Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning

`RL` `DL` `A3C` `AI`

- **Author : Yunqiu Xu**

---

1. Introduction

- Highlights:
  - The Track 1 (limited deathmatch with known map) champion of ViZDoom AI Competition 2016, 35% better than 2nd
  - A3C + Curriculum learning
  - ICLR2017
- Why direct application of A3C to 3D games is non-trival: sparse and long-term rewards
- The paper:
  - A3C + CNN: use recent 4 frames and game variables, to predict current value and next action
  - Curriculum learning: start from simple tasks and then gradually try harder ones
  - Control the difficulty: a variaty of game environment parameters, e.g. strength of opponents
  - Adaptive curriculum training:
    - Samples from a varying distribution of tasks to train the model
    - More stable and achieve higher score than A3C with same epochs
  - Requires no opponent's information: suitable as a general framework

2. Actor-critic Model

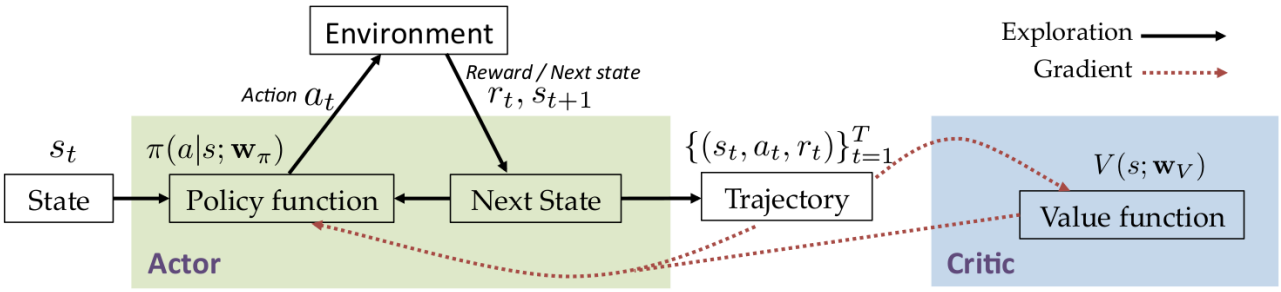

Figure 1: The basic framework of actor-critic model.

- Pick the baseline as the expect cumulative reward $V(s)$ of current state, here two functions reinforce each other:
  - correct $\pi(a|s) \rightarrow$ high-rewarding trajectories $\rightarrow$ update $V(s)$ towards the right direction
  - correct $V(s) \rightarrow$ pick right actions for $\pi(a|s)$
- A3C: independent multiple threads $\rightarrow$ reduce the correlation of game experience $\rightarrow$ less biased
- For on-polocy, same mutual behavior will lead to over estimation on a small propotion of environment $\rightarrow$ add an entropy term to loss $\rightarrow$ encourage diversity

$$\mathbf{w}_\pi \leftarrow \mathbf{w}_\pi + \alpha(R_t - V(s_t))\nabla_{\mathbf{w}_\pi} \log \pi(a_t|s_t) + \beta \nabla_{\mathbf{w}_\pi} H(\pi(\cdot|s_t)) \quad (1)$$

$$\mathbf{w}_V \leftarrow \mathbf{w}_V - \alpha \nabla_{\mathbf{w}_V} (R_t - V(s_t))^2 \quad (2)$$

- Batch A3C:
  - all agents act on the same model
  - send batches to the main process for gradient descent optimization

3. Method

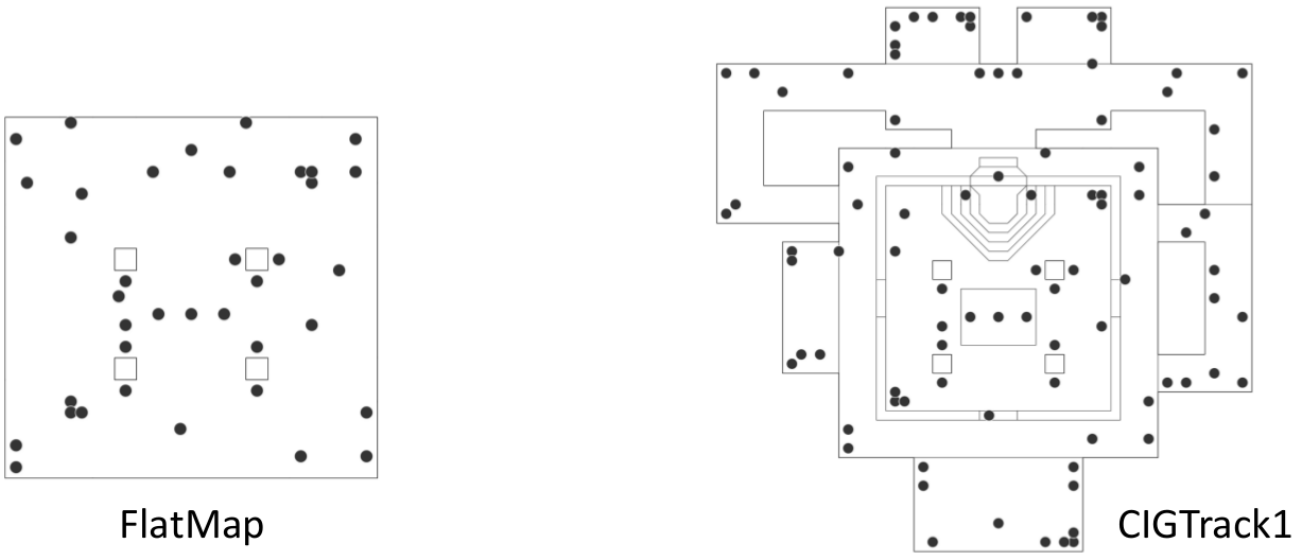

Figure 2: Two maps we used in the paper. `FlatMap` is a simple square containing four pillars . `CIGTrack1` is the map used in Track1 in ViZDoom AI Competition (We did not attend Track2). Black dots are items (weapons, ammo, medkits, armors, etc).

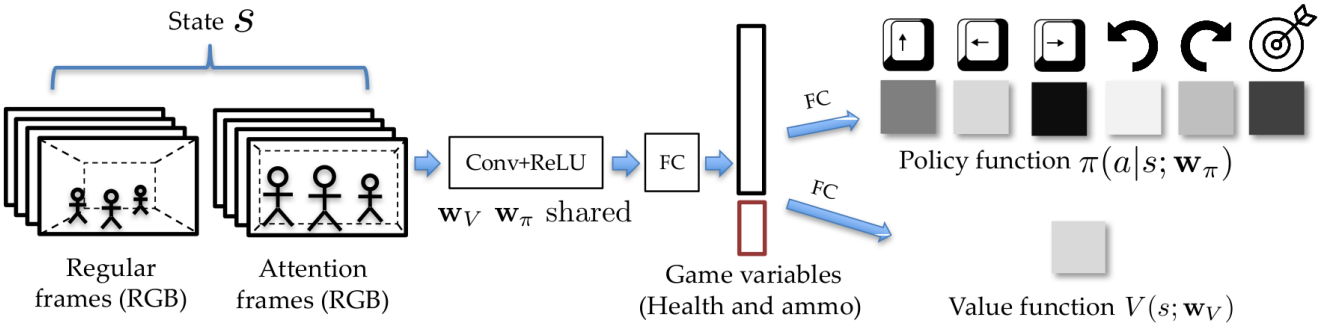

Figure 3: The network structure of the proposed model. It takes 4 recent game frames plus 4 recent attention frames as the input state s, and outputs a probability distribution $\pi(a|s)$ of the 6 discrete actions. The policy and value network share parameters.

Layer #	1	2	3	4	5	6	7
	C7x7x32s2	C7x7x64s2	MP3x3s2	C3x3x128	MP3x3s2	C3x3x192	FC1024

Table 1: Network parameters. *C7x7x32s2* = convolutional layer with 7x7 kernel, stride 2 and number of output planes 32. *MP* = MaxPooling. Each convolutional and fully connected layer is followed by a ReLU, except for the last output layer.

Parameters	Description	FlatMap	CIGTrack1
living	Penalize agent who just lives	-0.008 / action	
health_loss	Penalize health decrement	-0.05 / unit	
ammo_loss	Penalize ammunition decrement	-0.04 / unit	
health_pickup	Reward for medkit pickup	0.04 / unit	
ammo_pickup	Reward for ammunition pickup	0.15 / unit	
dist_penalty	Penalize the agent when it stays	-0.03 / action	
dist_reward	Reward the agent when it moves	9e-5 / unit distance	
dist_penalty_thres	Threshold of displacement	8	15
num_bots	Number of built-in bots	8	16

Table 2: Parameters for different maps.

- Architecture or network:
  - Input: most recent 4 frames and their center part, i.e. state s
  - The features extracted by CNN will be incorporated with game variables
  - Output: I think this is similar to Faster RCNN
    - Value function: regression
    - Policy function: regular softmax
- Training pipeline:
  - Slightly different from original A3C
  - Entropy term
  - Encourage exploration
    - multiply the policy output of the network by an exploration factor (0.2) before softmax
    - uniformly randomize the action for 10% random frames.
  - Skip frame: 3
    - Small $\rightarrow$ strong correlation
    - Big $\rightarrow$ reduces effective training samples
- Curriculum learning:
  - Why use : sparse or adversarial rewards $\rightarrow$ long time to converge
  - CL: train an agent with a sequence of progressively more difficult environments

- Reward shaping: apply reinforcement learning in a complicated environment with delayed reward
- Curriculum design: control difficulty level by changing game parameters
- Adaptive curriculum: assign a probability distribution on different levels for each thread

4. Result

4.1 Curriculum learning of flatmap

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Speed	0.2	0.2	0.4	0.4	0.6	0.8	0.8	1.0
Health	40	40	40	60	60	60	80	100

Table 3: Curriculum design for `FlatMap`. Note that enemy uses `RocketLauncher` except for Class 0 (`Pistol`).

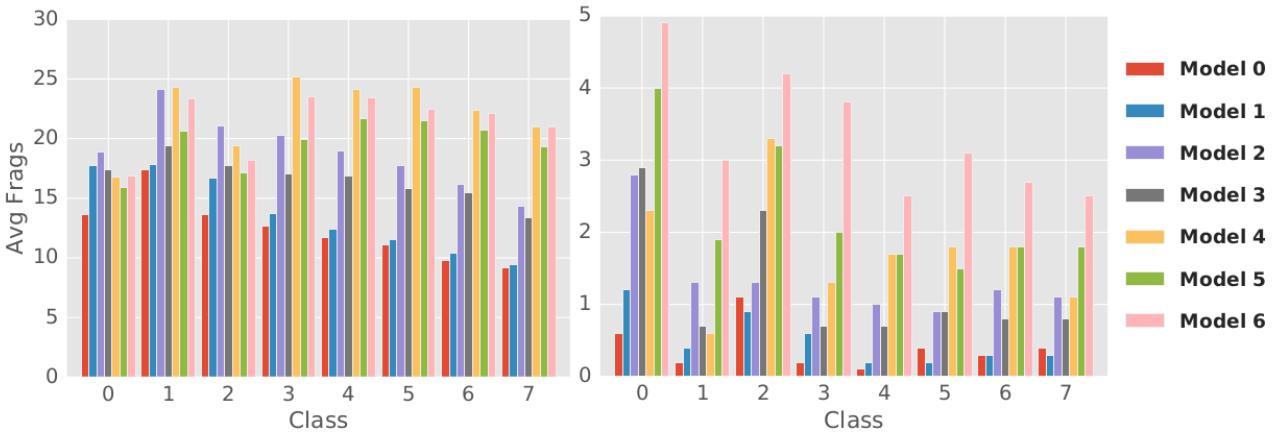

Figure 4: Average Frags over 300 episodes evaluation, on `FlatMap`(left) and `CIGTrack1`(right) with different levels of enemies (See Tbl. 3 for curriculum design). Models from later stages performs better especially on the difficult map, yet still keeps a good performance on the easier map.

- Curriculum learning increases the performance of the agents
- When an agent becomes stronger in the higher level of class, it is also stronger in the lower level of class without overfitting

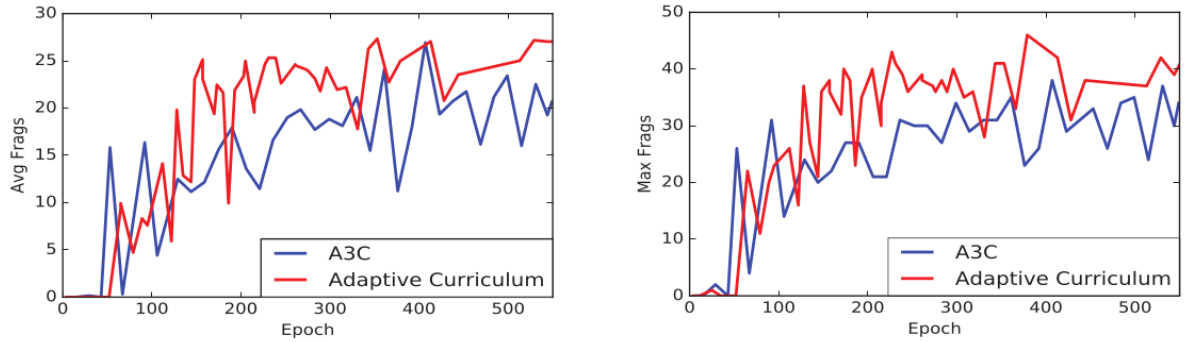

Figure 5: Performance comparison on Class 7 (hardest) of `FlatMap` between A3C [Mnih et al. (2016)] and adaptive curriculum learning, at different stage of training. Average frags and max frags are computed from 100 episodes. Adaptive curriculum shows higher performance and is relatively more stable.

- Pure A3C can learn on FlatMap but is slower
- In CIGTrack1, a direct application of A3C does not yield sensible performance.

4.2 Ablation analysis:

- The convolutional kernels of the current frame is less noisy than previous frames
  → the agent focuses on the current frame

	FlatMap			CIGTrack1		
	Min	Mean	Max	Min	Mean	Max
F1 bot (reverse history)	1	9.89	19	-2	2.39	9
F1 bot (duplicated history)	10	24.62	37	2	8.50	17
F1 bot (w/o PT rules)	14	22.80	36	1	8.66	18
F1 bot	16	25.17	37	5	10.34	17

Table 5: Performance evaluation (in terms of frags) on two standard scenarios `FlatMap` and `CIGTrack1` over 300 episodes. Our bot performs better with post-training rules.

- Effect of history frames : the agent uses motion information for better decision
- Post-training rules: improve the performance.

4.3 Total results

Round	1	2	3	4	5	6	7	8	9	10	11	12	Total
Our bot	**56**	**62**	n/a	**54**	**47**	43	**47**	**55**	**50**	**48**	**50**	**47**	**559**
Arnold	36	34	**42**	36	36	**45**	36	39	n/a	33	36	40	413
CLYDE	37	n/a	38	32	37	30	46	42	33	24	44	30	393

Table 6: Top 3 teams in ViZDoom AI Competition, Track 1. Our bot attended 11 out of 12 games, won 10 of them and won the champion by a large margin. For design details, see Arnold [Lample & Chaplot (2016)] and CLYDE [Ratcliffe et al. (2017)].