***Project Report***

*Car Racing with PPO, Learning from Raw Pixels*

***I. Members***

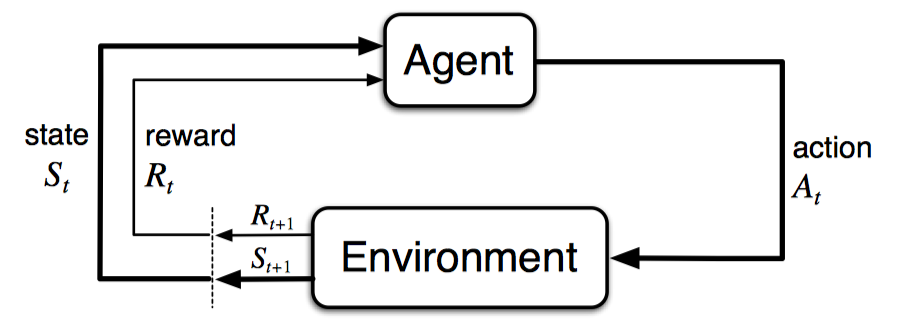
1. *Nguyễn Minh Dương (17021227)*
2. *Nguyễn Trung Hiếu (17021246) - leader , email : bipp811@gmail.com*
3. *Nguyễn Đức Lâm (17021280)*

***Assign :***

1. *Dương :*
2. *Hiếu :*
3. *Lâm :*

***II. Proximal Policy Optimization (PPO)***

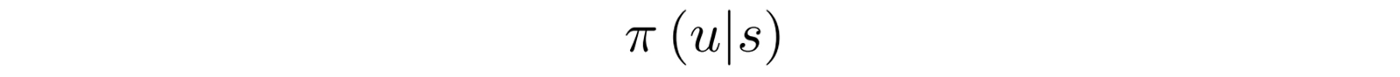
1. *Policy Gradient in RL*

******

*Principle :*

***We observe and act***

***We keep what is working and throw away what is not***

******

1. *PPO and its concepts*

It is a member of a new family of reinforcement learning methods known as [Policy Gradient methods](https://github.com/elsheikh21/car-racing-ppo/blob/master/docs/policy-optimization.md), which basically performs one policy update as per sample, however, PPO alternates between

1. Sampling data by interacting with the environment
2. Optimizing 'surrogate' objective function using **stochastic gradient ascent.**

We have introduced [PPO], a family of policy optimization methods that use **multiple epochs of stochastic gradient ascent to perform each policy update**. These methods have the stability and reliability of trust-region [[TRPO](https://arxiv.org/abs/1502.05477)] methods but are much simpler to implement, requiring **only a few lines of code change to a vanilla policy gradient implementation**, applicable in more general settings (for example, when using a joint architecture for the policy and value function), and have better overall performance.

**PPO is a on-policy !**

Online policy gradient methods are methods where

1. Agents can pick actions on their own
2. Follows most obvious setup
   1. Learn with exploration
   2. Play with exploitation
3. Agent follows his own policy
   1. Learn from experts (Imperfect)
   2. Learn from recorded sessions (Recorded Data)

### *Why PPO?*

1. It has some benefits of Trust Region Policy Optimization [TRPO], but much simpler (in terms of implementation), more general, and have better sample complexity.
2. It outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity and simplicity, and wall-time.

**(1) the Clipped Surrogate Objective**

**(2) Multiple epochs of stochastic gradient ascent to perform each policy update".**

**1 Clipped Surrogate Objective : : muc tieu thay the bi cat xen**

**The Clipped Surrogate Objective is a drop-in replacement for the policy gradient objective that is designed to improve training stability by limiting the change you make to your policy at each step.**

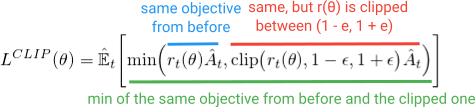
**For vanilla policy gradients**

****

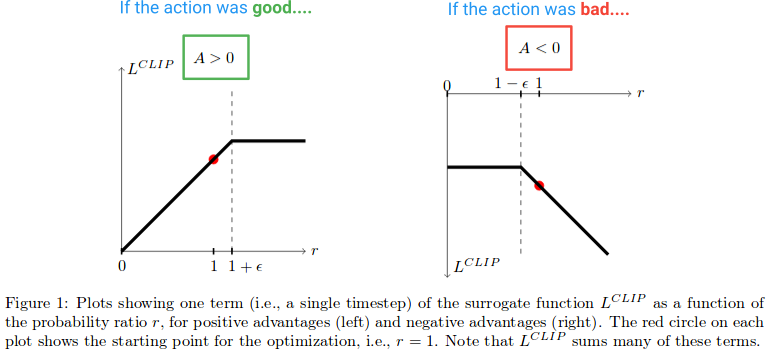
**TPRO Object function : current policy / previous policy**

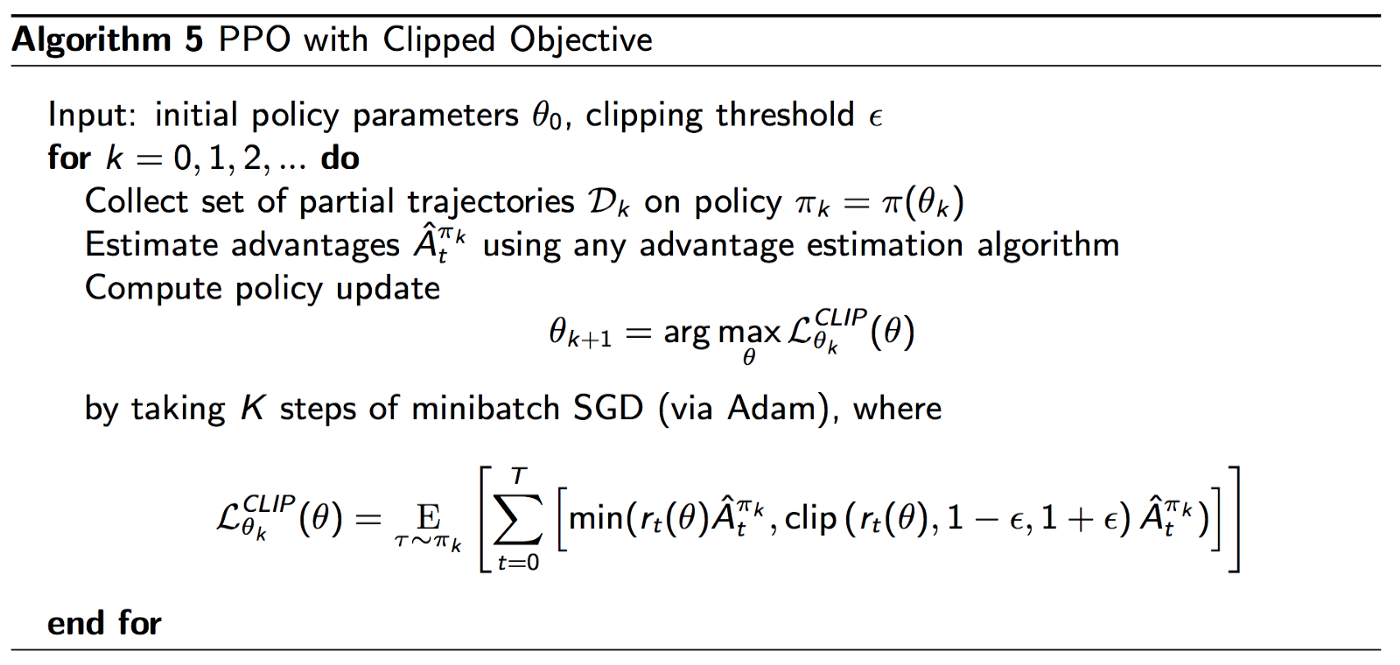
****

**PPO Object function : Clipped Surrogate Objective**

****

**?**

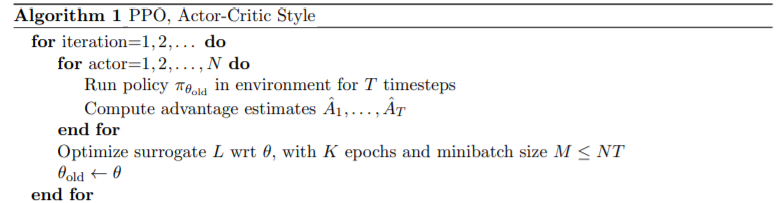
****



# **2. Multiple epochs for policy updating**

**Unlike vanilla policy gradient methods, and *because of the Clipped Surrogate Objective function*, PPO allows you to run multiple epochs of gradient ascent on your samples without causing destructively large policy updates. This allows you to squeeze more out of your data and reduce sample inefficiency.**

**PPO runs the policy using *N* parallel actors each collecting data, and then it samples mini-batches of this data to train for *K* epochs using the Clipped Surrogate Objective function. See full algorithm below (the approximate param values are: *K* = 3-15, *M* = 64-4096, *T* (horizon) = 128-2048):**



1. **What is stochastic gradient ascent**

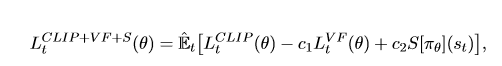
**where c1,c2 are coeﬃcients, and S denotes an entropy bonus, and LV F t is a squared-error loss (Vθ(st)−V targ t )2.**

**we must use a loss function that combines the policy surrogate and a value function error term. This objective can further be augmented by adding an entropy bonus to ensure suﬃcient exploration**

1. *Model : Convolutional Neural Network ( CNN )*
2. Convolutional Neural Networks - CNN

* CNN là một dạng đặc biệt của mạng nơ-ron được sử dụng chuyên sâu trong Computer Vision. Ứng dụng mạnh trong xe tự lái hay nhận dạng khuôn mặt.
* Kiến trúc và cách hoạt động của 1 mạng CNN bao gồm: đưa ảnh input qua các convolutional layer để trích xuất các đặc trưng của ảnh. Tiếp đó ảnh sẽ được đưa qua pooling layer nhằm giảm không gian/ dimensions để giảm độ phức tạp. Cuối cùng là thêm một mạng nơ ron với một tầng ẩn (hidden layer) ở cuối làm một fully connected layer để từ đó có thể classify ảnh

**Algorthim**

****

# Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution

This objective can further be augmented by adding an entropy bonus to ensure suﬃcient exploration

Beta

**Implementation**



* First we have raw pixels as our input to the network - multi layer perceptron and it outputs a probability of action per step
* Per epoch, it takes in a set of states and outputs the best policy.
* The best stochastic policy as it allows exploration and optimization of the problem

Every action will be repeated for 8 frames. To get velocity information, state is defined as adjacent 4 frames in shape (4, 96, 96). Use a two heads FCN ( fully convolutional network ) to represent the actor and critic respectively. The actor outputs α, β for each actin as the parameters of Beta distribution.

We use the CNN model with 6 levels Conv2d (torch.nn.Conv2d) and 6 ectified linear units ReLU (torch.nn.ReLU). The rewards, target and advatage values are calculated by the model:

*Iterate until termination condition or convergence*

1. Process observations
2. Feedforward to the NN
3. Interact with environment
4. Keep track of rewards, losses, states
5. Compute discounted rewards and advantage estimates
6. Backpropagation the error
7. perform the update for network params

III. Practice Result

Team is training agent and testing itt to get best result , we will update later .

**Ref :**

Policy Gradient <https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146>

Paper PPO : <https://arxiv.org/abs/1707.06347>

CNN : <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>

Project github : <https://github.com/xtma/pytorch_car_caring/blob/master/train.py>

SPD - Beta Distribution : <http://proceedings.mlr.press/v70/chou17a.html>