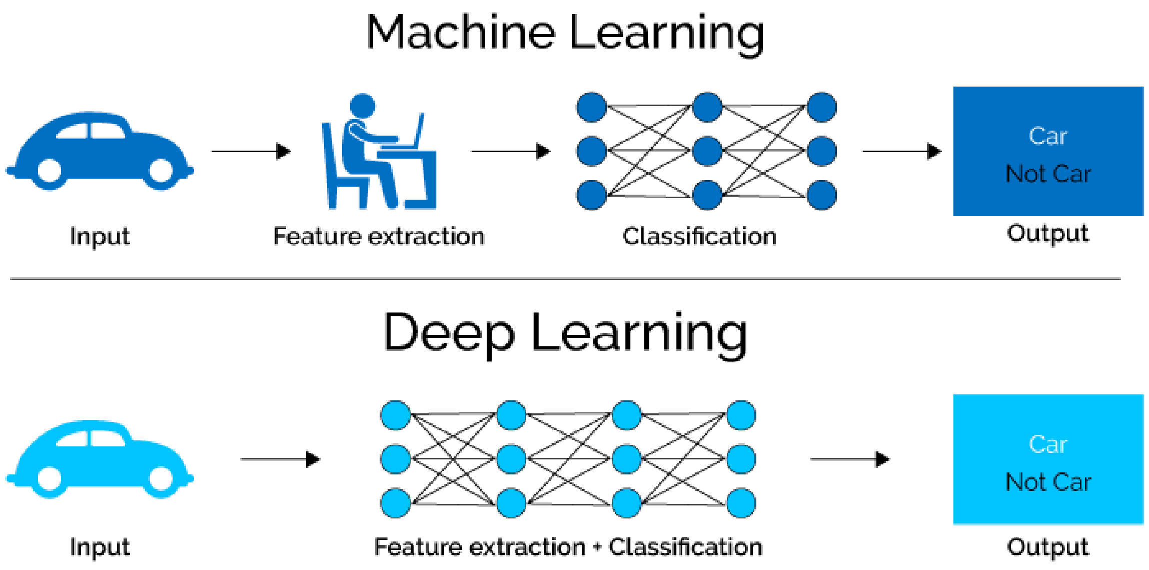
Control of Inverted Pendulum using DQN algorithm with ACTOR AND CRITIC NEURAL NETWORK Algorithm

Author - Kornia Bapari

Tools used - Matlab Deep learning Tool box, Brian Douglos youtube tutorials on Deep Learning

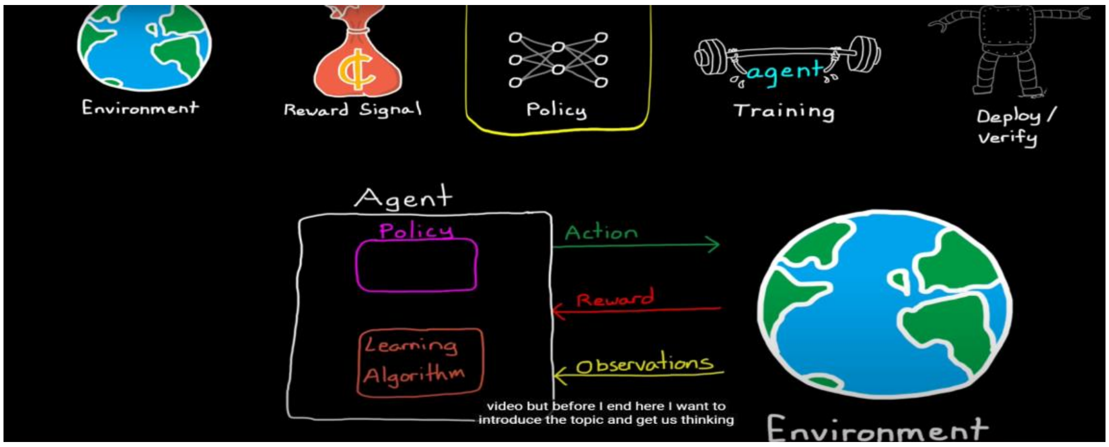


Chapter -1

Reinforcement learning Visualisation

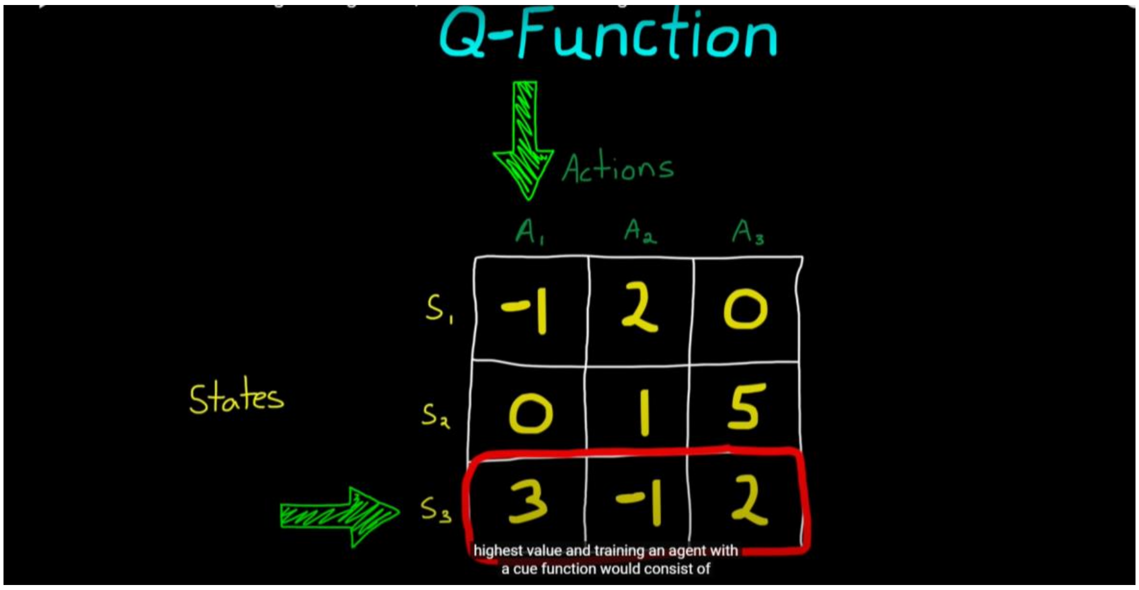
Reinforcement Learning is basically you have a AI algorithm that we call as Agent , this algo takes in reference , observations(states) and rewards and outputs actions

The hypothesis in ML is called policy here, where reinforcement learning comes from is that it changes the policy depending upon the observations and reward vs a constant hypothesis, so this can work on control problems and in dynamic environment



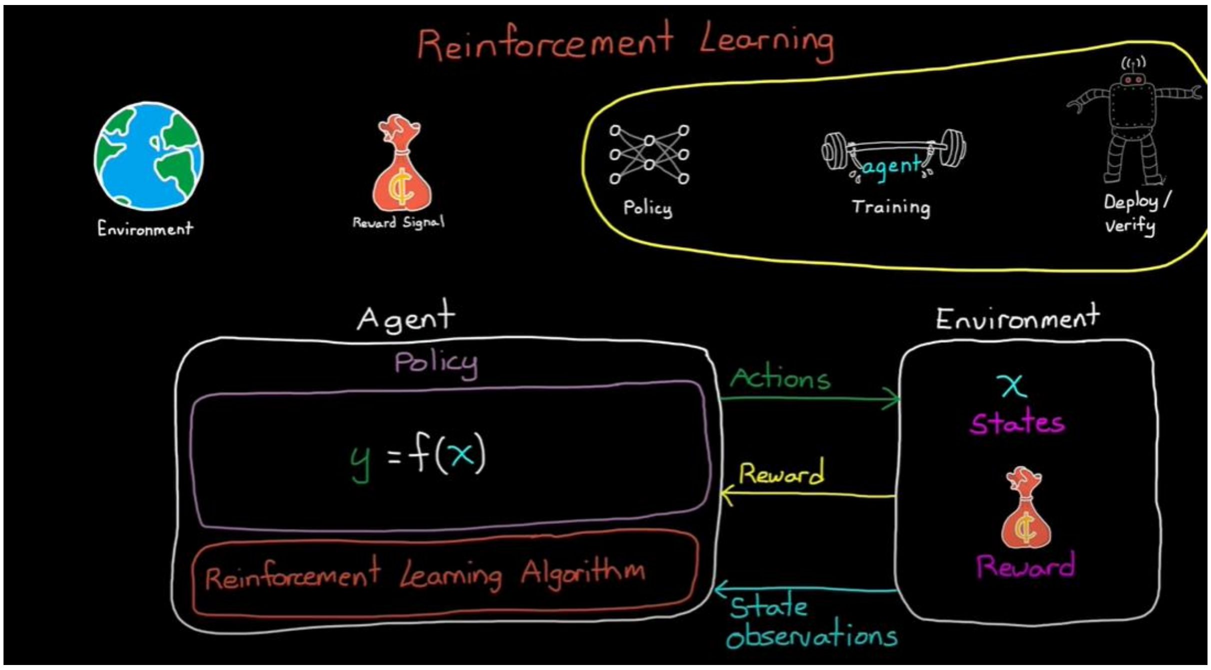
video but before I end here I want to introduce the topic and get us thinking

Now you can think of policy as data table like below

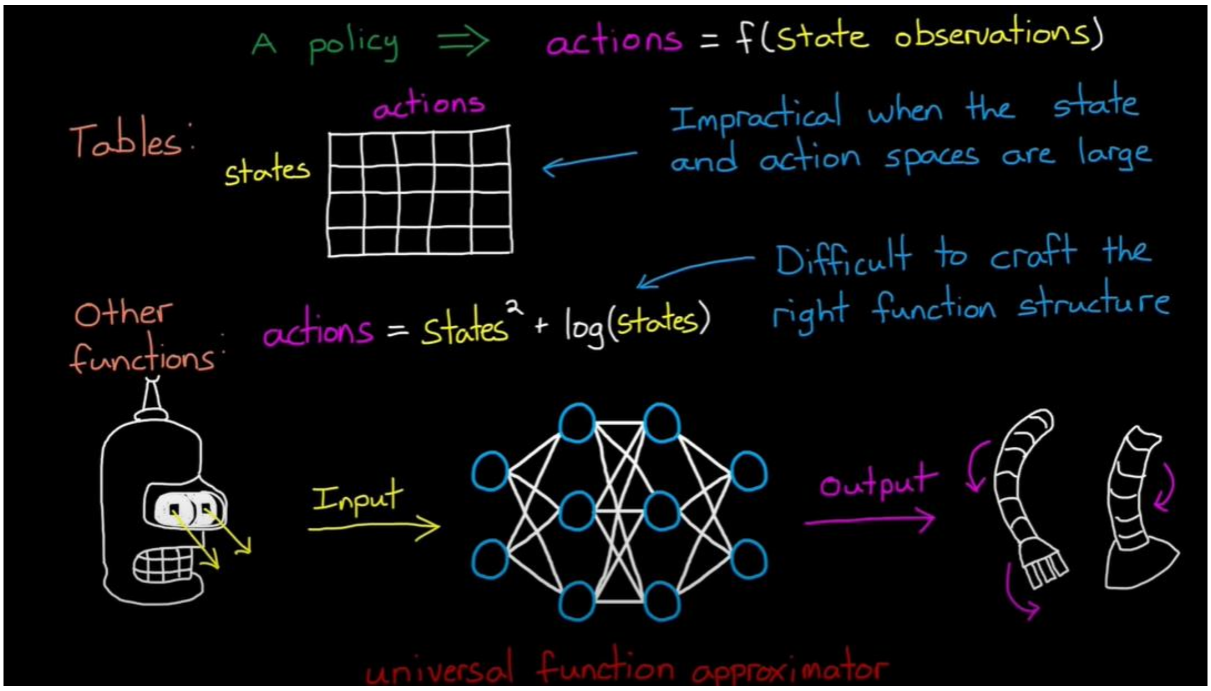

Q-Function
Actions
States

	A_1	A_2	A_3
S_1	-1	2	0
S_2	0	1	5
S_3	3	-1	2

highest value and training an agent with a cue function would consist of



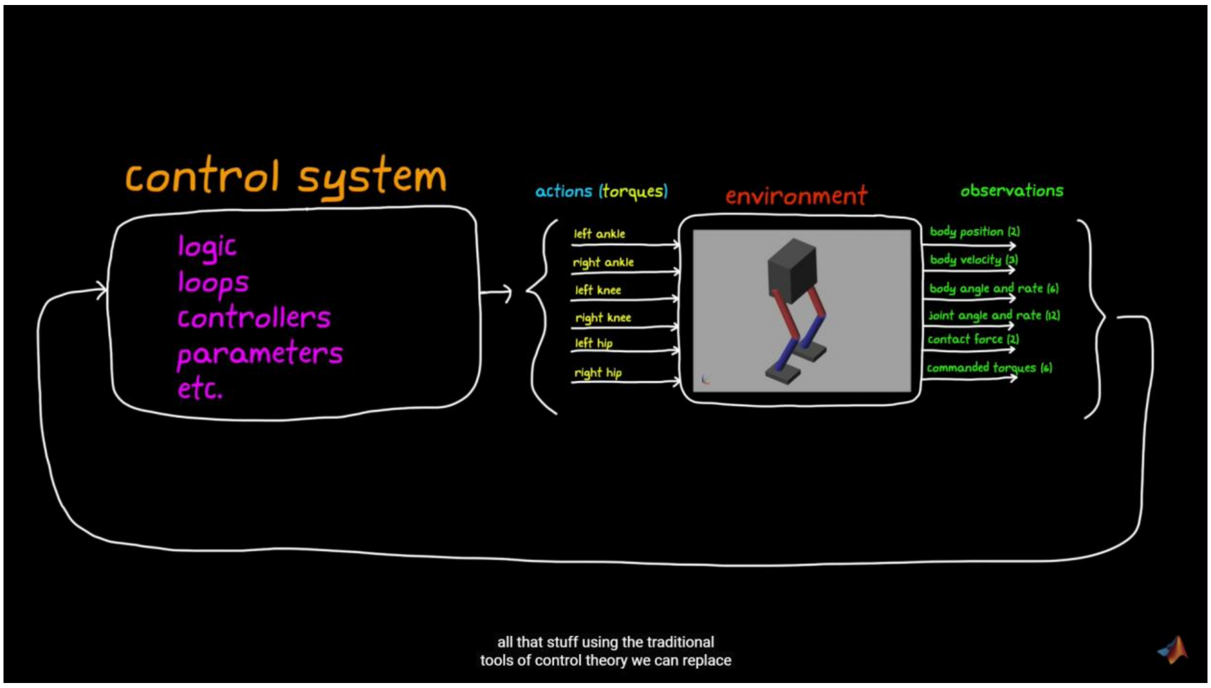
So we can use tables or functions as policies but drawback is that as your states and actions get large , its difficult to manage the tables , with function its difficult to craft them, with neural networks we can achieve both things, so you can take pixels as input as well as other states of robot as inputs to neural network and you can output actuator commands to robot.



A policy ⇒ actions = f(state observations)
Tables:
actions
states
Impractical when the state and action spaces are large
Other functions:
actions = states² + log(states)
Difficult to craft the right function structure
Input
output
universal function approximator

Example

Typical control problem

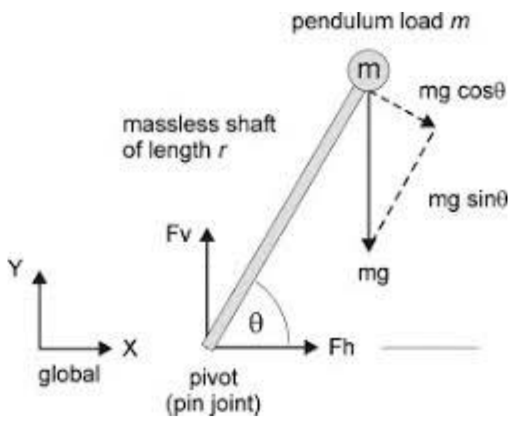


all that stuff using the traditional
tools of control theory we can replace

Chapter 2

Inverted Pendulum Problem Formulation-

Inverted Pendulum is control problem – the output we want is to supply torque to pendulum such that it stays upright



The angle theta, the body rate are states of the pendulum

and Torque command is actuator

Problem – we want the neural network to keep pendulum at zero angle

Penalisation to Algo

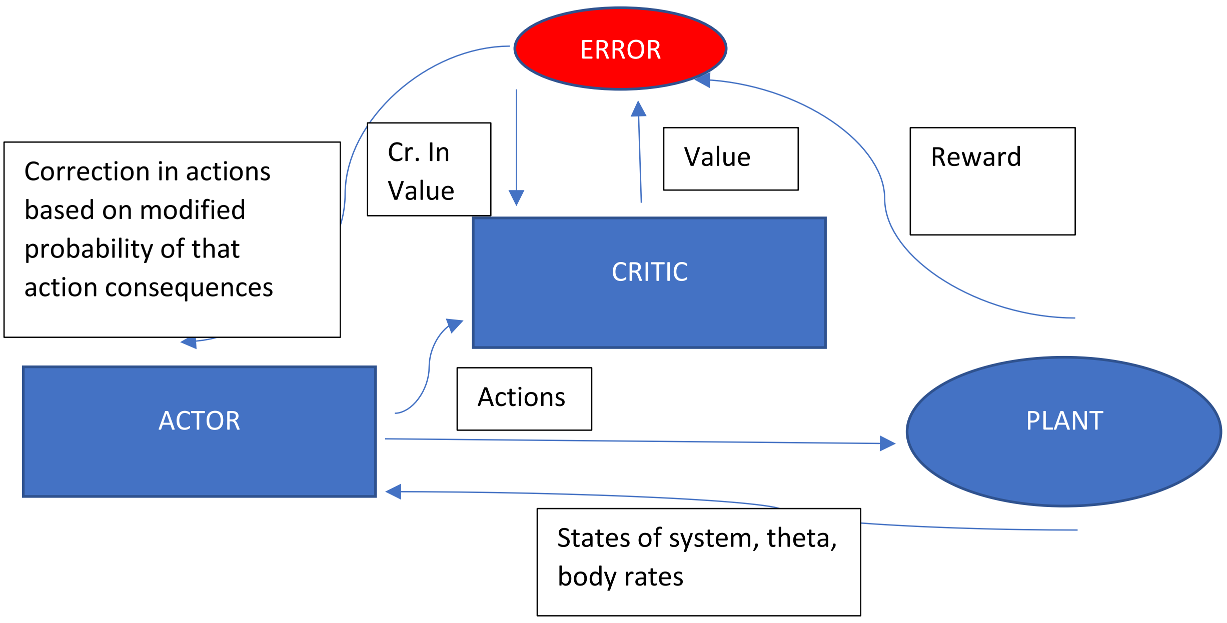
We will penalise the neural network if theta is large

If body rate is large

If actuation is large

So our reward function will be = r = -(theta + theta_dot + u)

ACTOR AND CRITIC ALGORITHM to create 2 neural networks for Pendulum Problem


ERROR
Correction in actions based on modified probability of that action consequences
Cr. In Value
Value
Reward
CRITIC
Actions
ACTOR
PLANT
States of system, theta, body rates

So We will use 2 neural networks structure, in which the actor network will take as input the states of system and will give actuator command , while this actuator command will be fed to plant as well as CRITIC neural structure, the critic will output value for that actuator command and will compare with the reward coming back from the plant as feedback and will modify this reward as value for future use, also based on the value it predicted minus the reward that actually happened, the error term will be feeded back to Actor to correct for probability consequences of that action.

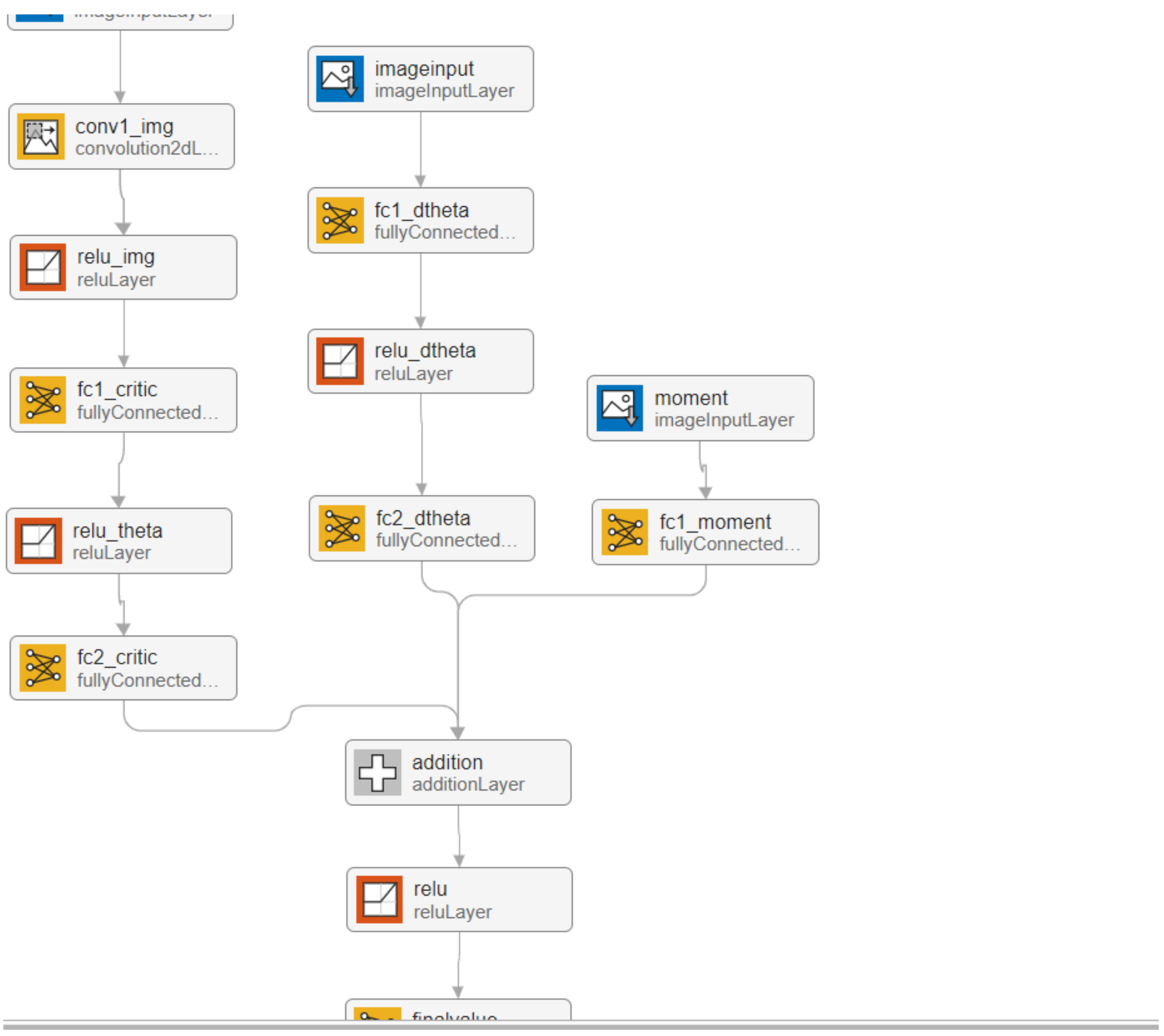
Thus Critic corrects for Actors parameters dynamically, and thus the reinforcement learning paves way to making adaptive control systems that will work with changing plant.

Chapter 3-

Code in MATLAB DEEP DESIGNER APP

The methodology developed in chapter 2 is used to create neural network on matlab designer app,

So you can see too main layers , input is the image of inverted pendulum



The designer app gives you option to auto generate the code in matlab , so I used the auto generate code with respect to network structure I created .