हर काम देश के नाम

NAVIGATION AND CONTROL OF AUTONOMOUS UNDERWATER VEHICLE USING DEEP REINFORCEMENT LEARNING

CMDE VKP PATHAK, CMDE(API),
IHQ-MOD(NAVY)/DAPI

WHAT IS AUV?

Unmanned vehicles for use under sea

Autonomy of operation

Military & commercial applications

Self carried batteries and streamline configuration

Autonomous capability for environmental perception, self localisation, analysis, decision making & operating missions

PATH FOLLOWING & TRAJECTORY TRACKING

Objective of control system – collision free optimal path & minimum tracking error

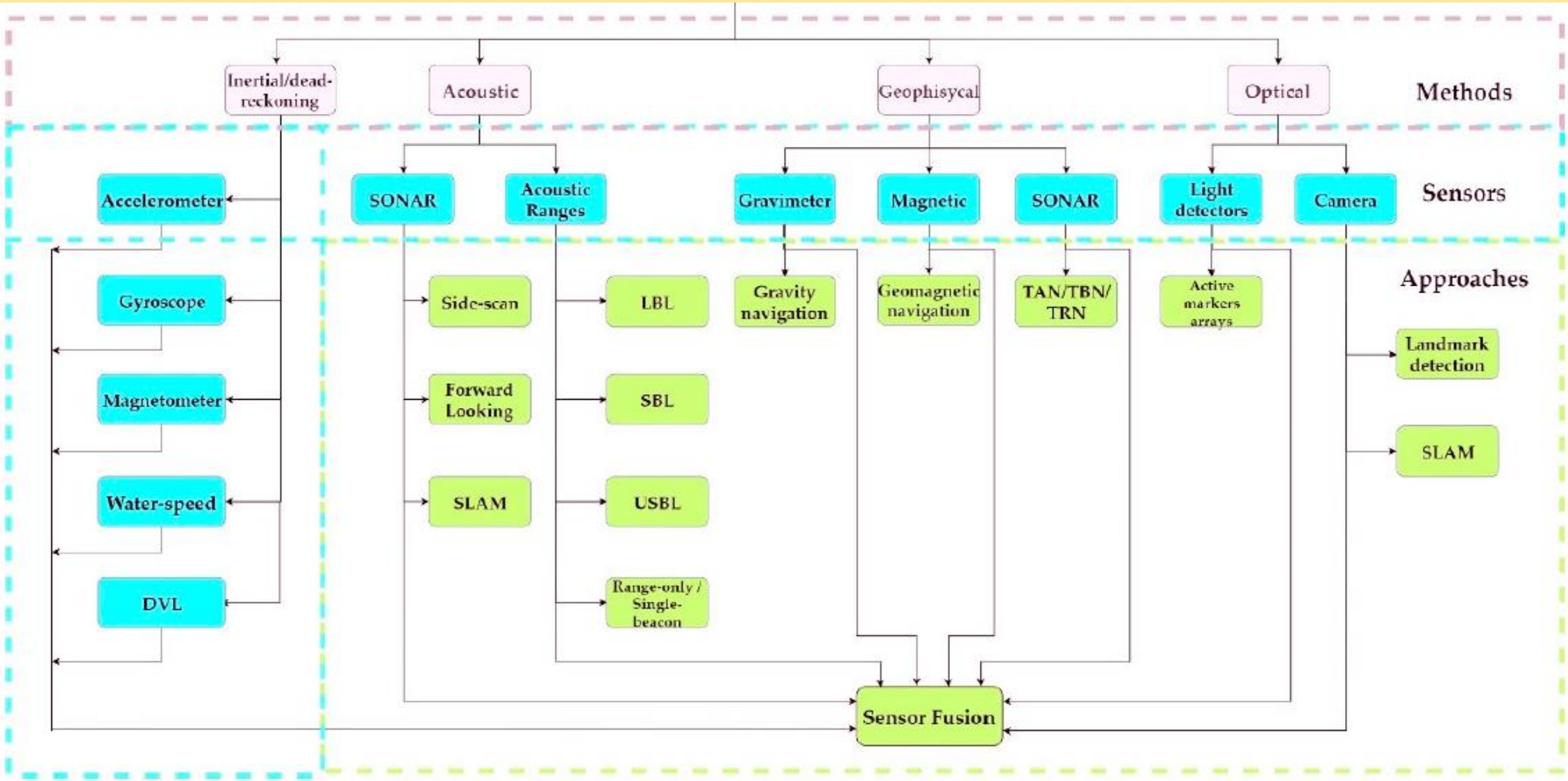
Localisation and navigation using Dead-Reckoning (DR) and Inertial Navigation Systems (INS)

Accuracy improvement through use of Doppler Velocity Log (DVL) or Terrain Aided Navigation (TAN) or GPS

Sensor fusion module used to improve state estimation by processing and merging available sensors data

Acoustic & vision based system for accuracies at short ranges

SENSOR FUSION



COLLISION AVOIDANCE

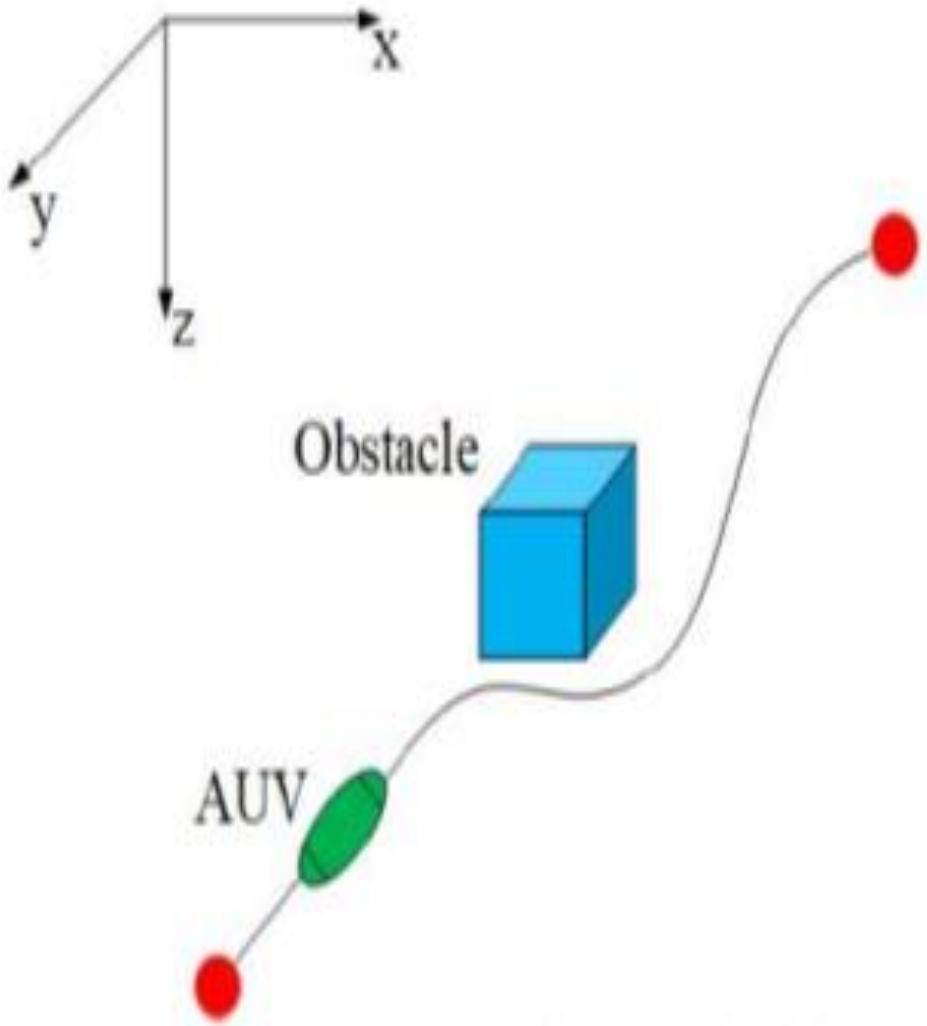
Inputs – INS, Sonar, optical cameras, DVL etc.

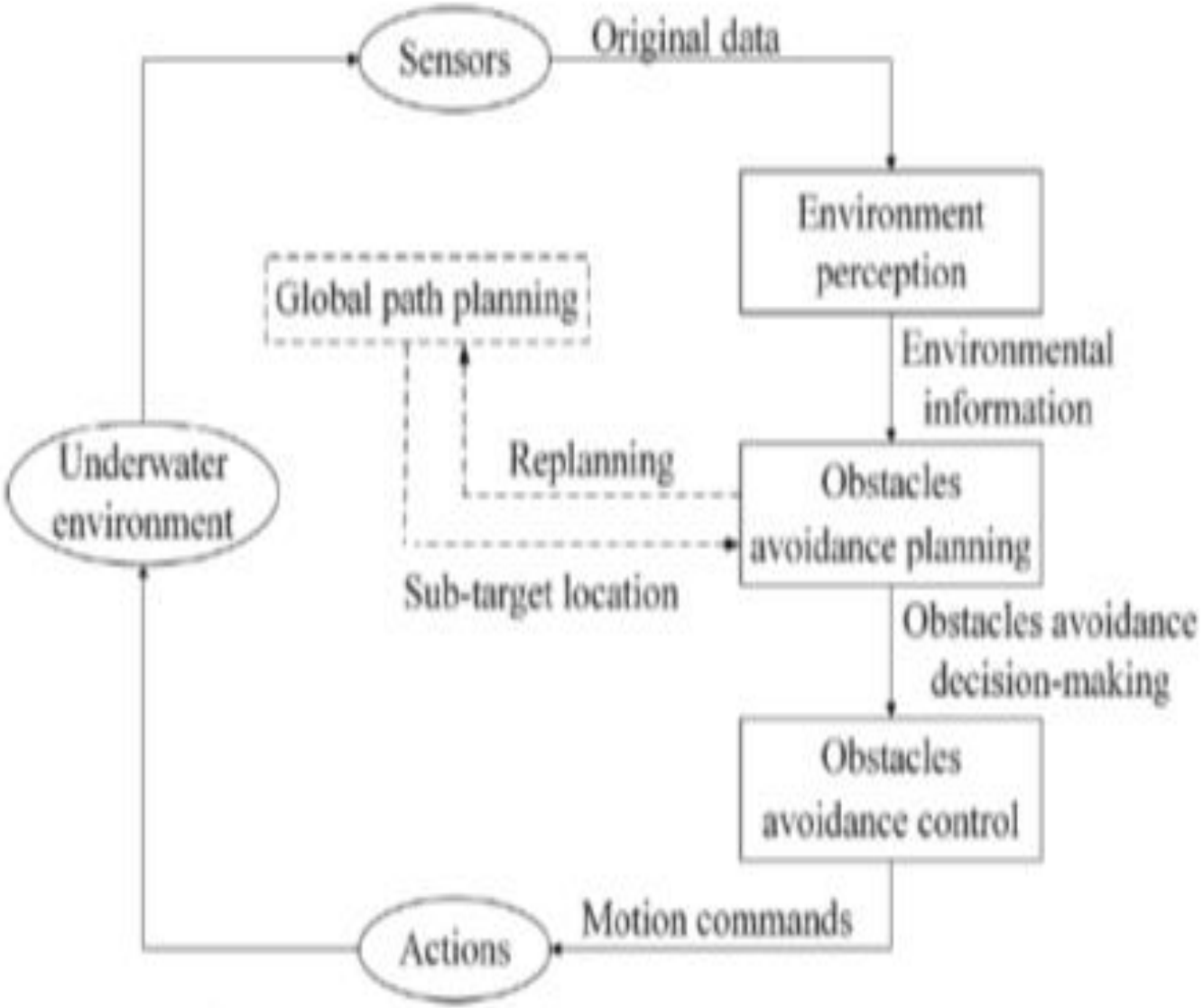
Output – Steering commands to control surfaces

Detection and avoidance logic - environnemental perception, obstacle avoidance planning and obstacle avoidance control

Challenges – Slow speed, control surface/ thruster efficiency, reaction and response time, kinematic and dynamic constraints

COLLISION AVOIDANCE

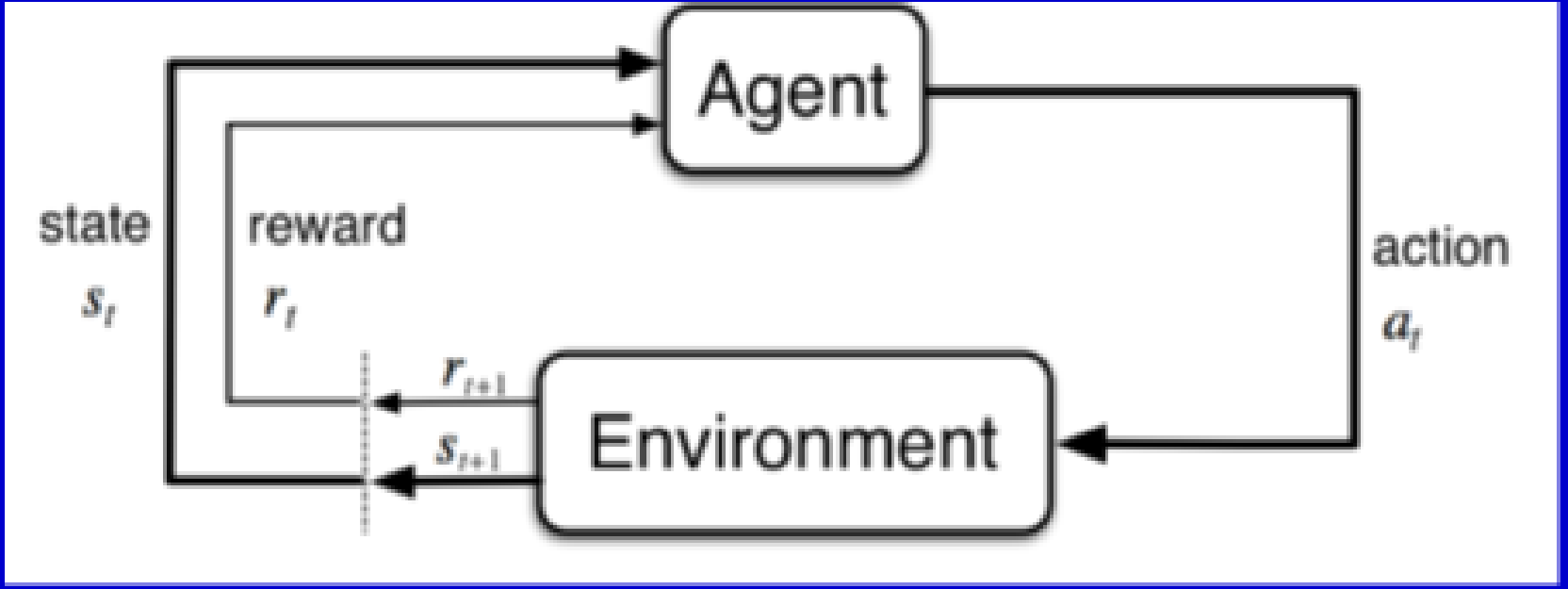




REINFORCEMENT LEARNING (RL)

Markov decision process (MDP)

Basic framework of RL:-



REINFORCEMENT LEARNING (RL)

Markov property – mapping of states to action

Markov process - future states depends only upon present state, not on preceded sequence

Markov decision processes - mathematical framework for modelling decision making $\pi(a|s) = p(a_t = a|s_t = s)$

Maximising long time cumulative rewards (with discount factor to address uncertainty) $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots\ldots\ldots = R_{t+1} + \gamma G_{t+1}$

REINFORCEMENT LEARNING (RL)

Value function - degree of goodness or badness for an agent in a certain state

'State-value' function $V(s) = E(G_t|s_t = s)$

Action-value function $Q(s,a) = E(G_t|s_t = s,\ a_t = a)]$

Relation between action-value & state-value $V(s) = \sum_{a \in A} \pi(a|s) Q_\pi(s,a)$

REINFORCEMENT LEARNING (RL)

Bellman equation $V_\pi(s) = E[R_{t+1} + \gamma V\pi(st = s)$

Objective of reinforcement learning is to find an optimal policy by maximizing value function V(s) or Q(s;a)

$$V^*(s) = max_\pi V(s)$$

$$Q^*(s,a) = max_\pi Q(s,a)$$

DEEP REINFORCEMENT LEARNING

Combination of neural network and Q-learning algorithm

Prediction network $Q(s;a;\theta)$ and target network $Q'(s;a;\theta')$

Experience replay mechanism

Parameters of Q network updated through gradient back propagation using MSE loss function

TEMPORAL DIFFERENCE

$$TD_t(a_t, s_t) = r_t + \gamma \max\big(Q(a, s_{t+1})\big) - Q(a_t, s_t)$$

Aids in learning Q-value - higher TD is good and small TD is bad for learning of Q-value

Q-value is thus augmented or reinforced from time 't-1' to time 't'

$$Q_t(a_t, s_t) = Q_{t-1}(a_t, s_t) + \alpha TD_t(a_t, s_t)$$

TEMPORAL DIFFERENCE

Q-values measure accumulation of good or bad TDs which are associated with the action-state pair (at; st).

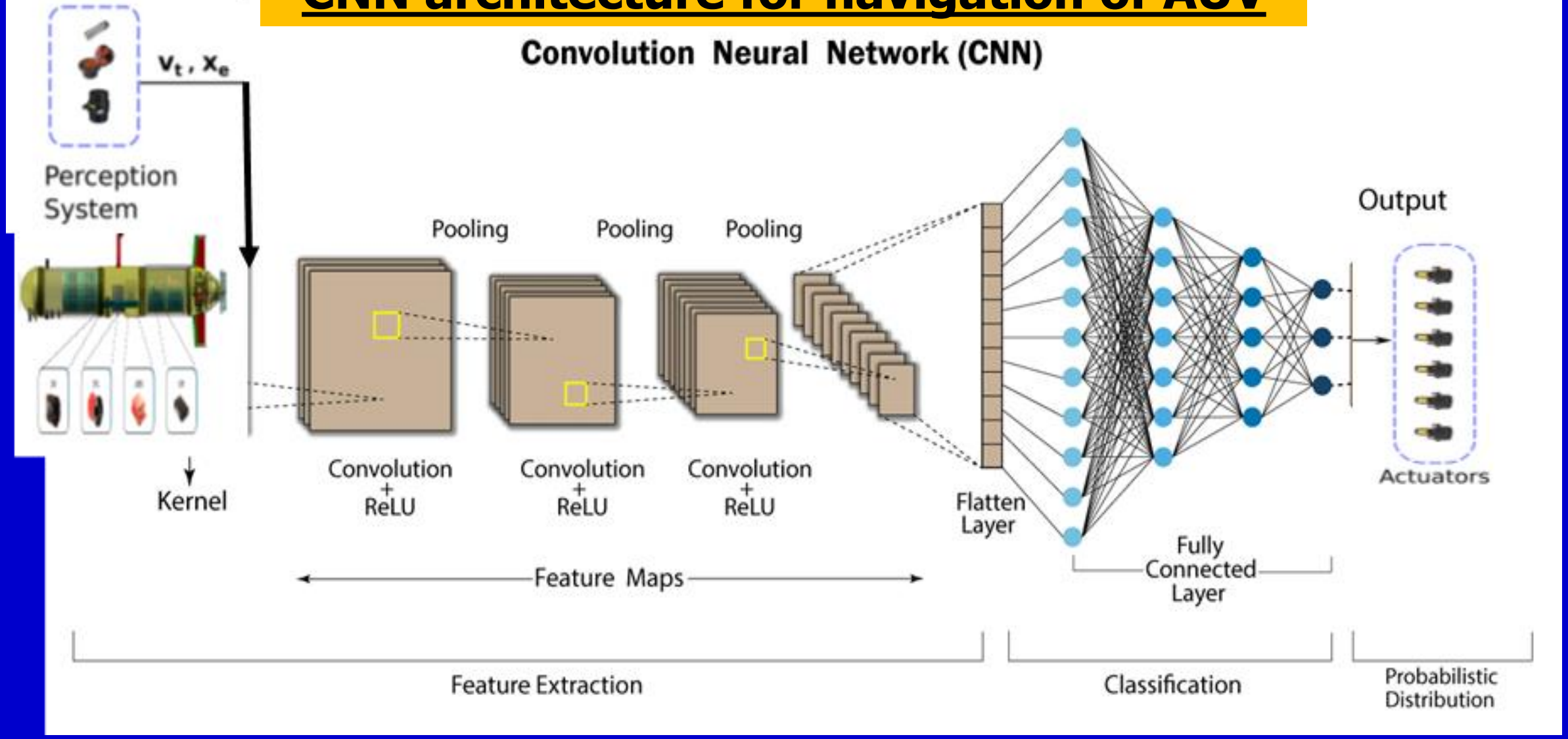
AI is reinforced for good cases, and weakened in bad cases

maximum of Q(at;st) $\mathrm{a} = \underset{a}{\mathrm{argmax}}[Q(a,s)]$

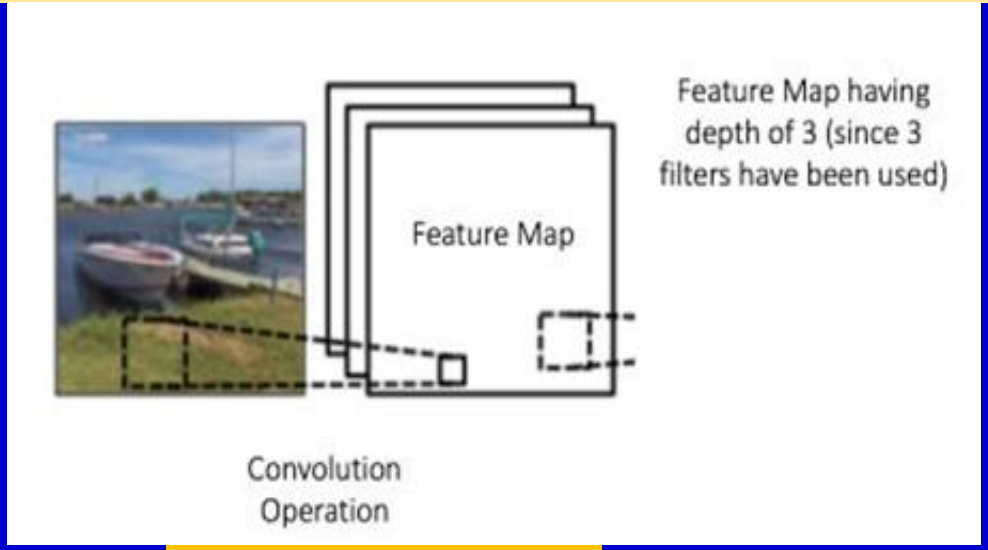
Softmax method $W_s: a \in A \rightarrow \frac{\exp\left(Q(s,a)\right)^{\tau}}{\sum_{a'} \exp\left(Q(s,a')\right)^{\tau}}$ $with\ \tau \geq 0$

DEEP Q-LEARNING FOR NAVIGATION OF AUV

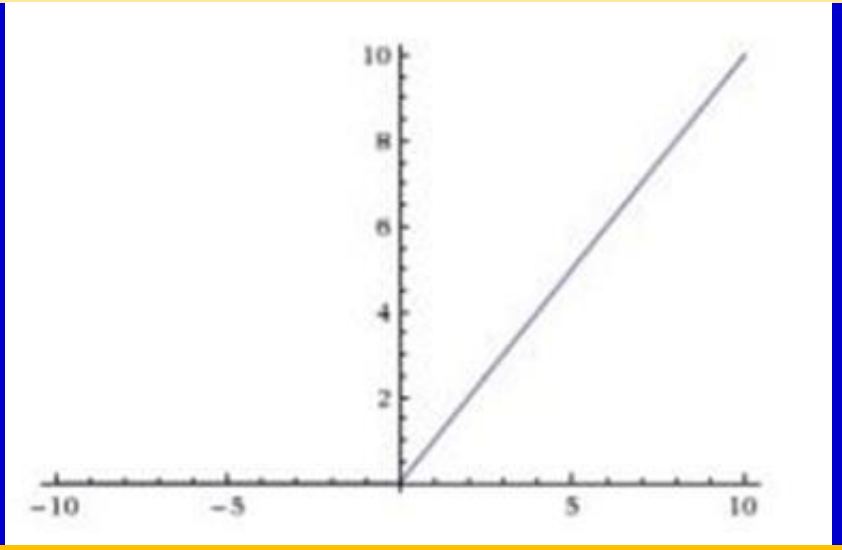
CNN architecture for navigation of AUV



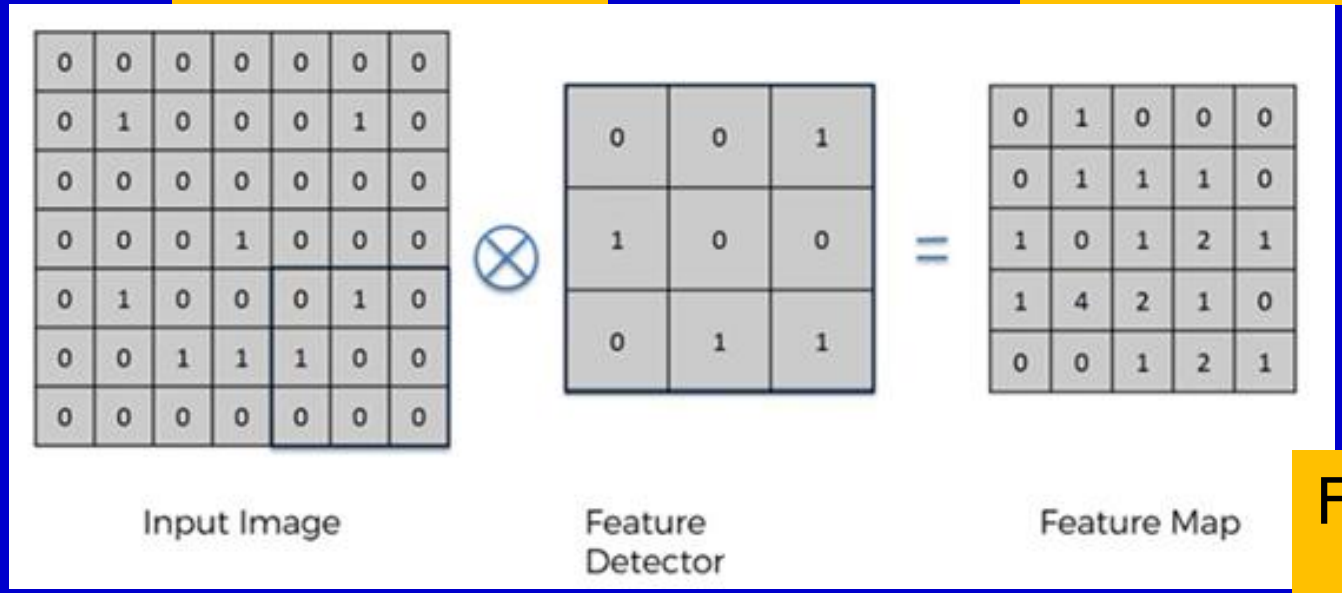
DEEP Q-LEARNING FOR NAVIGATION OF AUV



Feature maps

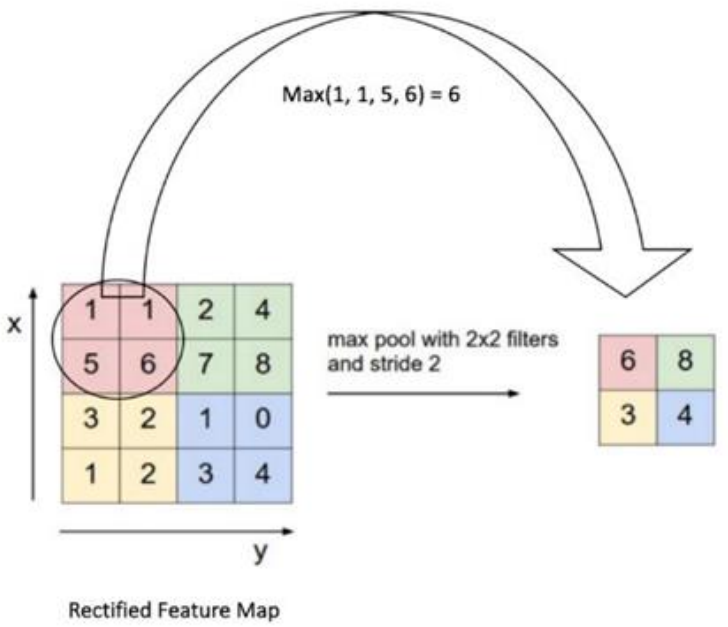


Relu function (Output – Max (Zero, Input)

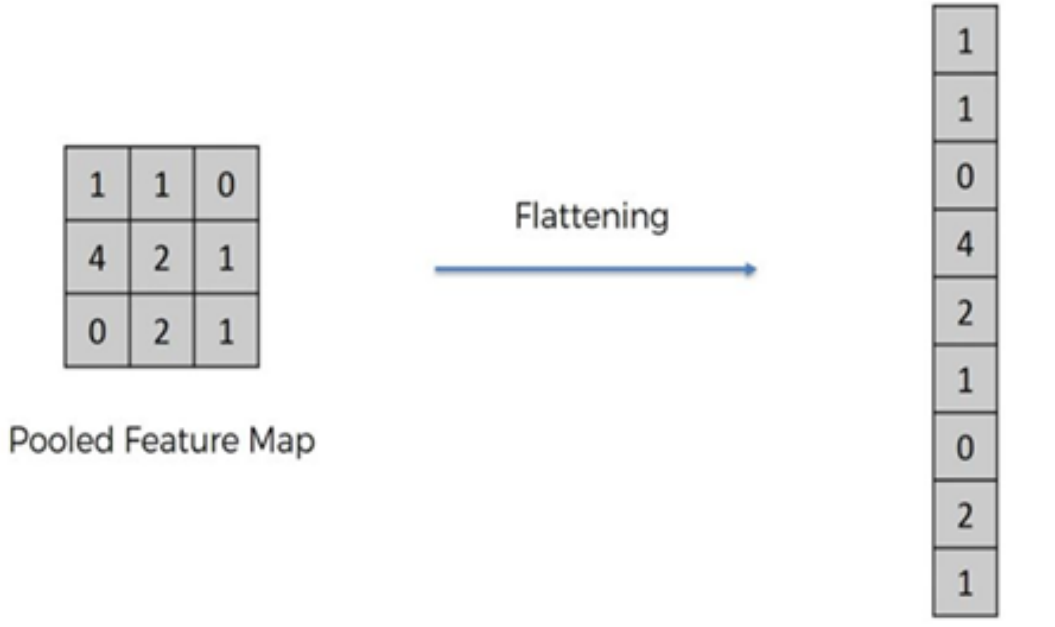


Feature map generated by convolution operation

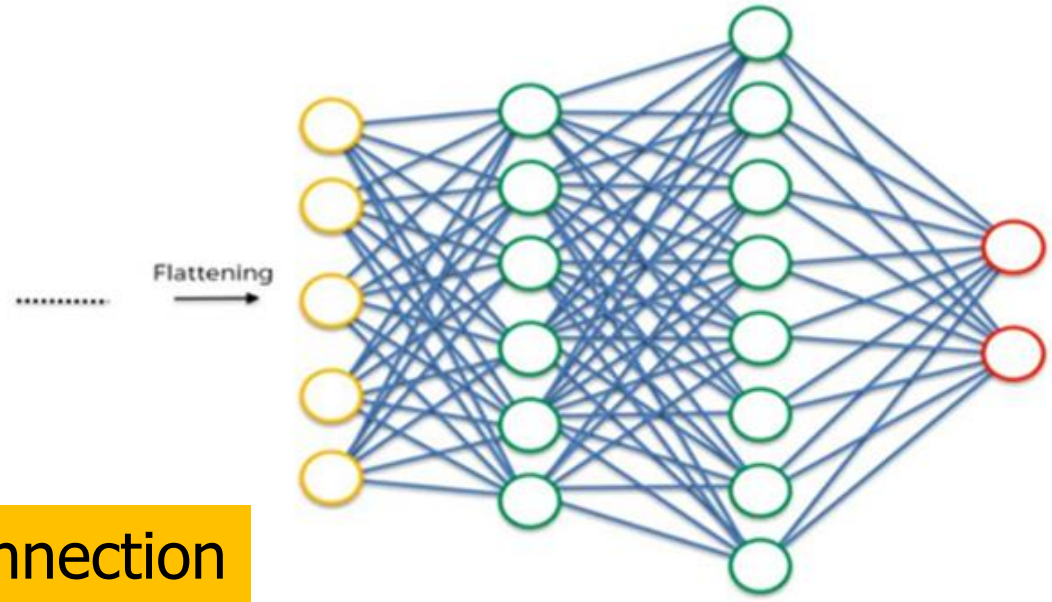
DEEP Q-LEARNING FOR NAVIGATION OF AUV



Max pooling



Flattening



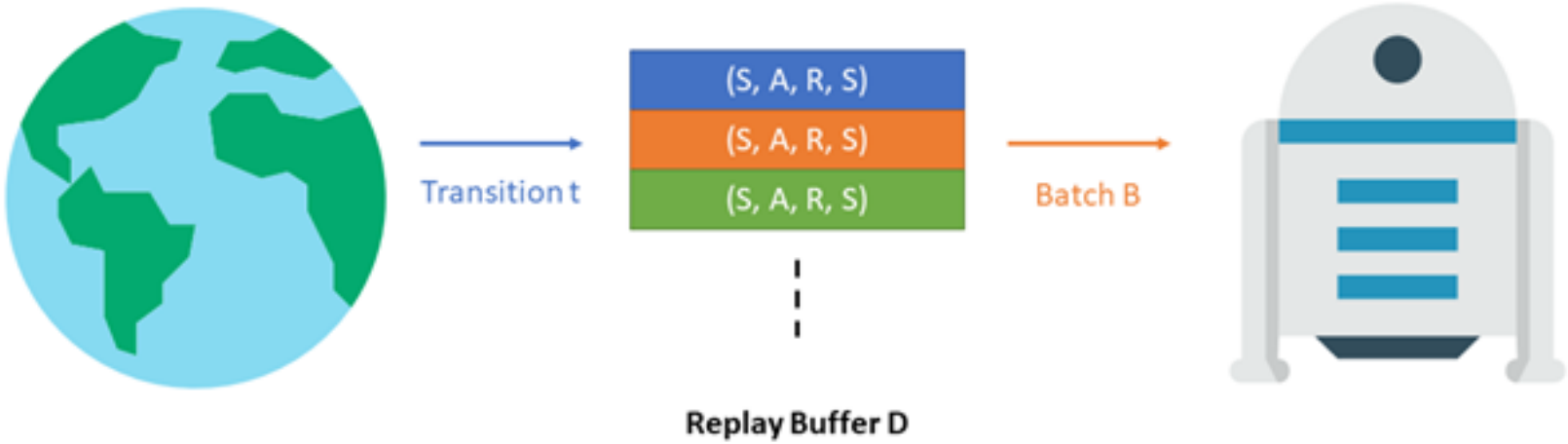
Full Connection

DEEP Q-LEARNING FOR NAVIGATION OF AUV

Experience replay - learning from batch of experiences instead of frame by frame or action after action learning

Experience replay breaks the pattern of sequential bias

- Save transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ into buffer and sample batch B
- Use batch B to train the agent

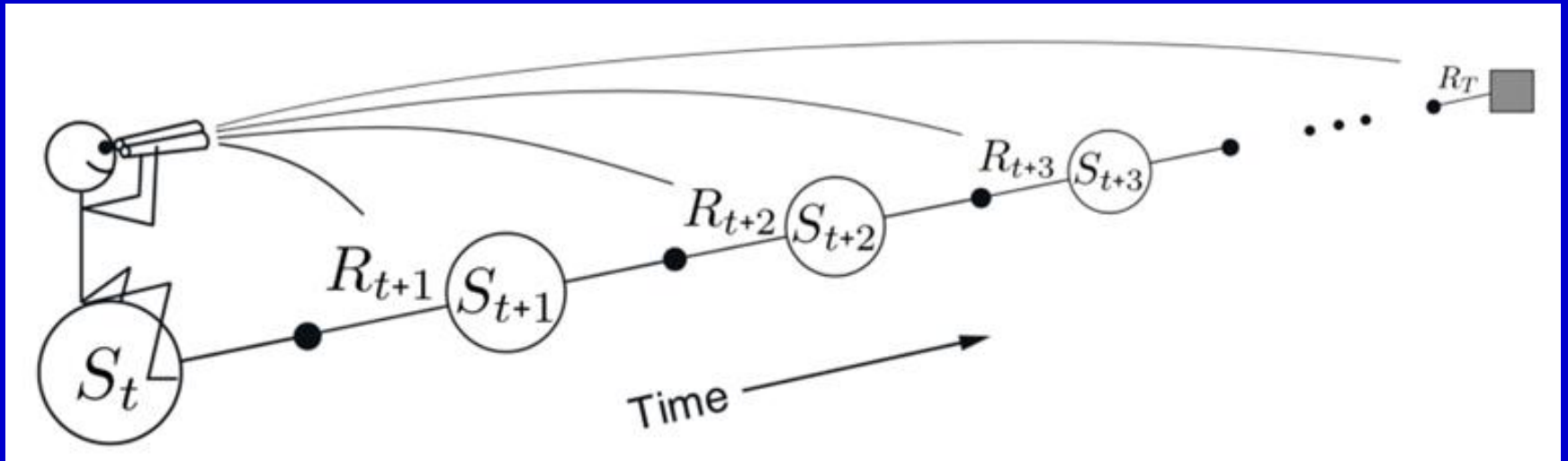


DEEP Q-LEARNING FOR NAVIGATION OF AUV

Eligibility trace - technique used to optimise learning by network

Agent will take N-steps and calculate total reward obtained from these N-steps

Eligibility trace provides information of reward or penalty obtained and specific step responsible for the same



ALGORITHM FOR DEEP Q-LEARNING

Q-values of all action ‘a’ and state ‘s’ pairs are initialized to 0

$$\forall a \in A, s \in S, Q_0(a,s) = 0$$

Experience Replay is initialized to an empty list M

With the initial state ‘s0’, random action is selected and first state ‘s1’ is reached.

At each time t ≥ 1, action ‘at’ is taken, where ‘at’ is a random draw from the ‘Ws’ distribution

$$a_t \sim W_{st}(.) = \frac{\exp(Q(s_t,.))^\tau}{\sum_{a'} \exp(Q(s_t,a'))^\tau}, with\ \tau \geq 0$$

ALGORITHM FOR DEEP Q-LEARNING

Reward ‘rt’ = R(at;st) is obtained

Next state ‘st+1’ is selected and taken, where ‘st+1’ is a random draw from the T(at;st; .) distribution

Transition (st; at; rt; st+1) is appended in M

A random batch is taken from $B \subset M$ of transitions

ALGORITHM FOR DEEP Q-LEARNING

For each transition (stB; atB; rtB; stB+1) of the random batch B:-

(i) Prediction is obtained :-

$$Q(s_{tB}, a_{tB})$$

(ii) Target is obtained:-

$$r_t + \gamma \max_a [Q(a, s_{t+1})]$$

(iii) Loss is calculated:-

$$Loss = \frac{1}{2}\left([r_t + \gamma \max_a [Q(a, s_{t+1})]\right) - Q(a_t, s_t))^2 = \frac{1}{2} TD_t(a_t, s_t)^2$$

Loss error is back-propagated to update weights according to how much they contributed to the error

ASYNCHRONOUS ACTOR-CRITIC AGENTS (A3C)

'n' agents A1, A2,..., An - each agent shares two networks viz. the actor and the critic

Critic evaluates present states, while actor evaluates possible values in present state

Actor is used to make decisions

At each epoch time of training for an agent, it takes the last version of shared networks and uses actor during 'n' steps in order to make a decision

ASYNCHRONOUS ACTOR-CRITIC AGENTS (A3C)

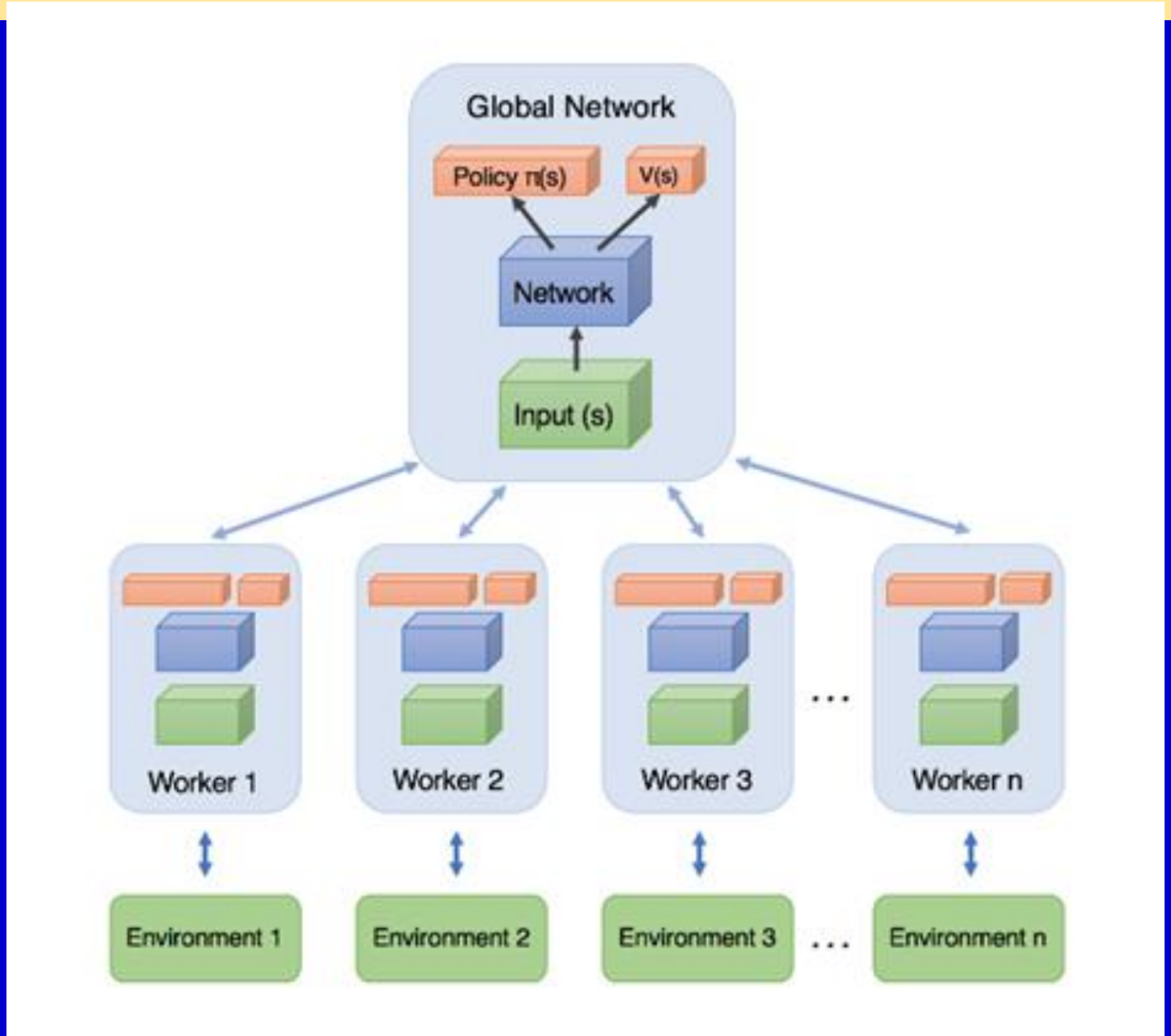
Over ‘n’ steps, it collects all observed new states, the values of these new states, the rewards, etc.

After the n steps, agent uses collected observations in order to update the shared models

Times of epoch and the times of updates of shared network by agent are asynchronous

Problem of an agent getting stuck in local minima is reduced by using A3C algorithm

ASYNCHRONOUS ACTOR-CRITIC AGENTS (A3C)



ACTOR-CRITIC

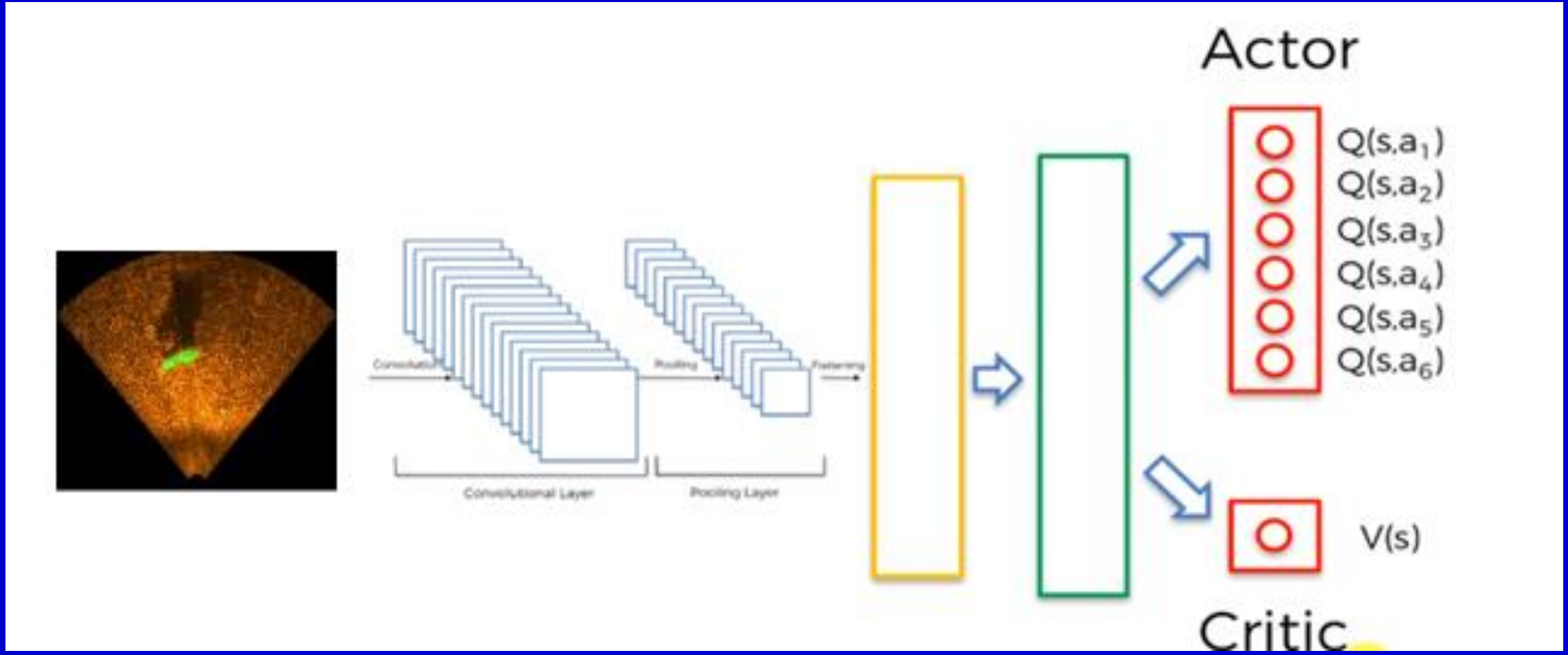
In A3C, there are two outputs – 1st is q-values for different actions and 2nd is value in that state

Critic measures how good the action taken is (value-based) V(s)

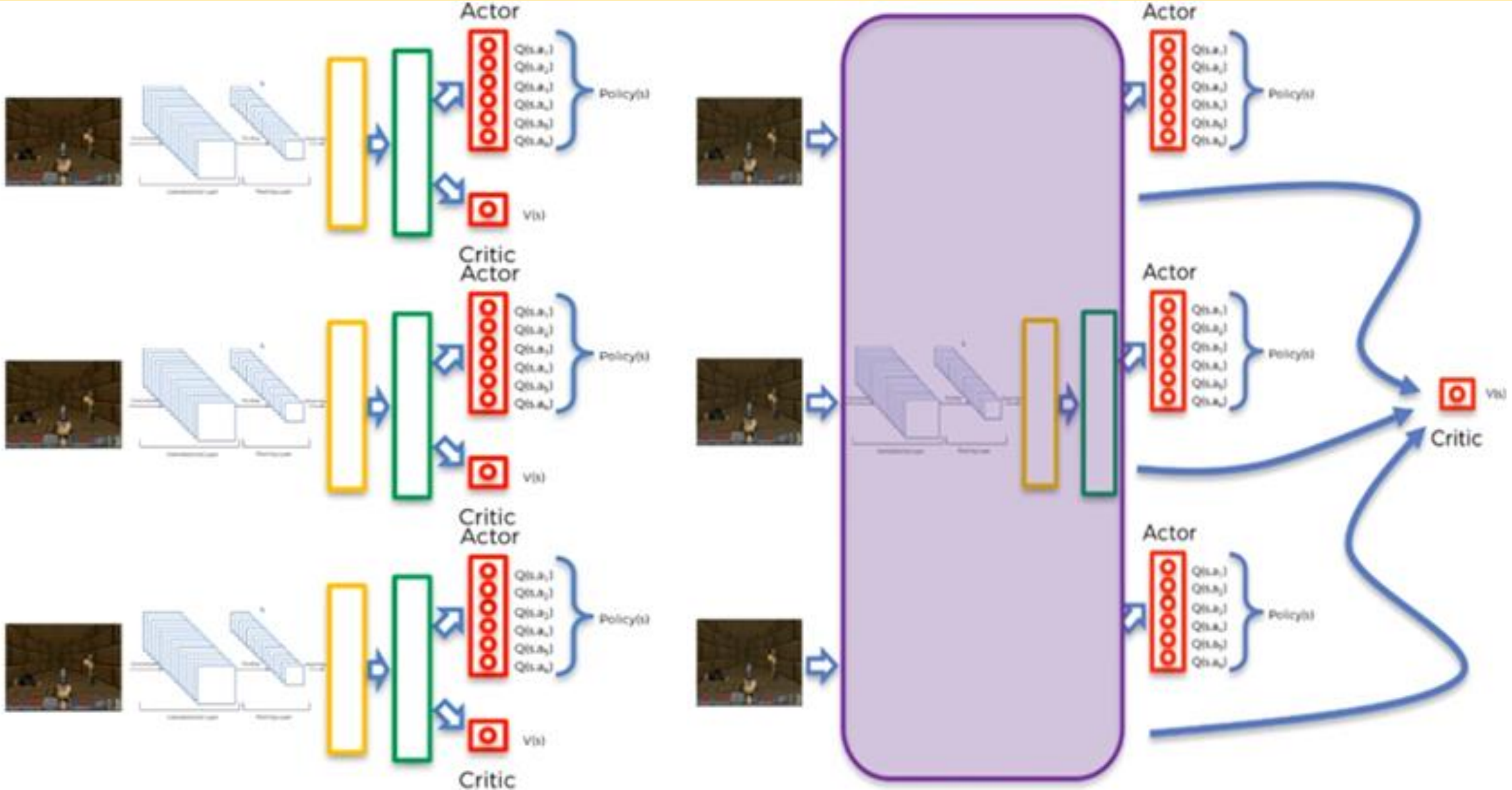
Actor outputs a set of action probabilities the agent can take (policy-based) Q(s,a)

Agent uses value estimate (the critic) to update policy (the actor) so that actor can output better actions that result in a higher reward

ACTOR-CRITIC



ASYNCHRONOUS



ADVANTAGE

Advantage - difference between q-value of an action taken and value of that state = $Q(s, a) — V(s)$

Advantage provides information on how much better is the q-value than the known value

High advantage - q-value is a lot better than known value. This behaviour is enforced and these actions repeated.

Low advantage - neural network tries to prevent these actions from occurring again

A3C ALGORITHM WITH LSTM

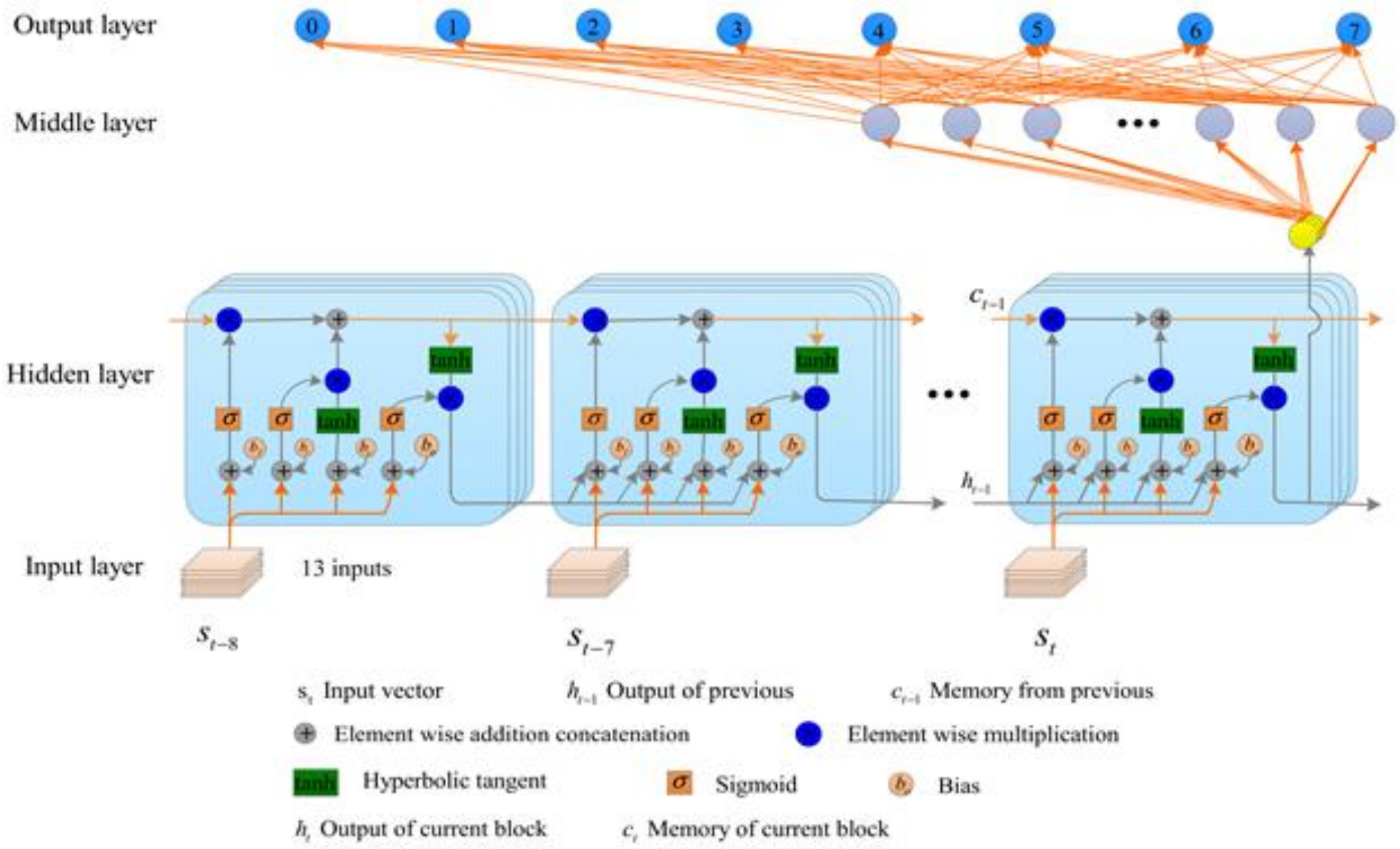
In dynamic environment, it is difficult to predict direction and motion of the obstacle

Network has to remember motion of moving obstacle from past frames, so that AUV can decide its own course to steer

Past frame memory is stored in LSTM which replaces hidden layer in fully connected layer

LSTM layer allows A3C algorithm to have memory which allows algorithm to remember what happened before

A3C ALGORITHM WITH LSTM



CONCLUSION

Selective overview of controlling AUVs with Deep Reinforcement learning

Conventional optimal control or adaptive control etc. have limitations for time varying systems like navigating an AUV

Non-linear system pose difficulties and are not robust for parameter variation or changes in the input form

Deep reinforcement learning are robust and well suited to controlling systems in dynamic non-linear environments

Their inherent parallel structure allows for very efficient processing

JAI HIND