Chapter-7
Other Topics-Overview

Topics

- Ensemble learning vs
- Hybrid learning vs
- End-to-end learning
- Reinforcement learning

Ensemble learning

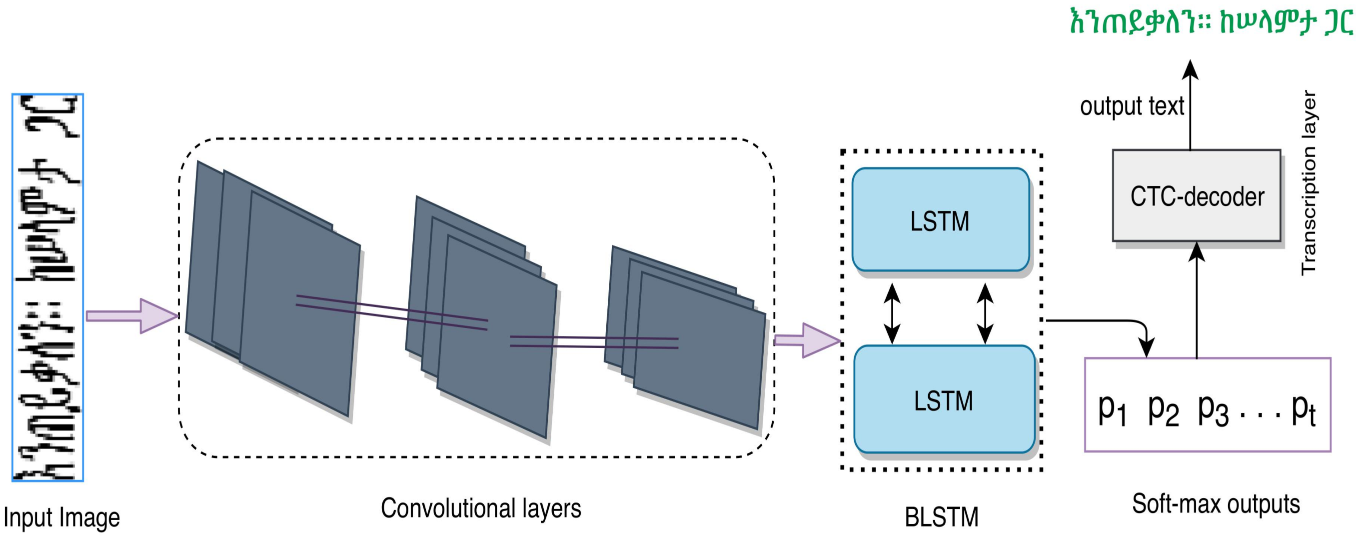
- **Ensemble learning** is the process by which **multiple models**, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem and to **improve model performance.**
  - **e.g AdaBoost:** algorithm for classification problems that add new machine learning models in a series where subsequent models attempt to fix the prediction errors made by prior models.
  - **Bagging:** Give equal weightage to all classifiers then majority voting
  - **Boosting:** Give weightage according to the accuracy of the classifier

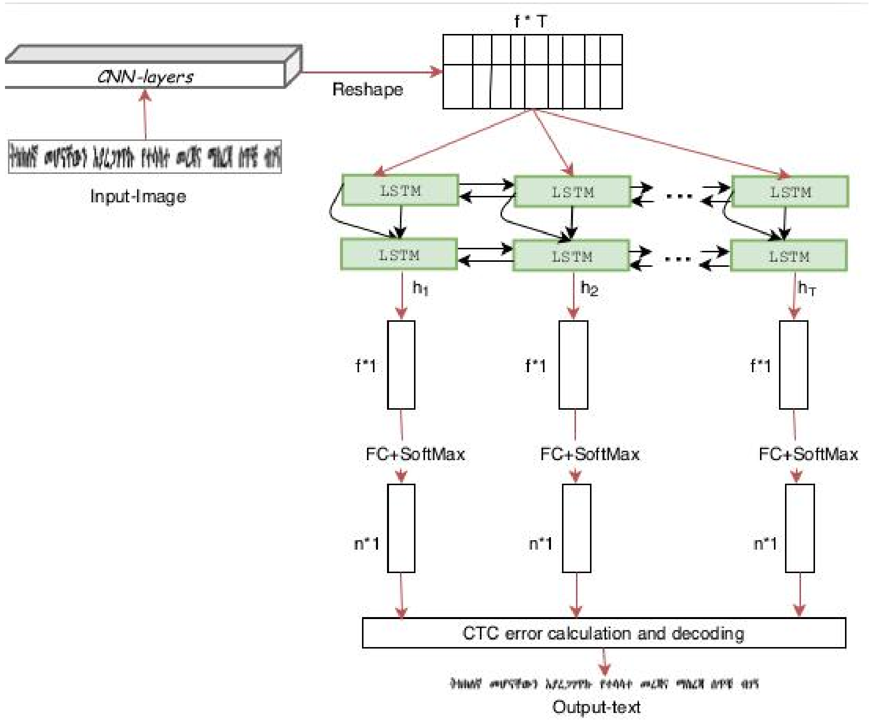
e.g a patient with a set of symptoms and taking opinion doctors

- **Random Forest:** A subset of input features is chosen to form each training set

Hybrid learning

- Combining two or more different machine learning techniques so as to build a model for solving a task.
  - **e.g** using one unsupervised learner (or cluster) to pre-process the training data and one supervised learner (or classifier) to learn.
  - Using one as feature extractor while the other as a sequence learner



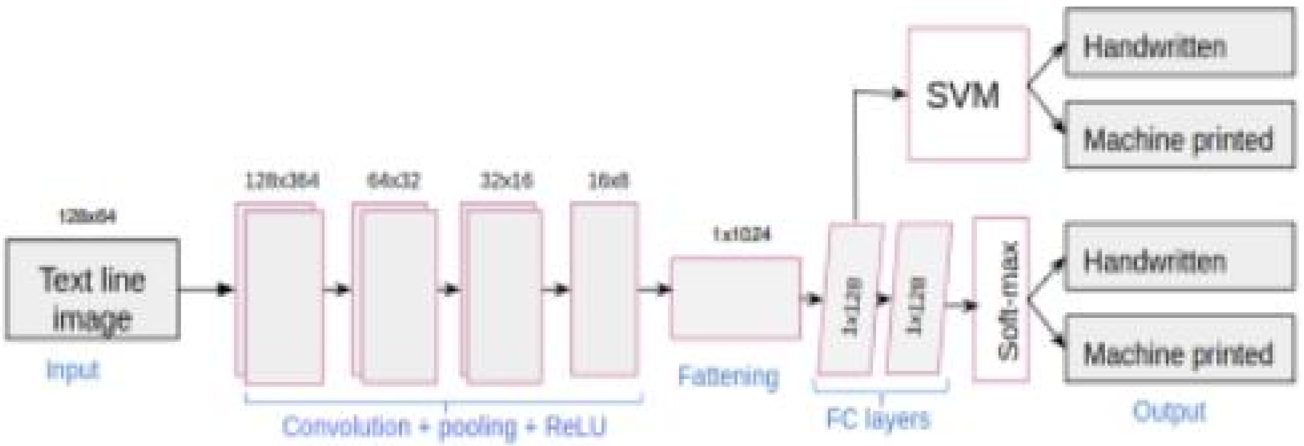


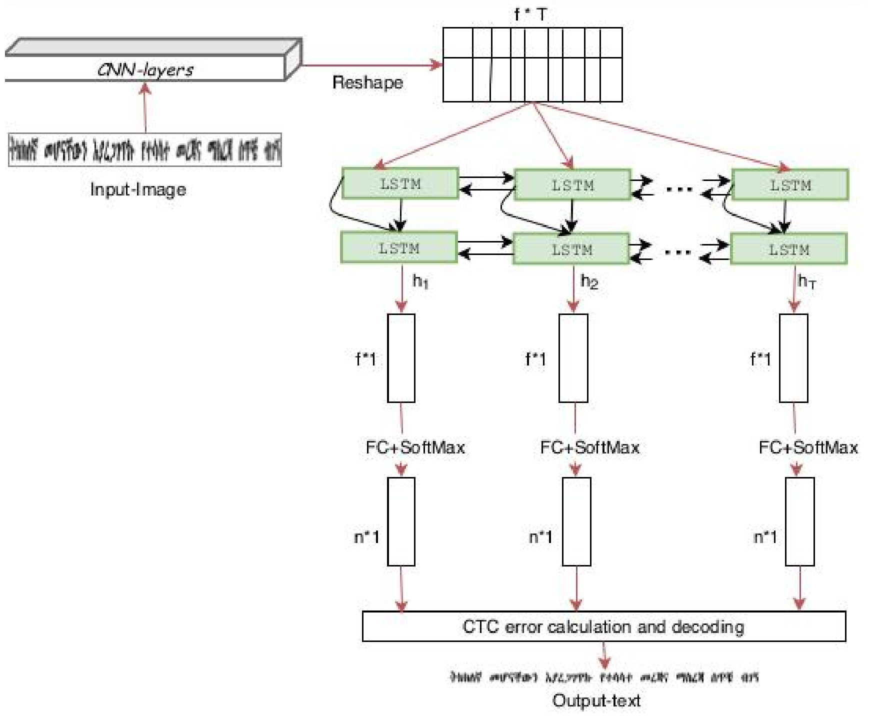
Are they the same?

- **Ensemble** methods work independently to vote on an outcome while **hybrid** methods work together to predict one single outcome, which no voting element present in it.

End-to- end learning

- CNN+SVM (hybrid but not end-to-end)
- CNN+LSTM+CTC (train end-to-end)



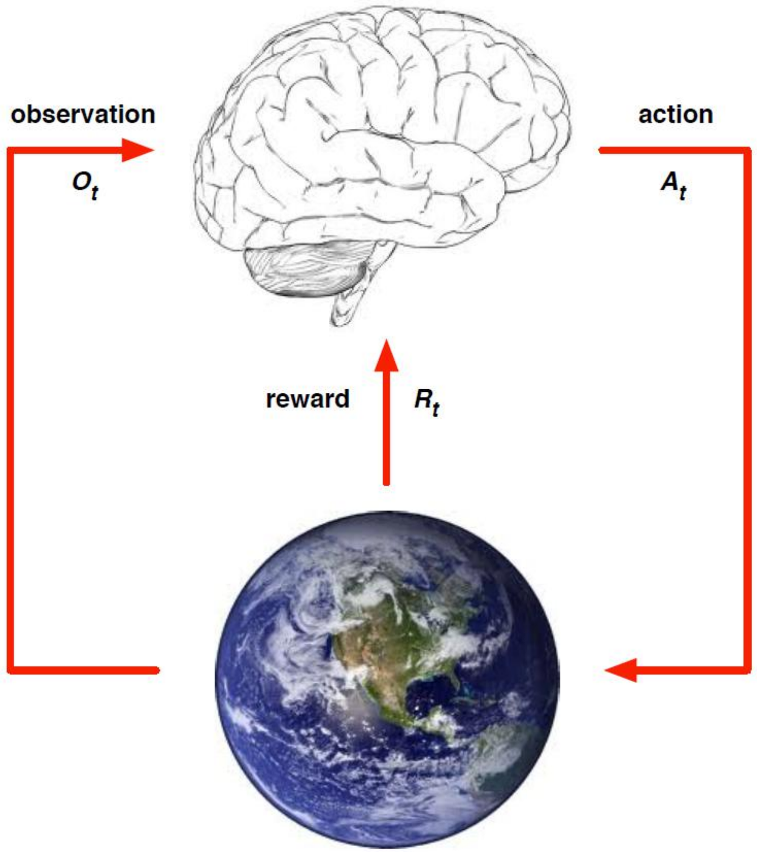


Reinforcement learning (RL)

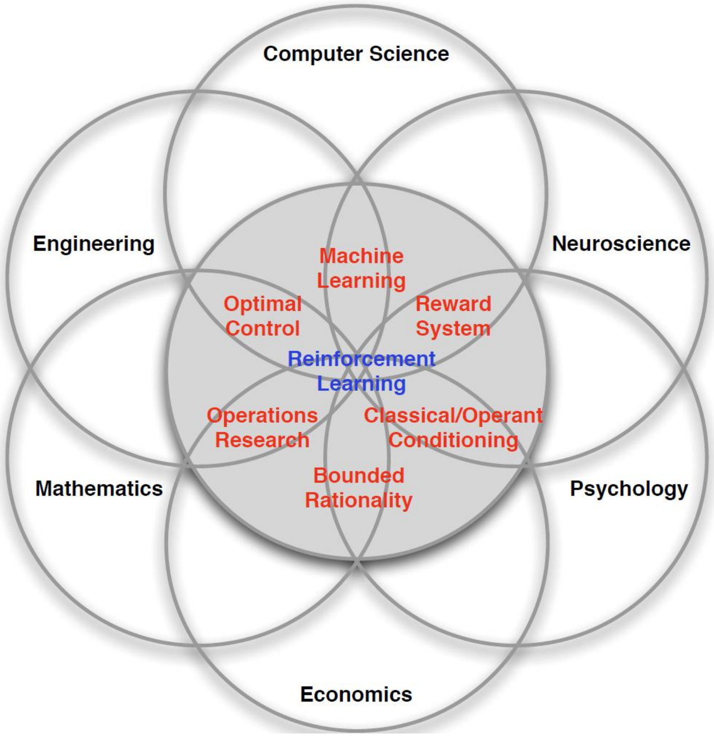
- Reinforcement learning (RL) is an area of ML concerned with how intelligent agents ought to **take actions** in an **environment** in order to maximize the notion of cumulative reward

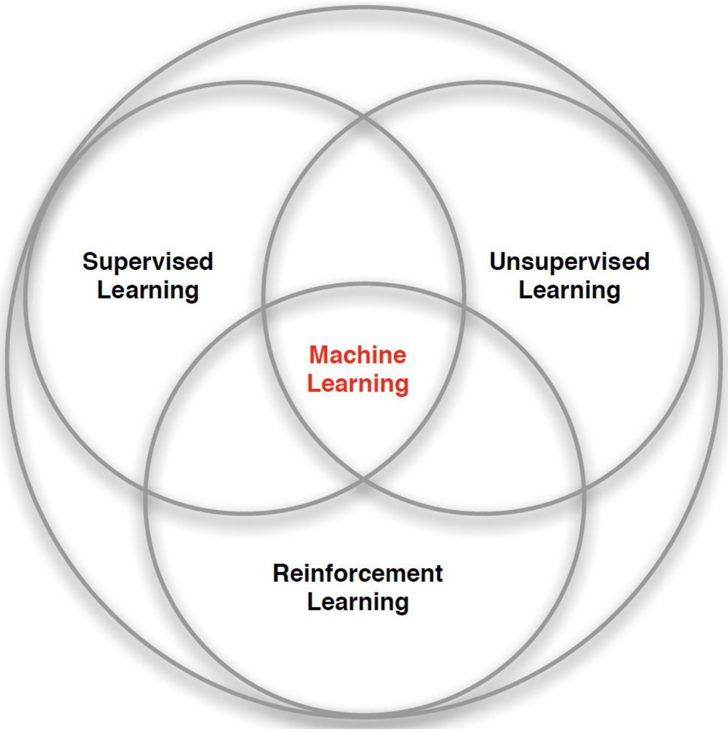
What is RL?

- Reward instead of data (no GT)
- Preferable states in the world are defined



Supervised or Unsupervised? Both!



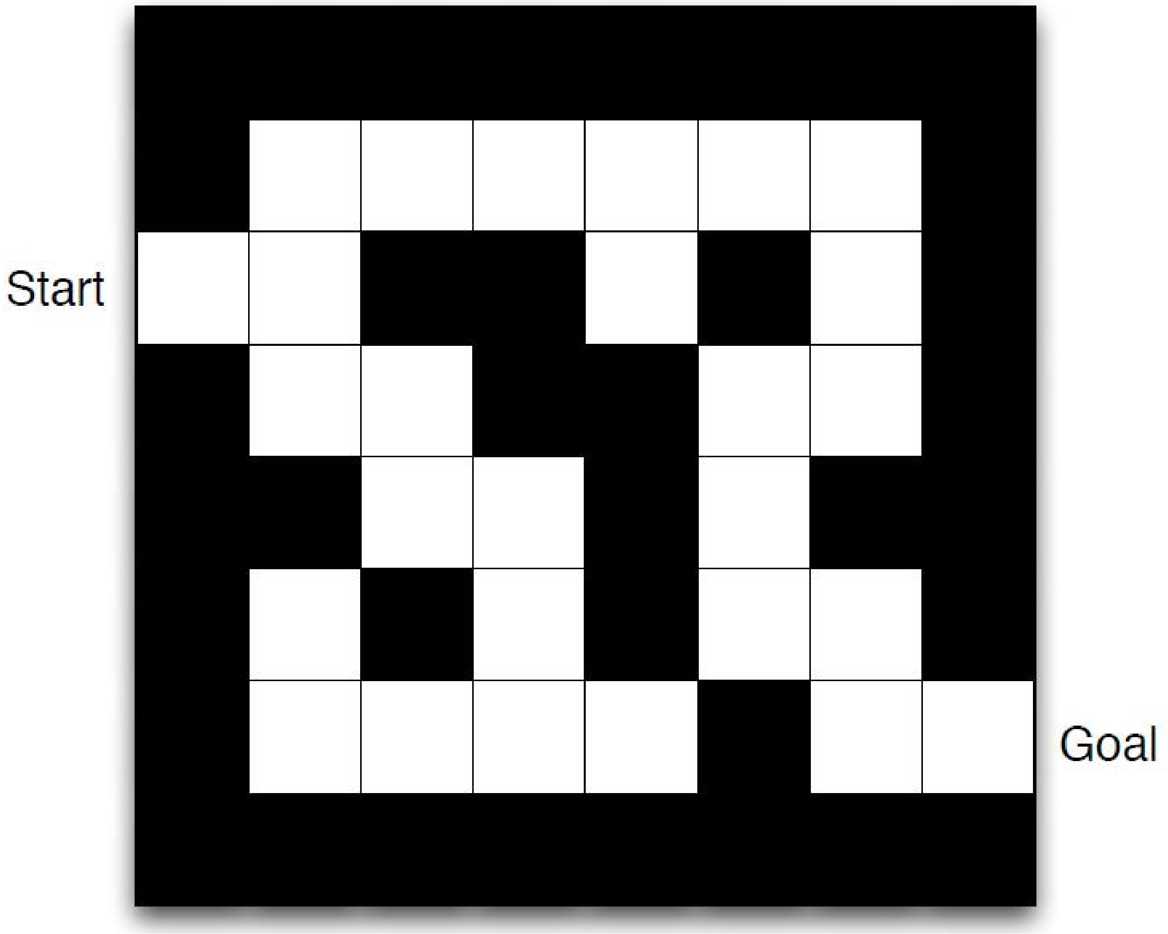


Settings of RL

- **Agent:** An agent takes actions; for example, a drone making a delivery
- **Action (A)**: A={a_1,a_2,...a_m} is the set of all possible moves the agent can make: e.g running right or left, jumping high or low, buying or selling
- **Environment:** The world through which the agent moves, and which responds to the agent.
  - The environment takes the agent's current state and action as input, and returns as output the agent's reward and its next state.
  - e.g If you are the agent, the environment could be the **laws of physics** and the **rules of society** that process your actions and determine the consequences of them.
- **State (S)**: A state is an immediate situation in which the agent finds itself in relation to other significant things such as tools, obstacles, enemies or prizes
- **Reward (R):** A reward is the feedback by which we measure the success or failure of an agent's actions in a given state
- **Policy (π(s))**: is the strategy that the agent employs to determine the next action based on the current state.
- **Value (Vπ(s))**: is defined as the expected long-term return of the current state under policy π
- **Model:**
  - **predict next state $P^a_{ss'}$**
  - **predict the next reward P^a_s**

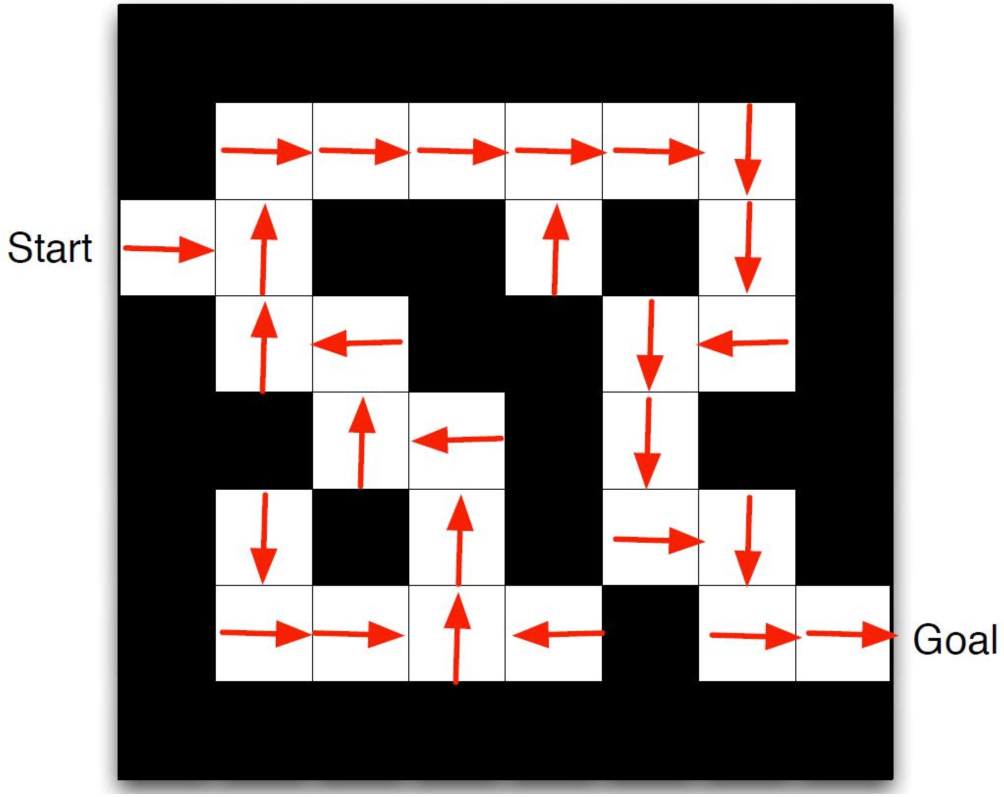
Example

- Reward: -1 per time step
- States: Agent's location
- Actions: N,E,S,W



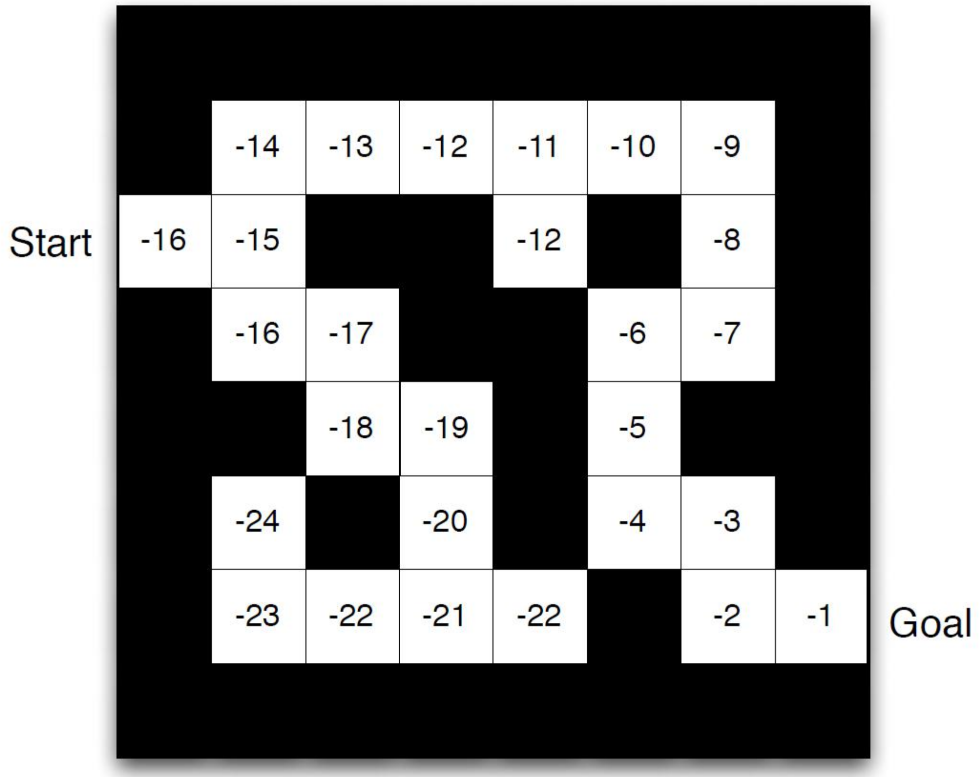
Policy

- Arrows represent policy $\pi(s)$ for each state s



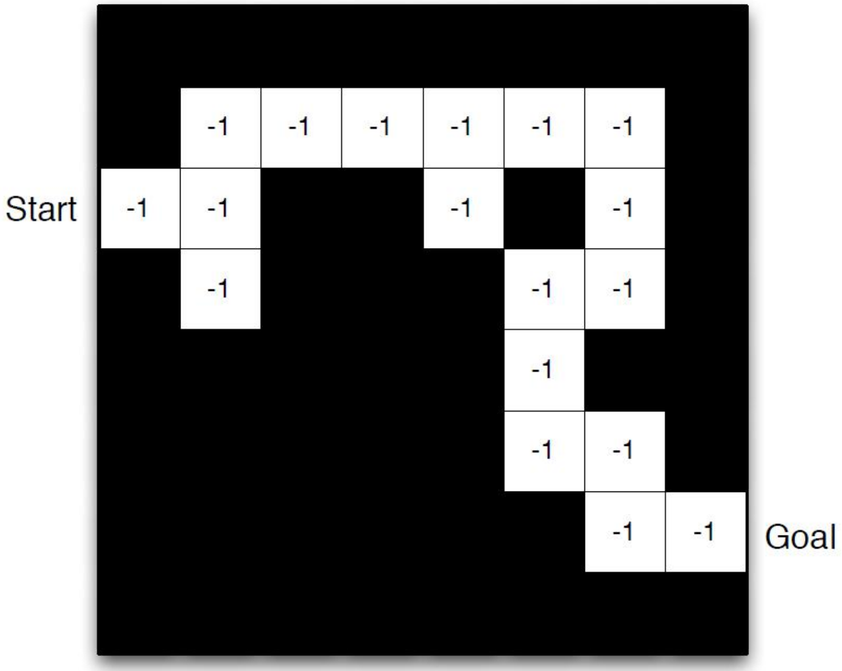
Value Function

- Numbers represent value $V_\pi(s)$ for each state s



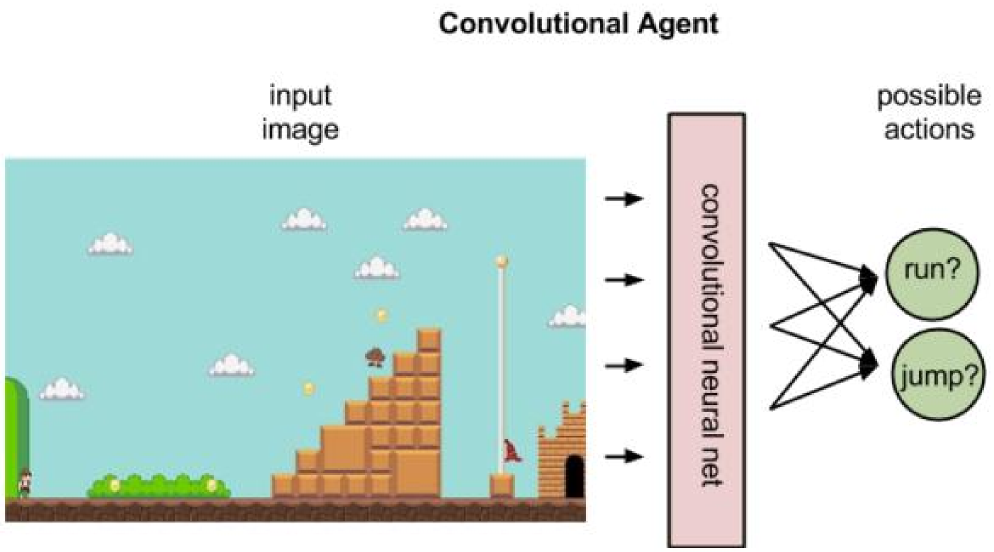
Model

- Grid layout represents transition model $P_{ss'}^{a}$
- Numbers represent immediate reward R_s^a from each state s (same for all a)



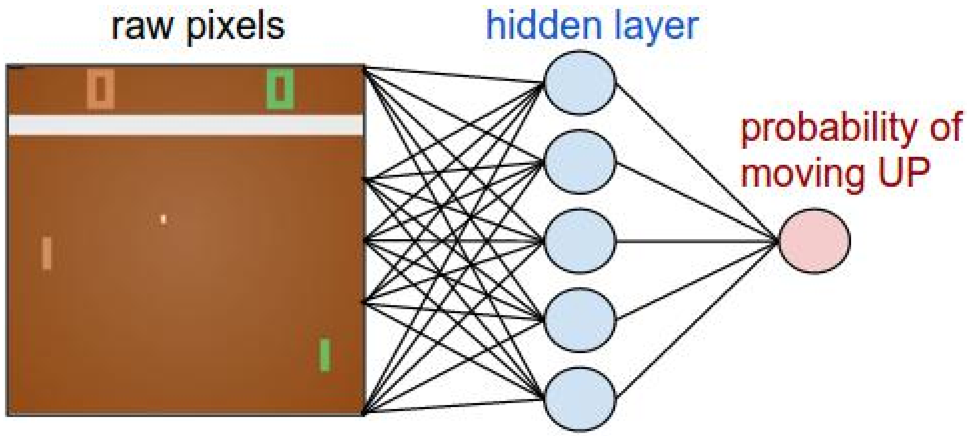
Example...

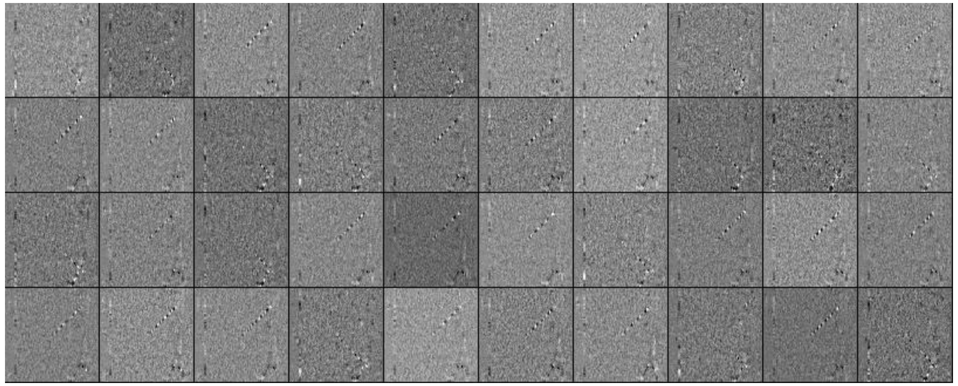
- Given an image that represents a state, a convolutional net can rank the actions possible to perform in that state
- The below image illustrates what a policy agent does, mapping a state to the best action.
  $a = \pi(s)$
- A policy maps a state to an action.

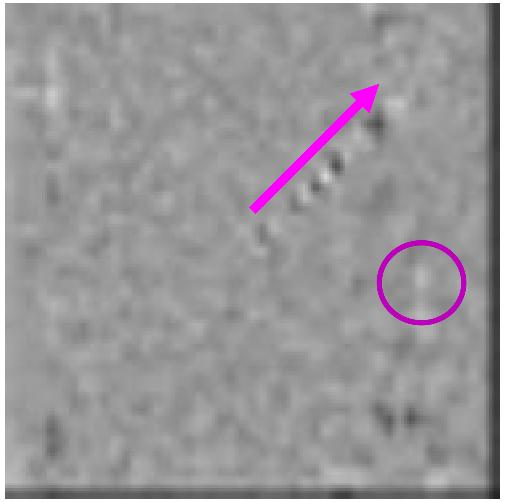


Neural Network as Optimal Policy Estimator

https://www.youtube.com/watch?time_continue=77&v=YOW8m2YGtRg



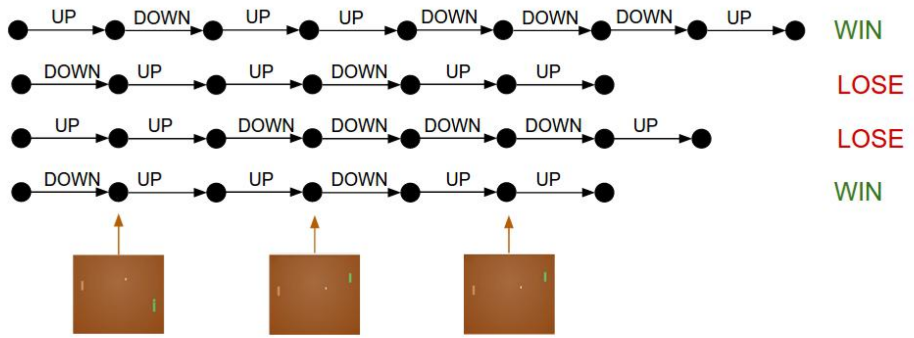


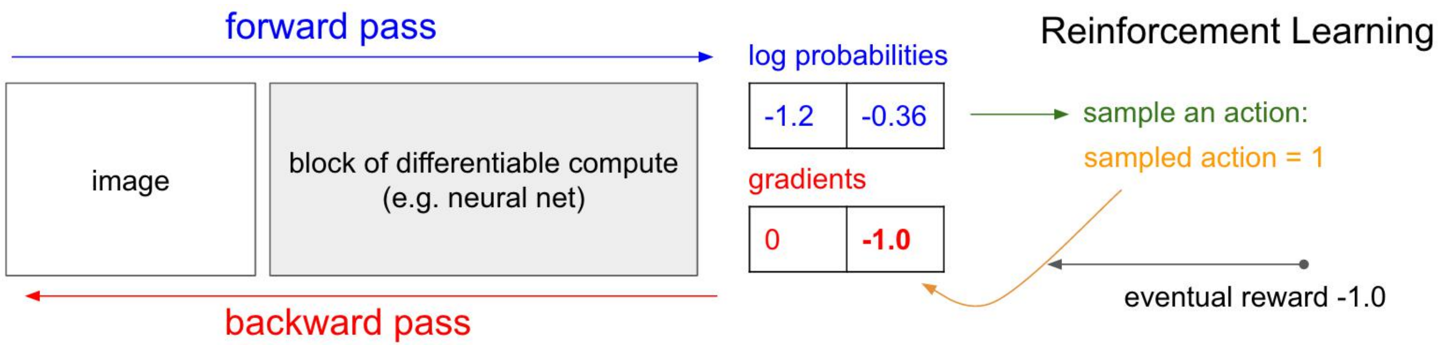


-->a pong game that simulates table tennis
(210*160*3) input image

How to Train such a Network ?

- "generate a dataset" to then perform supervised learning on-the-fly





Characteristics of RL

- **Input:** The input should be an initial state from which the model will start
- **Output:** There are many possible output as there are variety of solution to a particular problem
- **Training**: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.
- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Agent's actions affect the subsequent data it receives

Applications of RL

- Robotics for Industrial Operations
- Supply Chain & Logistics
- Traffic Control
- Bidding & Advertising
- Recommender Systems
- Load Balancing

Further Readings

- Deep Q-Learning
- Semi-supervised SVM (S3VM)
- Policy vs value vs policy+value-based learning
- Bandit problems and online learning

- Sources for further reading on RL
  - [Udacity (Georgia Tech.)] CS7642 Reinforcement Learning
  - [Stanford] CS229 Machine Learning - Lecture 16: Reinforcement Learning by Andrew Ng
  - [UC Berkeley] Deep RL Bootcamp
  - [UC Berkeley] CS294 Deep Reinforcement Learning by John Schulman and Pieter Abbeel
  - [CMU] 10703: Deep Reinforcement Learning and Control, Spring 2017
  - [MIT] 6.S094: Deep Learning for Self-Driving Cars
  - Lecture 2: Deep Reinforcement Learning for Motion Planning

or just go to this link https://wiki.pathmind.com/deep-reinforcement-learning

Information

Chapter 1: 8%

Chapter 2: 25%

chapter 3: 10%

chapter 4:   25%

chapter 5: 5%

chapter 6: 15%

chapter 7: 2%

General= 10%

5-10 questions...   you have already done 20% of the questions

Exam-date: TBA

Tips

Demo on overleaf

- https://www.overleaf.com/project

sample exam questions