# Progress Report
# *Space Invader AI*

Cédric Lecuyer 49002742
Electrical Engineering

## Summary

- Tools
  - Environment
  - OpenAI GYM
- My current project
  - How it works
  - My current results
- My future plan
- References

# Tools

## Tools

- Anaconda Environment
  - Enables to have multiple python environment (such as virualenv)
  - Create my environment with tensorflow, keras and OpenAI GYM
- Keras
  - High-level API for neural networks
  - Working using Tensorflow (can work with CNTK or Theano)

# Keras Example

```
from keras import models
from keras.layers import Dense
from keras.optimizers import Adam

model = models.Sequential()
model.add(Dense(NB_NODE_PER_HIDDEN_LAYER,
input_dim=inputNodeNumber, activation='relu'))
model.add(Dense(NB_NODE_PER_HIDDEN_LAYER, activation='relu'))
model.add(Dense(outputNodeNumber, activation='linear'))
model.compile(loss='mse', optimizer=Adam(lr=learningRate))
```

# OpenAI GYM

OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms

# GYM

- Provides an access and some tools to interact with a set of environments
  - Atari environments
  - Board games
  - Classic control

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render() #Start a video of the game
        action = env.action_space.sample() #Choose
randomly
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {}
timesteps".format(t+1))
            break
```

Example of Gym main functions extracted from [2]

# My current advancement

# Reinforcement Learning

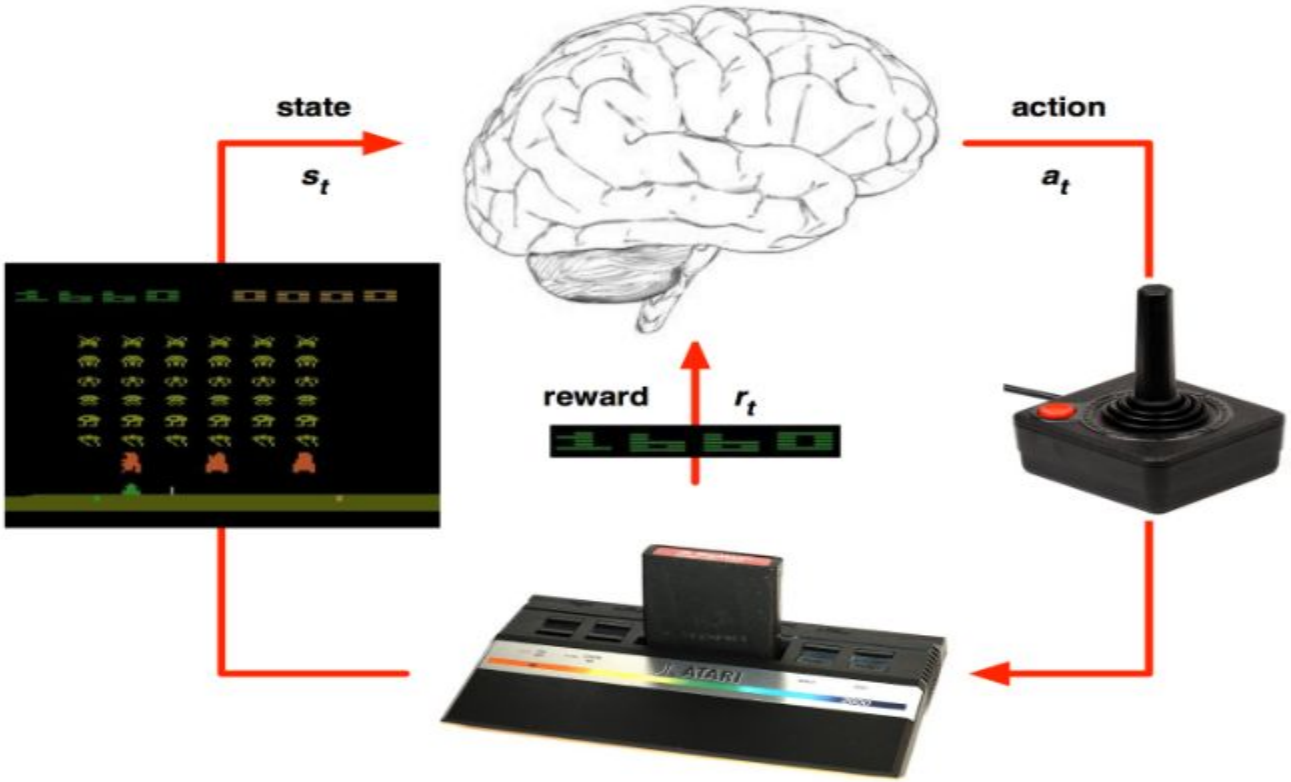
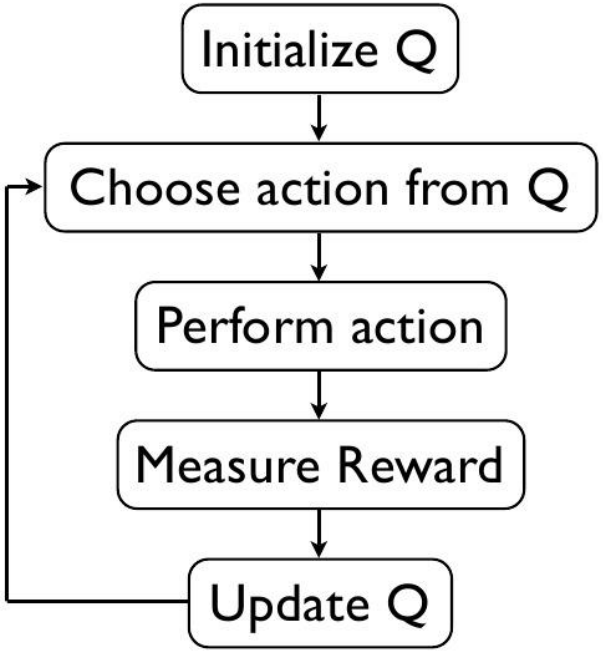



Diagram for agent with environment, extracted from [2]

## Q-Learning




Drawing for simple description of Q-learning, extracted from [3]

## Deep Q Learning

- Use neural network to calculate Q value of each action
- Our model chose higher Q value to choose the best action
- Use observation as input

# Neural Network Architecture

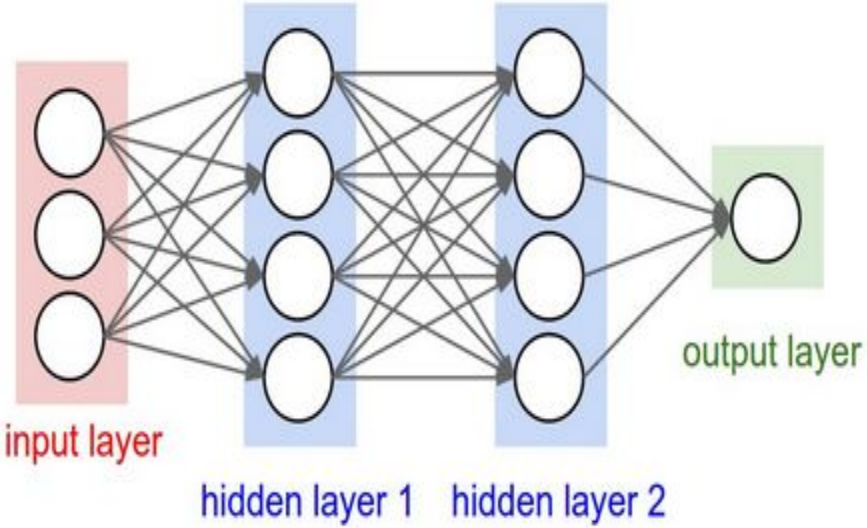


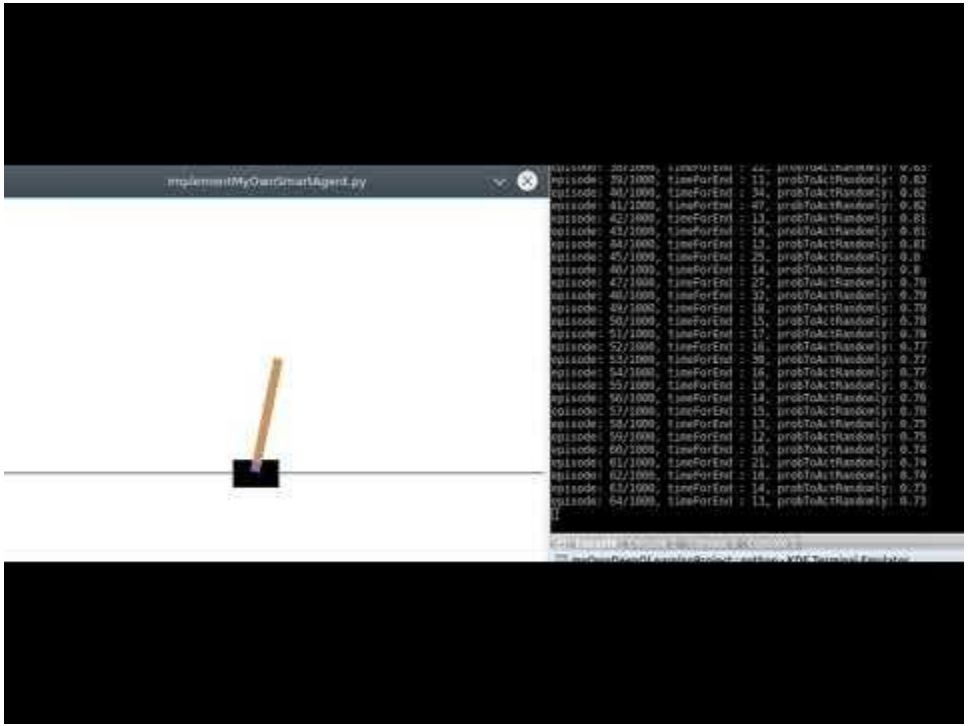

Diagram for neural network, extracted from [2]

- Input
  - Observation (Number of features)
- Hidden layer
  - Three layers
  - Fully-connected
  - 24 nodes in each layer
- Output :
  - Action (2 in my case)

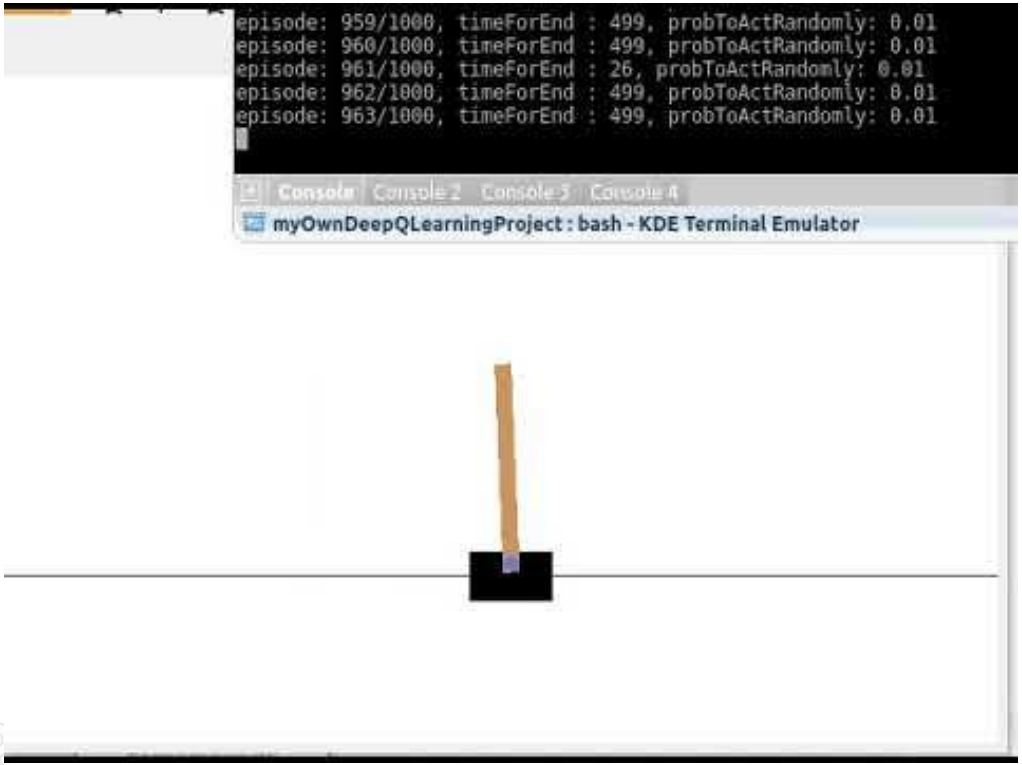
# Algorithm

```
Create Neural Network
Launch environment (Here game Cart-Pole)
For Episode Number :
    Launch one game session :
        Agent play (randomly or model predict)
        Agent remembers
    End of this game session
    If agent has played enough game
        Agent train with remember (Optimize best
        action)
end For
```

# Results
## Begin of agent



# Results
# End of Agent



## Failure

- Work only for this game
  - Not with other game even with two actions
    - Agent does not seem to learn
  - Don’t work cause action, observation are given in a different form

# My future advancement

## Future advancements

- Improve results
  - Train faster
  - Read other paper to optimize it
  - Look at other implementation (Lot of open code in OpenAI)
- Apply it to more complex Game (Space Invader or Breakout Game) =>

**Main objective**

# Figures

## Figures

[1] : “Documentation OpenAI GYM”, Online, available at https://gym.openai.com/docs/, Accesed at November 14 2017

[2] : “Deep-q-learning”, by Keon, February 6 2017,Online, available at https://keon.io/deep-q-learning, Accessed at November 14 2017

[3] : “Reinforcement Learning - Beginner Collection”, by Rubedo, 10 July 2017, Online, available at www.rubedo.com.br/2017/07/reinforcement-learning-beginner.html, Accessed at 16 October 2017

# References

# References

## Tools :

https://gym.openai.com/docs/

https://keras.io/

## Git :

https://github.com/rlcode/reinforcement-learning/blob/master/2-cartpole/1-dqn/cartpole_dqn.py

https://github.com/keon/deep-q-learning

## Doc :

https://www.intelnervana.com/demystifying-deep-reinforcement-learning/

## Video Link

Before

https://www.youtube.com/watch?v=D-5-CPk0d7w

End

https://youtu.be/Ec5gZGOYL5A?t=6m1s