\theta_{t}))\nabla_{\theta}Q(s_{t},a_{t};\theta_{t}) \quad (1)$$

[0056] Where α is the scalar learning rate and y_t^Q is the target value $r_t+\gamma \max_a Q(s_{t+1}, a; \theta^{-})$. θ^{-} are target network parameters fixed $\theta^{-}=\theta_t$.

[0057] Several important modifications to the updating process in Q-learning improve stability for a deep learning network using reinforcement learning provided in accordance with some embodiments of the invention. First the algorithm learns from sampled transitions from an experience buffer, rather than learning fully online, Second the algorithm uses a target network with parameters θ^{-}that are copied from the learning network $\theta^{-}\leftarrow\theta_t$ only every τ time steps and then kept fixed in between updates. Double DQN modifies the target y_t^Q and helps further:

$$y_{t}^{Q}\leftarrow r_{t}+\gamma \max_{a}Q(s_{t+1}, \arg\max_{a}Q(s_{t+1},a;\theta_{t});\theta^{-}). \quad (2)$$

[0058] A deep learning network that use reinforcement learning as provided in accordance with embodiments of this invention modifies DQN to approximate a distribution over Q-values via the bootstrap. At the start of each episode, A deep learning network that use reinforcement learning as provided in accordance with embodiments of this invention samples a single Q-value function from its approximate posterior. The system follows the policy which is optimal for that sample for the duration of the episode.

[0059] An exploration process is efficiently implemented by building up $K\in\mathbb{N}$ bootstrapped estimates of the Q-value function in parallel as in FIG. **3**. Importantly, each one of these value function heads Q_k (s, a; θ) is trained against its own target network $Q_k(s, a; \theta^{-})$. This means that each Q_1, .

. . . , Q_K provide a temporally extended (and consistent) estimate of the value uncertainty via TD Estimates. In order to keep track of which data belongs to which bootstrap head we store flags $\omega_1, \ldots, \omega_K \epsilon \{0,1\}$ indicating which heads are privy to which data. We approximate a bootstrap sample by selecting $k \epsilon \{1, \ldots, K\}$ uniformly at random and following Q_k for the duration of that episode.

Reinforcement Learning Exploration Process

[0060] In accordance with some embodiments of the invention, systems and methods provide reinforcement learning by providing a deep exploration process. The deep exploration process fits a state-action value function to a sample of data from set of data that includes artificial data and observed data. In accordance with some embodiments of the invention, the system receives a set of training data that includes observed data and artificial data. In accordance with some these embodiments, the artificial data is generated by sampling state-action pairs from a diffusely mixed generative model and assign each state-action pair stochastically optimistic rewards and random state transitions. In accordance with some other embodiments, the artificial set of data is generated by sampling the observed set of data with replacement. In accordance with a number of embodiments, the artificial data includes M elements where M is approximately greater than or equal to the number of elements in the observed dataset. The use of the combination of observed and historical data provides randomness in the samples to induce deep learning. An exploration process for providing reinforcement learning to a deep learning network in accordance with an embodiment of this invention is shown in FIG. **6**.

[0061] Process **600** performs exploration in M distinct episodes. In accordance with some embodiments, the number of episodes is received as an input and in accordance with some other embodiments the number of episodes may set or selected by the process based on the size of the deep learning network. A set of data including historical and artificial data is obtained (**605**). In accordance with some embodiments, the observed data is read from a memory. In accordance with some other embodiments, the observed data is received from another system.

[0062] In accordance with some embodiments, the artificial data is generated by sampling the observed data with replacement. In accordance with some other embodiments, the artificial data is generated from the observed data based on a known distribution of the original data. In accordance with some other embodiments, the artificial data is generated independent of the observed data. In accordance with some of these embodiments, the artificial data is generated by sampling state-action pairs from a diffusely mixed generative model; and assigning each state-action pair stochastically optimistic rewards and random state transitions. In accordance with some embodiments, the artificial dataset includes M elements of data where M is approximately equal to or greater than the number of elements in the observed dataset.

[0063] An approximator function is also received as input (**610**). In accordance with some embodiments, the approximator function may be set for process **600** and stored in memory for use. The approximator estimates a state-action value function for a data set. In accordance with some embodiments, the approximator function may be a neural network trained to fit a state-action value function to the data set via a least squared iteration.

[0064] The observed and artificial data are sampled to obtain a training set of data (**615**). In accordance with some of these embodiments, the training data includes M samples of data. In accordance with a number of these embodiments, M is equal to or greater than a number of episodes to observe. In accordance with some embodiments, the sampling is performed in accordance with a known and/or a provided distribution. A bootstrap process is applied to the union of the observed data and the training data to obtain a new distribution and the approximator function is applied to the distribution to generate a randomized state-value function (**620**). For each time step, a state, s, is observed based on the training data and an action, a, is selected based on the state of the system from the sample of data and the policy π (**630**). The reward, r_{lt} realized and resulting transition state, s_{lt+1} are observed (**635**). The state, s, the action, a, and the resulting transition state, s_{lt+1} are stored as resulting data in memory. The selecting (**630**) and observing of the results are repeated until the time period ends (**640**). The observed set of data is then updated with the results (**650**). In accordance with some embodiments, a training mask is maintained that indicates the result data from particular time steps of each episode to add to the observed set of data and the mask is read to determine which elements of the resulting data to add to the observed data. This is then repeated for each of the M episodes (**645**) and process **600** ends.

[0065] Although one process for providing reinforcement learning for a deep learning network in accordance with an embodiment of the invention is described with respect to FIG. **6**. Other methods that add, removed, and/or combine steps in process **600** may be performed without departing from the various embodiments of this invention.

[0066] Fitting a model like a deep neural network is a computationally expensive task. As such, it is desirable to use incremental methods to incorporate new data sample into the fitting process as the data is generated. To do so, parallel computing may be used. A process that performs multiple concurrent explorations in accordance with an embodiment of the invention is shown in FIG. **7**.

[0067] Process **700** performs exploration in M distinct episodes for K separate exploration process. In accordance with some embodiments, the number of episodes is received as an input and in accordance with some other embodiments the number of episodes may set or selected by the process based on the size of the deep learning network. A set of data including historical and artificial data is obtained (**705**). In accordance with some embodiments, the observed data is read from a memory. In accordance with some other embodiments, the observed data is received from another system.

[0068] In accordance with some embodiments, the artificial data for one or more exploration processes is generated by sampling the observed data with replacement. In accordance with some other embodiments, the artificial data for one or more exploration processes is generated from the observed data based on a known distribution of the original data. In accordance with some other embodiments, the artificial data for one or more of the exploration processes is generated independent of the observed data. In accordance with some of these embodiments, the artificial data is generated by sampling state-action pairs from a diffusely mixed generative model; and assigning each state-action

pair stochastically optimistic rewards and random state transitions. In accordance with some embodiments, the artificial dataset includes M elements of data where M is approximately equal to or greater than the number of elements in the observed dataset.

[0069] An approximator function is also received as input (**705**). In accordance with some embodiments, the approximator function may be set for process **700** and stored in memory for use. The approximator estimates a state-action value function for a data set. In accordance with some embodiments, the approximator function may be a neural network trained to fit a state-action value function to the data set via a least squared iteration. In accordance with some embodiments, 1 to K number of approximators may be used where each of the 1 to K approximators is applied to the training set data of one or more of the K exploration processes.

[0070] The observed and artificial data are sampled to obtain a training set of data for each of the K independent processes (**715**). In accordance with some of these embodiments, the training data for each exploration process includes M samples of data. In accordance with a number of these embodiments, M is equal to or greater than a number of episodes to observe. In accordance with some embodiments, the sampling for one or exploration processes is performed in accordance with a known and/or a provided distribution. In accordance with some embodiments, one or more of the exploration process may have the same set of artificial data.

[0071] For each of the K exploration processes, a bootstrap process is applied to the union of the observed data and the artificial data to obtain a new distribution and the approximator function is applied to the distribution to generate a randomized state-value function (**720**). For each exploration process, exploration is performed in the following manner. For each time step, a state, s, is observed and an action, a, is selected based on the state of the system from the sample of data and the policy π (**730**). The reward, r_{lt} realized and resulting transition state, s_{lt+1} are observed (**735**). The state, s, the action, a, and the resulting transition state, s_{lt+1} are stored as resulting data in memory. The selecting (**730**) and observing of the results are repeated until the time period for the episode ends (**740**). The observed set of data is individually updated for each exploration process with the results (**750**). To do so, a bootstrap mask may be maintained to indicate the observed data that is available to each exploration process. In accordance with some other embodiments, the observed data is updated with the data from all of the different K exploration processes. In accordance with some embodiments, a training mask is maintained that indicates the result data from particular time steps of each episode for each exploration process to add to the observed set of data and the mask is read to determine which elements of the resulting data to add to the observed data. This is then repeated for each of the M episodes (**745**) and process **700** ends.

[0072] Although one process for providing reinforcement learning for a deep learning network using multiple exploration processes in accordance with an embodiment of the invention is described with respect to FIG. **7**. Other methods that add, removed, and/or combine steps in process **700** may be performed without departing from the various embodiments of this invention.

Testing for Deep Exploration

[0073] The following is an explanation of a series of didactic computational experiments designed to highlight the need for deep exploration. These environments can be described by chains of length N>3 as shown in FIG. **8**. Each episode of interaction lasts N+9 steps after which point an exploration process resets to the initial state, s_2. These are toy problems intended to be expository rather than entirely realistic. Balancing a well known and mildly successful strategy versus an unknown, but potentially more rewarding, approach can emerge in many practical applications.

[0074] The environments in these problems may be described by a finite tabular MDP. However, processes tested only interact with the MDP through raw pixel features. The two feature mappings of the tests are $\phi_{1hot}(s_t):=(1\{x=s_t\})$ and $\phi_{therm}(s_t):=(1\{x\leq s_t\})$ in $\{0, 1\}^N$. The results for ϕ_{therm} were better for all Deep Q-learning Network (DQN) variants due to better generalization, but the difference was relatively small. A Thompson Sampling DQN is the same as a bootstrapped DQN (a deep learning network that uses reinforcement learning provided in accordance with an embodiment of the invention), but resamples every time step. Ensemble DQN uses the same architecture as bootstrapped DQN, but with an ensemble policy.

[0075] For purposes of this discussion, a process has successfully learned the optimal policy when the process has successfully completed one hundred episodes with optimal reward of 10. For each chain length, each learning system was executed for 2000 episodes across three seeds. The median time to learn for each system is shown in FIG. **9**, together with a conservative lower bound of $99+2^{N-11}$ on the expected time to learn for any shallow exploration strategy. As seen in graphs **901-904**, only bootstrapped DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) demonstrates a graceful scaling to long chains which require deep exploration.

[0076] Bootstrapped DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) explores in a manner similar to the provably-efficient algorithm PSRL but bootstrap DQN uses a bootstrapped neural network to approximate a posterior sample for the value. Unlike PSRL, bootstrapped DQN directly samples a value function and does not require further planning steps. The bootstrap DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) is similar to RLSVI, which is also provably-efficient, but with a neural network instead of linear value function and bootstrap instead of Gaussian sampling. The analysis for the linear setting suggests that this nonlinear approach will work well as long as the distribution $\{Q^1, \ldots, Q^K\}$ remains stochastically optimistic, or at least as spread out as the "correct" posterior.

[0077] Bootstrapped DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) relies upon random initialization of the network weights as a prior to induce diversity. The initial diversity is enough to maintain diverse generalization to new and unseen states for large and deep neural networks. The initial diversity is effective for this experimental setting, but will not work in all situations. In general, a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of

the invention may be necessary to maintain some more rigorous notion of "prior", potentially through the use of artificial prior data to maintain diversity. One potential explanation for the efficacy of simple random initialization is that unlike supervised learning or bandits, where all networks fit the same data, each of Q^K heads has a unique target network. This, together with stochastic minibatch and flexible nonlinear representations, means that even small differences at initialization may become bigger as the heads refit to unique TD errors.

Atari Evaluation

[0078] A deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention was evaluated across 49 Atari games on the Arcade Learning Environment. The domains of these games are not specifically designed to showcase the tested deep learning network. In fact, many Atari games are structured so that small rewards always indicate part of an optimal policy that may be crucial for the strong performance observed by dithering strategies. The evaluations show that exploration via bootstrapped DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) produces significant gains versus ϵ-greedy in this setting. Bootstrapped DQN reaches peak performance roughly similar to DQN. However, the improved exploration of the bootstrapped DQN reaches human performance on average 30% faster across all games. This translates to significantly improved cumulative rewards through learning.

Deep Learning Network Set-Up for Atari Evaluation

[0079] The bootstrapped DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) evaluated had a network structure is identical to the convolutional structure of a DQN except the bootstrapped DQN split 10 separate bootstrap heads after the convolutional layer.

[0080] 49 Atari games were used as for our experiments. Each step of the process corresponds to four steps of the emulator, where the same action is repeated, the reward values of the process are clipped between −1 and 1 for stability. The processes are evaluated and reported performance based upon the raw scores.

[0081] The convolutional part of the network (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) used is identical to the one used in other systems. The input to the network is 4×84×84 tensor with a rescaled, grayscale version of the last four observations. The first convolutional (cony) layer has 32 filters of size 8 with a stride of 4. The second cony layer has 64 filters of size 4 with stride 2. The las cony layer has 64 filters of size 3. We split the network beyond the final layer into K=10 distinct heads, each one is fully connected and identical to the single head of a DQN that includes a fully connected layer to 512 units followed by another fully connected layer to the Q-Values for each action. The fully connected layers all use Rectified Linear Units (ReLU) as a non-linearity. Gradients 1/K that flow from each head are normalized.

[0082] Each of the networks tested were trained with RMSProp with a momentum of 0.95 and a learning rate of 0.00025. The discount was set to γ=0.99, the number of steps between target updates was set to τ=10000 steps. The processes were trained for a total of 50 m steps per game, which corresponds to 200 m frames. The processes were stopped every 1 m frames for evaluation. Furthermore, the bootstrapped DQN used an ensemble voting policy. The experience replay contains the 1 m most recent transitions. The network was updated every 4 steps by randomly sampling a minibatch of 32 transitions from the replay buffer to use the exact same minibatch schedule as DQN. For training, a ϵ-greedy policy with ϵ being annealed linearly from 1 to 0.01 over the first 1 m timesteps.

Gradient Normalization in Bootstrap Heads

[0083] Most literature in deep reinforcement learning for Atari focuses on learning the best single evaluation policy, with particular attention to whether this above or below human performance. This is unusual for the reinforcement learning literature, which typically focuses upon cumulative or final performance.

[0084] Based on the results, bootstrapped DQN (a deep learning network that uses reinforcement learning as provided in accordance with an embodiment of the invention) makes significant improvements to the cumulative rewards of DQN on Atari, while the peak performance is much more. Furthermore, using bootstrapped DQN without gradient normalization on each head typically learned even faster than our implementation with rescaling **1**/K, but the network was somewhat prone to premature and suboptimal convergence.

[0085] In order to better the benchmark "best" policies reported by DQN, bootstrapped DQN should use the gradient normalization. However, it is not entirely clear whether gradient normalization represents an improvement for all settings.

[0086] Where a reinforcement learning system is deployed to learn with real interactions, cumulative rewards present a better measure for performance. In these settings, the benefits of gradient normalization are less clear. However, even with normalization 1/K bootstrapped DQN significantly outperforms DQN in terms of cumulative rewards.

Sharing Data in Bootstrap Heads

[0087] In the Atari tests, all network heads (exploration processes) of the bootstrapped DQN share all the data, so the bootstrapped are not actually a traditional bootstrap at all. This is different from the regression task, where bootstrapped data was essential to obtain meaningful uncertainty estimates. There are several theories for why the networks maintain significant diversity even without data bootstrapping in this setting.

[0088] First, the network heads all train on different target networks. As such, when facing the same (s, a, r, ś) datapoint, the various heads can reach drastically different Q-value updates. Second, Atari is a deterministic environment and any transition observation is the unique correct datapoint for this type of setting. Third, the networks are deep and the heads are initialized from different random values so the heads will likely find quite diverse generalization even when the heads agree on given data. Finally, since all variants of DQN take many frames to update policy, it is likely that even using ρ=0.5 the heads would still populate their replay memory with identical datapoints. Thus, using ρ=1 to save on minibatch passes seems like a

reasonable compromise and the use of ρ=1 doesn't seem to negatively affect performance too much in this setting. More research is needed to examine exactly where/when this data sharing is important.

Results Tables

[0089] In Table 1, shown in FIG. **13**, the average score achieved by the various systems are shown during the most successful evaluation period and compared to human performance and a uniformly random policy. DQN is an implementation of DQN with the hyperparameters specified above, using the double Q-Learning update. The peak final performance of DQN is similar under bootstrapped DQN to previous benchmarks.

[0090] To compare the benefits of exploration via bootstrapped DQN, the results of bootstrapped DQN are benchmarked our performance against the most similar prior work on incentivizing exploration in Atari. To do so, the bootstrapped DQN is compared to AUC-100. Based on the results, bootstrapped DQN out performs this prior work significantly.

Implementing Bootstrapped DQN at Scale

[0091] In the evaluations, the number of heads needed to generate online bootstrap samples for DQN in computationally efficient manner was evaluated. The following three key questions need to be answered to determine the optimal number of heads: how many heads needed, how should gradients be passed to the shared network and how should data be bootstrapped online? To do so, significant compromises were made in order to maintain computational cost comparable to DQN.

[0092] More heads leads to faster learning, but even a small number of heads captures most of the benefits of bootstrapped DQN. For the evaluations, K=10 was used.

[0093] The shared network architecture allows training of this combined network via backpropagation. Feeding K network heads to the shared convolutional network effectively increases the learning rate for this portion of the network. In some games, the increased learning rate leads to premature and sub-optimal convergence. The best final scores were achieved by normalizing the gradients by 1/K, but the normalizing of the gradients also leads to early learning.

[0094] To implement an online bootstrap, an independent Bernoulli mask $w_1, \ldots, w_K \sim \text{Ber}(p)$ was used for each head in each episode. These flags are stored in the memory replay buffer and identify which heads are trained on which data. However, when trained using a shared minibatch the network will also require an effective 1/p more iterations; this is undesirable computationally. Surprisingly, the bootstrapped DQN performed similarly irrespective of p and all outperformed DQN. In light of empirical observation for Atari, p=1 to is used save on minibatch passes. As a result, bootstrapped DQN runs at a similar computational speed to vanilla DQN on identical hardware.

Efficient Exploration in Atari

[0095] In the evaluations, bootstrapped DQN drives efficient exploration in several Atari games. For the same amount of game experience, bootstrapped DQN generally outperforms DQN with ϵ-greedy exploration. FIG. **10** demonstrates this effect for a diverse section of games.

[0096] On games where DQN performs well, bootstrapped DQN typically performs better. Bootstrapped DQN does not reach human performance on Amidar (DQN does) but does on Beam Rider and Battle Zone (DQN does not). To summarize this improvement in learning time, the number of frames required to reach human performance is considered. If bootstrapped DQN reaches human performance in 1/x frames of DQN, bootstrapped DQN has improved by x. FIG. **11** shows that Bootstrapped DQN typically reaches human performance significantly faster.

[0097] On most games where DQN does not reach human performance, bootstrapped DQN does not solve problem by itself. On some challenging Atari games where deep exploration is conjectured to be important, the results for bootstrapped DQN are not entirely successful, but sill promising. In Frostbite, bootstrapped DQN reaches the second level much faster than DQN but network instabilities cause the performance to crash. In Montezuma's Revenge, bootstrapped DQN reaches the first key after 20 m frames (DQN never observes a reward even after 200 m frames) but does not properly learn from this experience. Our results suggest that improved exploration may help to solve these remaining games, but also highlight the importance of other problems like network instability, reward clipping and temporally extended rewards.

Overall Performance

[0098] Bootstrapped DQN is able to learn much faster than DQN. Graph **1201** of FIG. **12** shows that bootstrapped DQN also improves upon the final score across most games. However, the real benefits to efficient exploration mean that bootstrapped DQN outperforms DQN by orders of magnitude in terms of the cumulative rewards through learning (shown in graph **1202** of FIG. **12**. In both graphs, performance is normalized relative to a fully random policy. The most similar work to bootstrapped DQN presents several other approaches to improved exploration in Atari. For example, AUC-20 is optimized for a normalized version of the cumulative returns after 20 m frames. According to this metric, averaged across the 14 games considered, bootstrapped DQN improve upon both base DQN (0.29) and the AUC-20 best method (0.37) to obtain 0.62. These results together with results tables across all 49 games are provided in the table shown in FIG. **13**.

Visualizing Bootstrapped DQN

[0099] The following is some more insight to how bootstrapped DQN drives deep exploration in Atari. In each game, although each head $Q^1, \ldots, Q^{10}$ learns a high scoring policy, the policies found by each head are quite distinct. Although each head performs well, each follows a unique policy. By contrast, ϵ-greedy strategies are almost indistinguishable for small values of ϵ and totally ineffectual for larger values. Deep exploration is key to improved learning, since diverse experiences allow for better generalization.

[0100] Disregarding exploration, bootstrapped DQN may be beneficial as a purely exploitative policy. In the evaluations, all of the heads are combined into a single ensemble policy, for example by choosing the action with the most votes across heads. This approach might have several benefits. First, the ensemble policy can often outperform any individual policy. Second, the distribution of votes across heads to give a measure of the uncertainty in the optimal

policy. Unlike a conventional DQN, bootstrapped DQN can know what it doesn't know. In an application where executing a poorly-understood action is dangerous this could be crucial, the uncertainty in this policy is surprisingly interpretable: all heads agree at clearly crucial decision points, but remain diverse at other less important steps.

[0101] Although the present invention has been described in certain specific aspects, many additional modifications and variations would be apparent to those skilled in the art. It is therefore to be understood that the present invention can be practiced otherwise than specifically described without departing from the scope and spirit of the present invention. Thus, embodiments of the present invention should be considered in all respects as illustrative and not restrictive. Accordingly, the scope of the invention should be determined not by the embodiments illustrated, but by the appended claims and their equivalents.

What is claimed is:

1. A deep learning system comprising:
at least one processor;
memory accessible by each at least one processor;
instructions that when read by the at least one processor direct the at least one processor to:
maintain a deep neural network; and
apply a reinforcement learning process to the deep neural network where the reinforcement learning process includes:
receive a set of observed data and a set artificial data,
for each of one or more episodes:
sample from a set of data that is a union of the set of observed data and the set of artificial data to generate set of training data;
determine a state-action value function for the set of training data using a bootstrap process and an approximator where the approximator that estimates a state-action function for a dataset;
for each time step in each one or more episode:
determine a state of the system for a current time step from the set of training data;
select an action based on the determined state of the system and a policy mapping actions to the state of the system;
determine results for the action including a reward and a transition state that result from the selected action; and
store result data for the current time step that includes the state, the action, the transition state, and
update the set of the observed data with the result data from at least one time step for each of the one or more the episodes.

2. The deep learning system of claim **1** wherein the instructions further direct the at least one processor to generate the set of artificial data from the set of observed data.

3. The deep learning system of claim **2** wherein the instructions to generate the artificial data include instruction that direct the at least one processor:
sample the set of observed data with replacement to generate the set of artificial data.

4. The deep learning system of claim **2** wherein the instructions to generate the artificial data include instructions that direct the at least one processor to:
sample a plurality state-action pairs from a diffusely mixed generative model; and
assign each of the plurality of sampled state-action pairs stochastically optimistic rewards and random state transitions.

5. The deep learning system of claim **1** wherein the instructions further direct the at least one processor to:
maintain a training mask that indicates the result data from each of the time period in each episode to be used in training; and
wherein the updating of the set of observed data includes adding the result data from each time period of an episode indicated in the training mask.

6. The deep learning network of claim **1** where the instructions further direct the processor to:
receive the approximator as an input.

7. The deep learning network of claim **1** wherein the instructions further direct the processor to:
read the approximator from memory.

8. The deep learning network of claim **1** wherein the approximator is a neural network trained to fit a state-action value function to the data set via a least squared iteration.

9. The deep learning network of claim **1** wherein a plurality of reinforcement learning processes are applied to the deep neural network.

10. The deep learning network of claim **9** wherein each of the plurality of reinforcement learning processes independently maintain the set of observed data.

11. The deep learning network of claim **9** wherein the plurality of reinforcement learning processes cooperatively maintain the set of observed data.

12. The deep learning process of claim **9** wherein the instruction further direct the processor to:
maintain a bootstrap mask that indicates each element in the set of observed data that is available to each of the plurality of reinforcement learning process.

13. A method performed by at least one processor executing instructions stored in memory to perform the method to provide reinforcement learning in a deep learning network, the method comprising:
receiving a set of observed data and a set artificial data;
for each of one or more episodes:
sampling from a set of data that is a union of the set of observed data and the set of artificial data to generate set of training data,
determining a state-action value function for the set of training data using a bootstrap process and an approximator where the approximator that estimates a state-action function for a dataset,
for each time step in each one or more episode:
determining a state of the system for a current time step from the set of training data;
selecting an action based on the determined state of the system and a policy mapping actions to the state of the system;
determining results for the action including a reward and a transition state that result from the selected action; and
storing result data for the current time step that includes the state, the action, the transition state, and
updating the set of the observed data with the result data from at least one time step of each of the one or more episodes.

14. The method of claim **13** further comprising generating the set of artificial data from the set of observed data.

15. The method of claim **14** further comprising:

sampling the set of observed data with replacement to generate the set of artificial data.

16. The method of claim **14** further comprising:

sampling a plurality state-action pairs from a diffusely mixed generative model; and

assigning each of the plurality of sampled state-action pairs stochastically optimistic rewards and random state transitions.

17. The method of claim **13** further comprising:

maintaining a training mask that indicates the result data from each of the time period in each episode to be used in training; and

wherein the updating of the set of observed data includes adding the result data from each time period of an episode indicated in the training mask.

18. The method of claim **13** further comprising:

receiving the approximator as an input.

19. The method of claim **13** further comprising:

read the approximator from memory.

20. The method of claim **13** wherein the approximator is a neural network trained to fit a state-action value function to the data set via a least squared iteration.

21. The method of claim **13** wherein a plurality of reinforcement learning methods are applied to the deep neural network.

22. The method of claim **21** wherein each of the plurality of reinforcement learning methods independently maintain the set of observed data.

23. The method of claim **21** wherein the plurality of reinforcement learning methods cooperatively maintain the set of observed data.

24. The method of claim **21** further comprising:

maintaining a bootstrap mask that indicates each element in the set of observed data that is available to each of the plurality of reinforcement learning process.

* * * * *