DL Competition3 Team24 Report

周聖諺 Shen-Yen Chou

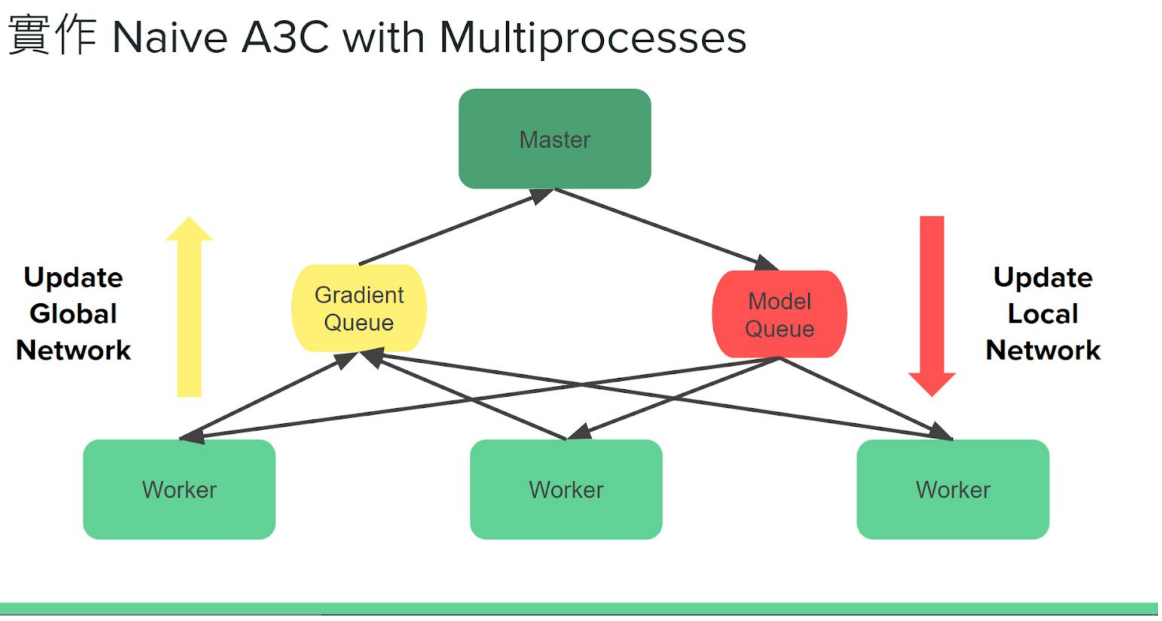
馮翔荏 FENG-XIANG REN

1. Model

Model: A3C, 128 dense layer

Optimizer: Adam with Learning Rate 0.0001

2. Architecture



We use master-slave multiprocessing architecture, since Python doesn't support real multi-threading. Every worker has an independent environment and local model. When they finish an episode, they push the gradients into the gradient queue and receive the latest model from the master from the model queue.

3. Observations & Tricks

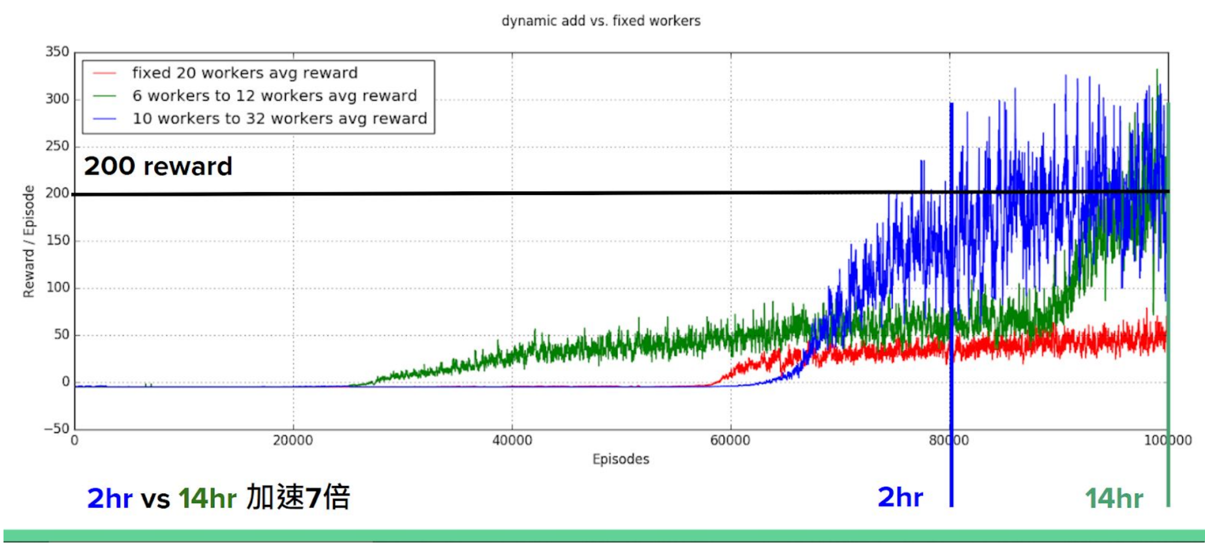
(1) Unstable convergence of A3C: Since A3C updates the model asynchronously, the gradient is not correct(but close). It is hard to converge well.

(2) Master overload: Since the workers update the model frequently in the beginning, the master cannot handle so many gradients. Once the worker > 20, it would collapse. However, when the episode gets longer, the master updates less time.

(3) Parallel on TF2: SInce the TF2 is not compatible well on multiprocessing, there are many bugs and tricks to parallelize the TF2. For more detail, we write in the appendix https://github.com/FrankCCCCC/rl_collection/blob/master/docs/tricks_of_A3C_with_tf2.md

(4) Dynamic multiprocessing: Due to the master overload, we decrease the worker in the beginning and then add new workers when the episodes get longer. We measure the average time of one episode and once the time of an episode > 20s, we add a new worker. When the average time of an episode increases every 10s, we add another worker. To make sure the new worker gets stable, there are at least 1000 episodes between 2 new workers. It seems 7 times faster than the original one in the 100000 episodes.

動態增加所花的時間



(5) Entropy regularization: It can enhance the exploration ratio of the agent, but it is hard to tune well. We've tried it, but it is hard to converge.
(6) Low sample-efficiency: It is the biggest problem of Actor-Critic. However we don't have enough time to implement distributed PPO. It should have a better result.

4. Loss

Because we train each work as an A2C agent, we compute each loss locally. The formula is : Sum of a trajectory(-log(Pr(s, a| Theta)) * Advantage + coefficient of value * Value - coefficient of entropy * cross entropy of action distribution). We replace the Advantage to actor loss and replace the Value to critical loss.

5. bonus model

In order to change the input from game state to frame ,we used Conv2D layers to build the model .It is the same as the LAB 17 and replaces the dense layer from 512 units to 128 units.

6. Appendix

(1)Tricks of parallelize TF2

Problem1: Functional Process instead of Inherited Process Class

眾所皆知，在Python要Spawn一個新的Process有兩種方式，一種是繼承mutiprocessing.process並修改run() method，另一種是直接將Function傳入Process。如果使用繼承process的方式實作，會出現Cannot pickle的Error，網

路上普遍的說法是因為Tensorflow底層是由C實作，所以很多物件無法轉換成Python的binary pickle檔，所以才會出現此錯誤。

Problem2: Use with tf.device() to specify the wanted device

在很多TF 1.x的A3C實作，可以看到都用了server = tf.train.server這個API，然後在每個Worker的Session會用tf.Session(target = server.target)，給不同Worker指定不同的計算資源，但TF2.0移除了tf.Session，如果直接在新Process呼叫Tensorflow的API的話，就會出現Blas GEMM的Error，所以如果要只用TF 2.0的API指定計算資源的話，就可以用tf.device()完成。

另外，Blas GEMM的錯誤，可以用限制Tesorflow占用的GPU memory解決

tf_config = tf.ConfigProto()

tf_config.gpu_options.allow_growth = True

tf_config.gpu_options.per_process_gpu_memory_fraction = 0.9

tf_config.allow_soft_placement = True

Problem3: Limit the CUDA_VISIBLE_DEVICES

避免Run out of memory的問題，因為Tensorflow預設的執行方式會盡量Allocate所有能用的GPU記憶體來加快執行速度，如果同時間又有其他任務占用該GPU，就會導致TF沒辦法Allocate足夠資源導致錯誤，所以在有多個GPU的共用機器上，最好就直接指定一個沒有使用的GPU來使用。但如果只有一個GPU且沒有其他任務占用該GPU的話，一般來說不用設定。