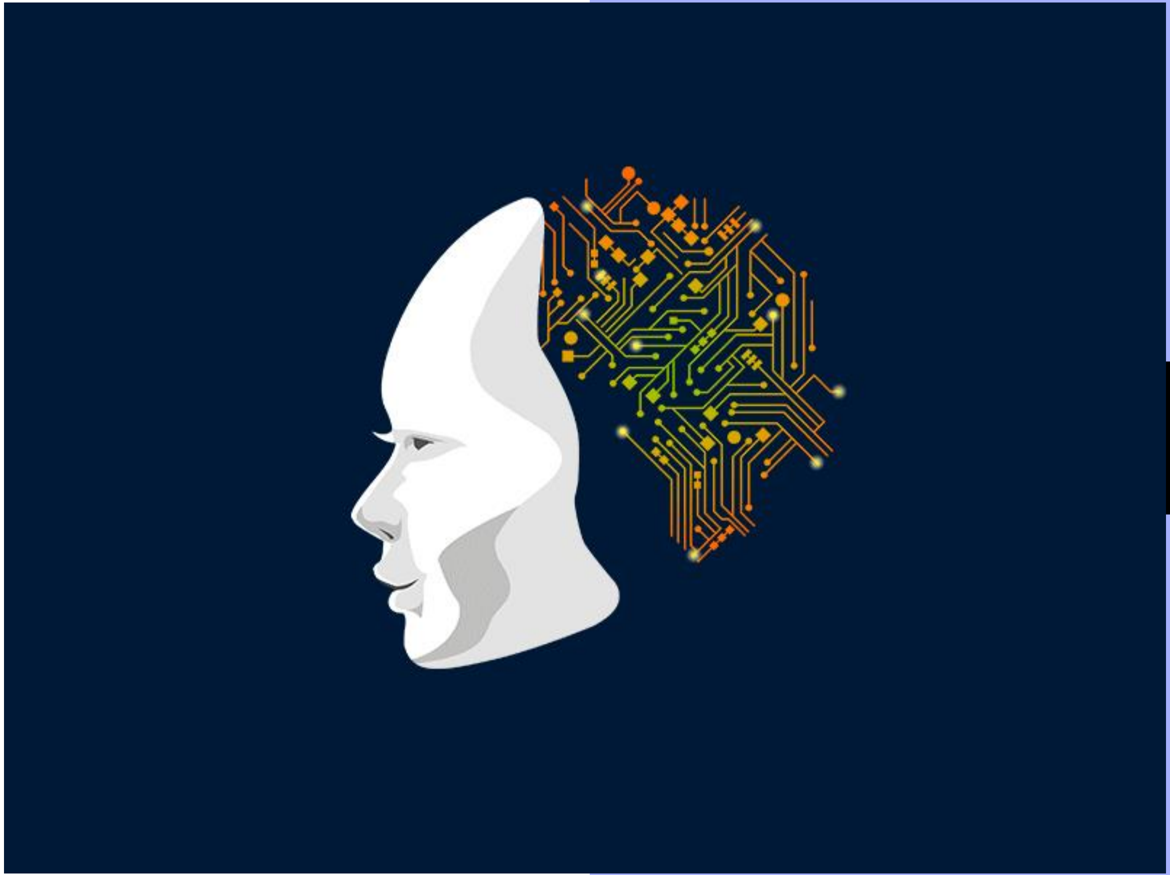
Deep Reinforcement Learning for Robotic Systems

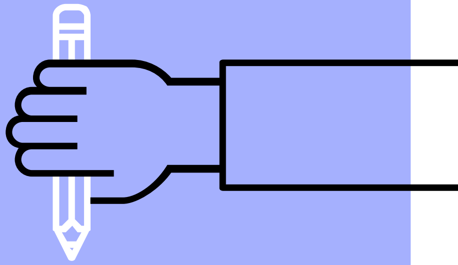


Mughees Asif

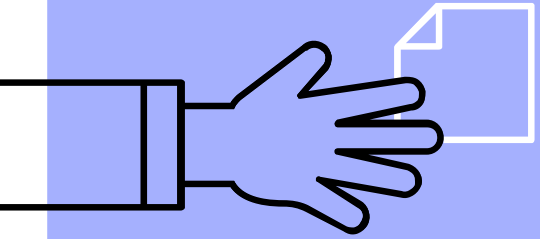
180288337

DEN331: Project Presentation

Outline

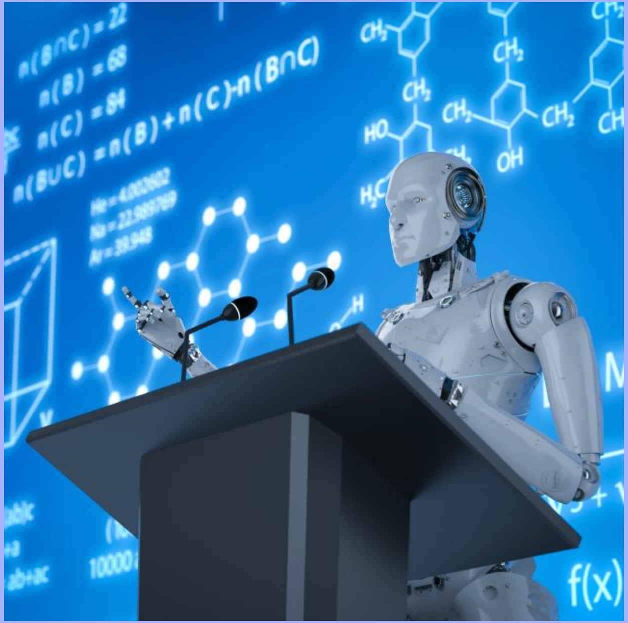


1. Artificial Intelligence (**AI**)
2. Deep Reinforcement Learning (**DRL**)
3. Proximal Policy Optimisation (**PPO**)
4. Current progress and working example
5. Summary



Artificial Intelligence (AI) Overview

- The capability of a machine to imitate intelligent human behaviour[1].
- **270%** increase in the use of AI algorithms in the past 4 years[2].
- Revenue projected to hit **£100 billion** by 2025[2].
- Use cases involve modelling customer behaviour, streamlining repetitive tasks, and enabling predictive analysis.



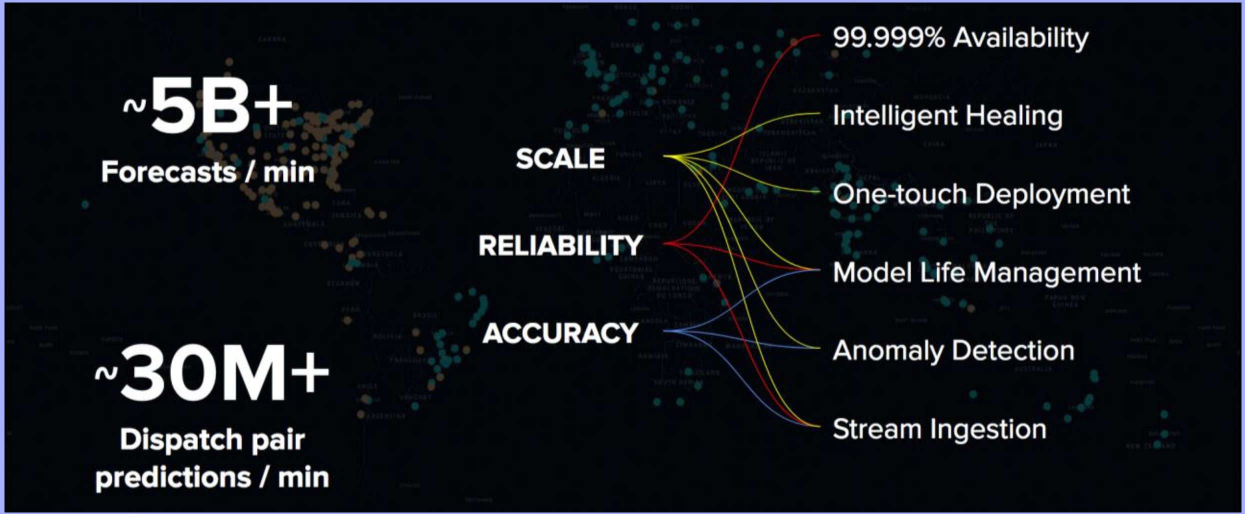
Artificial Intelligence (AI) Example

- **Miso Robotics**: Flippy.
- A fully autonomous robotic kitchen assistant that uses cloud-based monitoring, thermal imaging and deep learning.
- Improves cooking performance by studying the external environment and food temperature.

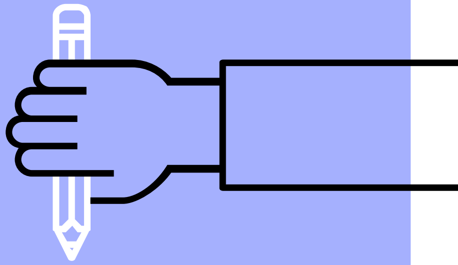
Artificial Intelligence (AI) Example

- **Uber:** Michelangelo real-time machine learning system.
- Use cases involve efficient ride-sharing marketplace, identify suspicious or fraudulent accounts, and suggest optimal pickup and drop-off points.

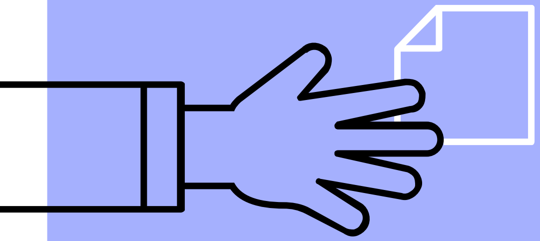
[4]



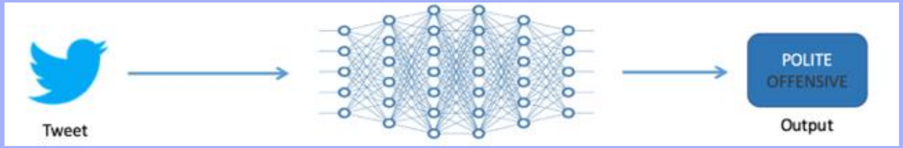
Outline



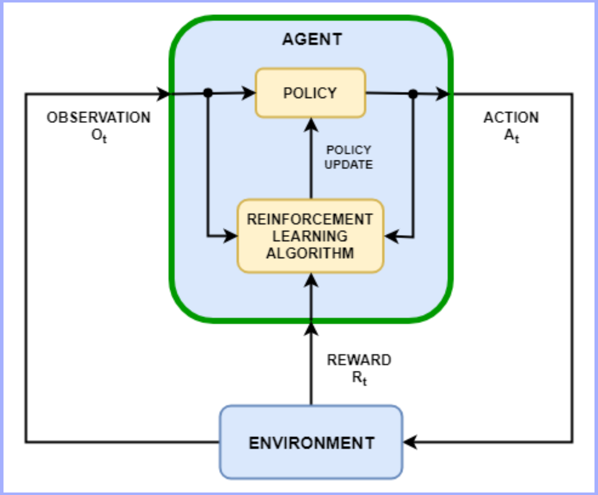
1. Artificial Intelligence (**AI**)
2. Deep Reinforcement Learning (**DRL**)
3. Proximal Policy Optimisation (**PPO**)
4. Current progress and working example
5. Summary



Deep Reinforcement Learning (DRL)



5



6

- **Deep Learning**: Abstraction and extraction of pertinent features from a given data set.
- **Reinforcement Learning**: A computational learning model using sequential trial and error.

Drawbacks

Neuroplasticity

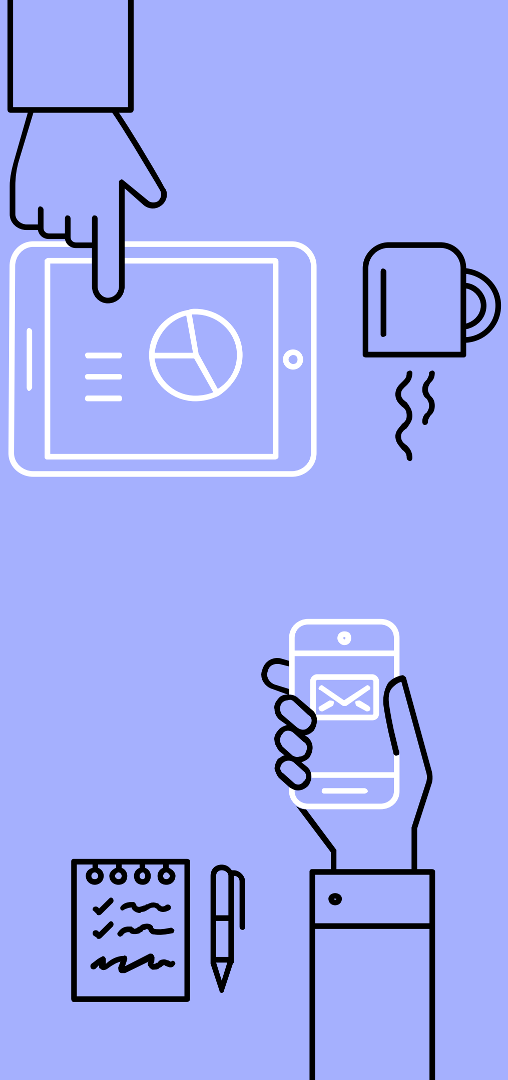
Lack of neuroplasticity decreases the performance, as the complexity is increased.

Overtraining

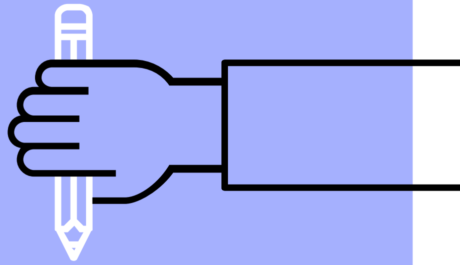
Model can predict training examples with very high accuracy, but can not transfer results to new data, leading to poor performance.

Sample efficiency

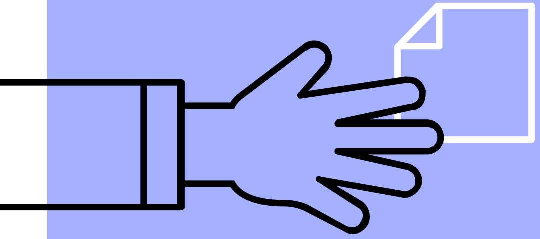
Amount of data needed for training, in order to reach a certain level of performance.



Outline

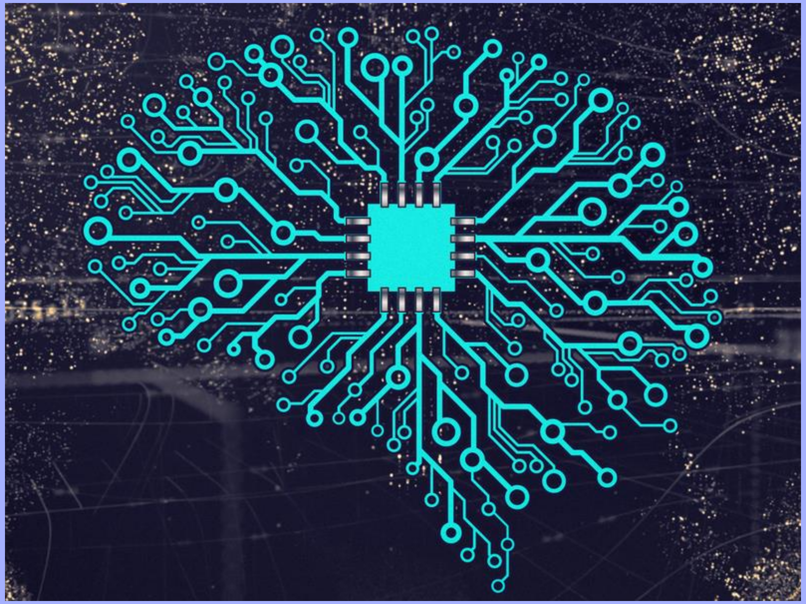


1. Artificial Intelligence (**AI**)
2. Deep Reinforcement Learning (**DRL**)
3. Proximal Policy Optimisation (**PPO**)
4. Current progress and working example
5. Summary



Proximal Policy Optimisation (PPO)

- Developed by **OpenAI** in 2017, to increase accuracy of robotic systems and gaming agents.
- Leverages a **clipped/surrogate function** that maintains stability of learning for the agent, within an **optimal training zone** that has a pre-defined environment range.



[7]

Mathematical Modelling

$$\mathbf{L}^{\mathbf{PPO}}(\theta) = \hat{\mathbb{E}}_{\mathbf{t}}\left[\mathbf{L}^{\mathbf{CLIP}}(\theta) - \mathbf{c}_1 \mathbf{L}^{\mathbf{VF}}(\theta) + \mathbf{c}_2 \mathbf{S}[\pi_\theta](\mathbf{s}_{\mathbf{t}})\right]$$

$\mathbf{L}^{\mathbf{CLIP}}(\theta)$

Ensures **reusability** of the policy updates.

$\mathbf{c}_1 \mathbf{L}^{\mathbf{VF}}(\theta)$

Determines **desirability** of the current state.

$\mathbf{c}_2 \mathbf{S}[\pi_\theta](\mathbf{s}_{\mathbf{t}})$

Ensures **optimum** exploration of an environment by the agent.

PPO Pseudo-code[8]

Outer thread

Sporadically collects information to maintain optimum training probability

Inner thread

Initiates policy gradient interaction with the simulated environment to calculate and store the desirability of the action

Algorithm 1 PPO, Actor-Critic Style

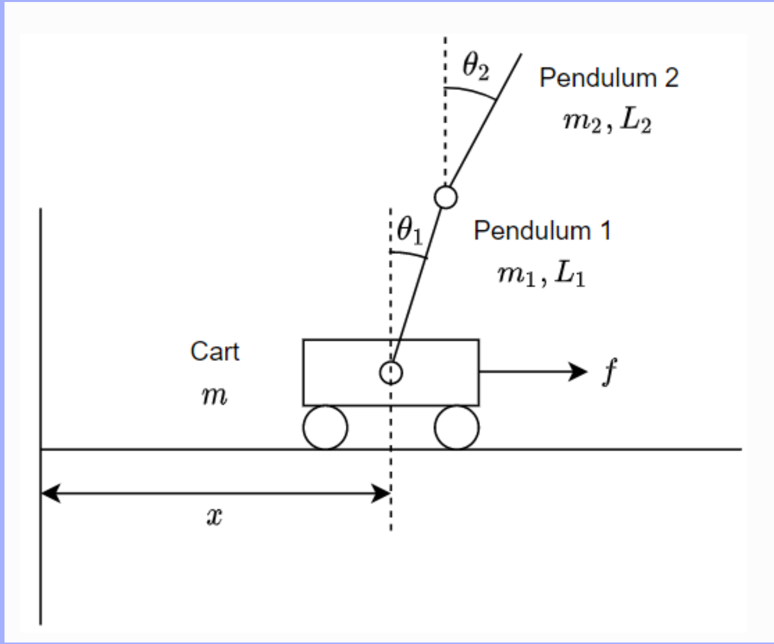
for iteration$=1,2,\ldots$ **do**
  for actor$=1,2,\ldots,N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1,\ldots,\hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \le NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

Outline

1. Artificial Intelligence (**AI**)
2. Deep Reinforcement Learning (**DRL**)
3. Proximal Policy Optimisation (**PPO**)
4. Current progress and working example
5. Summary

Problem definition and progress

- **Problem**: Use the PPO algorithm to train an agent to balance a double inverted pendulum (DIP).
- **Use cases**: Rocket initialisation trajectory, spacecraft berthing and docking, and stabilisation of human prosthetics.
- **Current**: The equations of motion and modelling of the system have been completed.
- **Future**: Develop the PPO algorithm and integrate into completed system to optimally train the agent.



10

Preliminary work

- Used the **SymPy** package to **derive** the dynamics of the double pendulum.

```
qddot_0 = (-4.0*f*cos(2.0*q_2) + 6.0*f + 4.0*qdot_1**2*cos(q_1) + qdot_1**2*cos(q_1 - q_2) - qdot_1**2*cos(q_1 + 2.0*q_2) + 2.0
*qdot_1*qdot_2*cos(q_1 - q_2) + qdot_2**2*cos(q_1 - q_2) - 29.43*sin(2.0*q_1) + 9.81*sin(2.0*q_1 + 2.0*q_2))/(3.0*cos(2.0*q_1)
- 22.0*cos(2.0*q_2) - cos(2.0*q_1 + 2.0*q_2) + 34.0)

qddot_1 = (8.0*f*sin(q_1) - 4.0*f*sin(q_1 + 2.0*q_2) + 3.0*qdot_1**2*sin(2.0*q_1) + 23.0*qdot_1**2*sin(q_2) + 22.0*qdot_1**2*si
n(2.0*q_2) + qdot_1**2*sin(2.0*q_1 + q_2) + 46.0*qdot_1*qdot_2*sin(q_2) + 2.0*qdot_1*qdot_2*sin(2.0*q_1 + q_2) + 23.0*qdot_2**2
*sin(q_2) + qdot_2**2*sin(2.0*q_1 + q_2) - 490.5*cos(q_1) + 215.82*cos(q_1 + 2.0*q_2))/(3.0*cos(2.0*q_1) - 22.0*cos(2.0*q_2) -
cos(2.0*q_1 + 2.0*q_2) + 34.0)

qddot_2 = -((100.0*qdot_1**2*sin(q_2) + 981.0*cos(q_1 + q_2))*(-9.0*(sin(q_1) + 0.333333333333333*sin(q_1 + q_2))**2 + 28.0*cos
(q_2) + 42.0) + 0.5*(200.0*qdot_1*qdot_2*sin(q_2) + 100.0*qdot_2**2*sin(q_2) - 2943.0*cos(q_1) - 981.0*cos(q_1 + q_2))*(25.0*co
s(q_2) + 3.0*cos(2.0*q_1 + q_2) + cos(2.0*q_1 + 2.0*q_2) + 13.0) + 50.0*(2.0*sin(q_1) + 3.0*sin(q_1 - q_2) - 2.0*sin(q_1 + q_2)
- sin(q_1 + 2.0*q_2))*(2.0*f + 3.0*qdot_1**2*cos(q_1) + qdot_1**2*cos(q_1 + q_2) + 2.0*qdot_1*qdot_2*cos(q_1 + q_2) + qdot_2**2
*cos(q_1 + q_2)))/(75.0*cos(2.0*q_1) - 550.0*cos(2.0*q_2) - 25.0*cos(2.0*q_1 + 2.0*q_2) + 850.0)
```

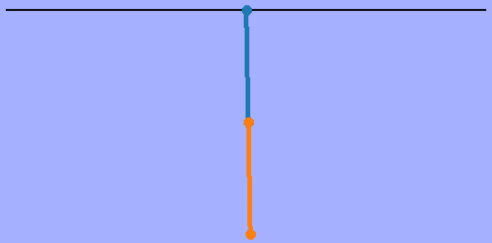
- Used the **do-mpc** package to **solve** the Euler-Lagrangian system dynamics of the double pendulum.

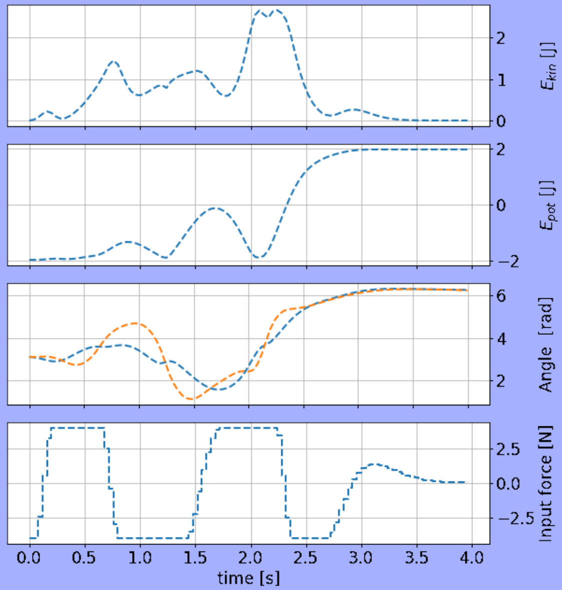
$$
\begin{aligned}
h_1\ddot{x} + h_2\ddot{\theta}_1\cos(\theta_1) + h_3\ddot{\theta}_2\cos(\theta_2) &= (h_2\dot{\theta}_1^2\sin(\theta_1) + h_3\dot{\theta}_2^2\sin(\theta_2) + u)\\
h_2\cos(\theta_1)\ddot{x} + h_4\ddot{\theta}_1 + h_5\cos(\theta_1 - \theta_2)\ddot{\theta}_2 &= (h_7\sin(\theta_1) - h_5\dot{\theta}_2^2\sin(\theta_1 - \theta_2))\\
h_3\cos(\theta_2)\ddot{x} + h_5\cos(\theta_1 - \theta_2)\ddot{\theta}_1 + h_6\ddot{\theta}_2 &= (h_5\dot{\theta}_1^2\sin(\theta_1 - \theta_2) + h_8\sin(\theta_2))
\end{aligned}
$$

https://github.com/mughees-asif/dip#double-inverted-pendulum-dip-modelling

Working example #1

```
# model parameters
m0 = 0.6  # kg, mass of the cart
m1 = 0.2  # kg_1, mass of the first rod
m2 = 0.2  # kg_2, mass of the second rod
L1 = 0.5  # m_1,  length of the first rod
L2 = 0.5  # m_2,  length of the second rod
g = 9.81 # m/s^2, Gravity
```

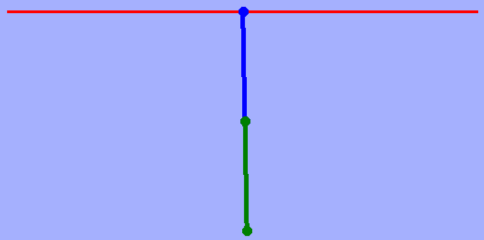


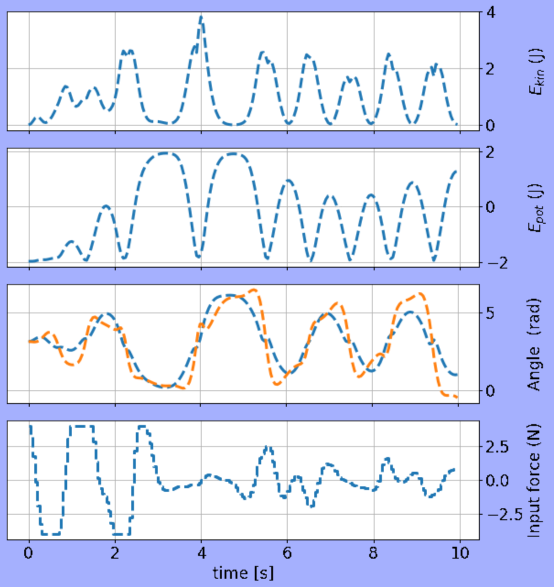


https://github.com/mughees-asif/dip#double-inverted-pendulum-dip-modelling

Working example #2

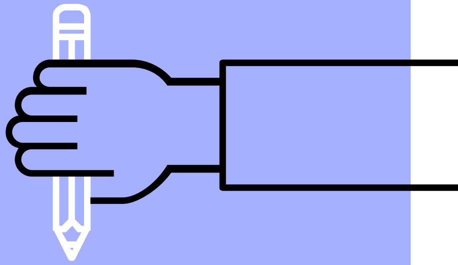
```python
# model parameters
m0 = 3  # kg, mass of the cart
m1 = 1  # kg_1, mass of the first rod
m2 = 1  # kg_2, mass of the second rod
L1 = 0.5  # m_1,  length of the first rod
L2 = 0.5  # m_2,  length of the second rod
g = 9.81 # m/s^2, Gravity
```



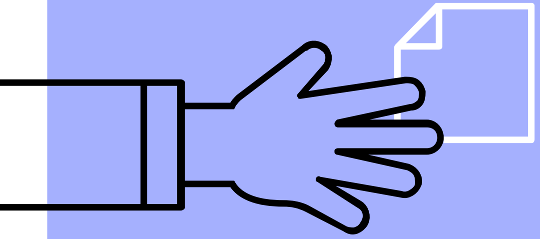


https://github.com/mughees-asif/dip#double-inverted-pendulum-dip-modelling

Outline



1. Artificial Intelligence (**AI**)
2. Deep Reinforcement Learning (**DRL**)
3. Proximal Policy Optimisation (**PPO**)
4. Working example with context to the final-year project
5. Summary



Summary

Artifical Intelligence

The ability of a machine to perceive its environment, whilst *reacting* in a way that successfully maximises a user-defined output.

Deep Reinforcement Learning

A synchronous combination of leveraging the advantages of using deep learning (DL) with reinforcement learning (RL).

Proximal Policy Optimisation

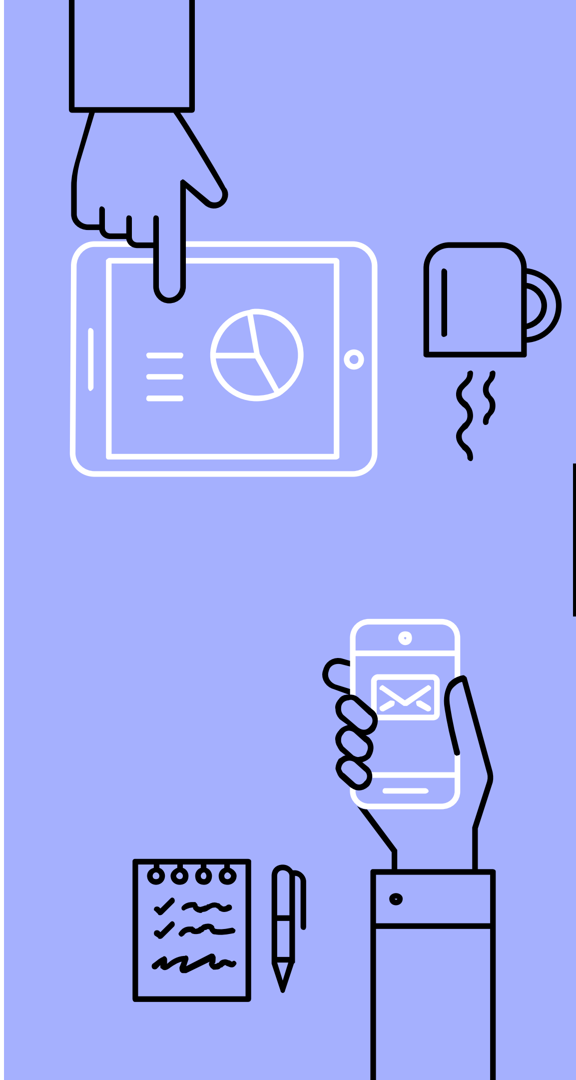
A set of deep reinforcement learning algorithms that use clipped/surrogative objectives to ensure the agent stays within an optimal training zone.

Current progress

Modelling of the system has been completed and the initial set of findings have been displayed.

Future steps

Develop the PPO algorithm that is applicable to this system and train an agent to execute the task of balancing a double inverted pendulum.



THANKS!

Any questions?

Citations

[1] Merriam-Webster. n.d. *Definition Of Artificial Intelligence*. [online] Available at: <https://www.merriam-webster.com/dictionary/artificial%20intelligence> [Accessed 24 November 2020].

[2] Lin, Y. *10 Artificial Intelligence Statistics You Need to Know in 2020* . [Infographic]. Nov. 2020. Available at: https://www.oberlo.co.uk/blog/artificial-intelligence-statistics.

[3] Gadgetify. n.d. *Flippy Burger Flipping Robotic Kitchen Assistant*. [online] Available at: <https://www.gadgetify.com/flippy-burger-robot/> [Accessed 24 November 2020].

[4] Turakhia, C., 2017. *Engineering More Reliable Transportation With Machine Learning And AI At Uber*. [online] Uber Engineering Blog. Available at: <https://eng.uber.com/machine-learning/> [Accessed 24 November 2020].

[5] Paradzhanyan, L., Gunter, A. and Brotsos, J., 2020. *Demystifying Deep Learning And Artificial Intelligence* . [online] The New Stack. Available at: <https://thenewstack.io/demystifying-deep-learning-and-artificial-intelligence/> [Accessed 25 November 2020].

[6] Mathworks. n.d. *Reinforcement Learning Agents*. [online] Available at: <https://uk.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html> [Accessed 25 November 2020].

[7] Karidis, V., 2020. *Proximal Policy Optimization: AI'S Best Way To Transfer Knowledge?* [online] Ocular Point. Available at: <https://www.ocularpoint.com/post/proximal-policy-optimization-ai-s-best-way-to-transfer-knowledge> [Accessed 26 November 2020].

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal policy optimization algorithms.* CoRR, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347.

[9] Wang C, Zhang Q, Tian Q, Li S, Wang X, Lane D, Petillot Y, Wang S. *Learning Mobile Manipulation through Deep Reinforcement Learning*. Sensors. 2020; 20(3):939.

[10] do-mpc. n.d. *Double Inverted Pendulum — Do-Mpc 4.0.0 Documentation*. [online] Available at: <https://www.do-mpc.com/en/latest/example_gallery/DIP.html> [Accessed 27 November 2020].