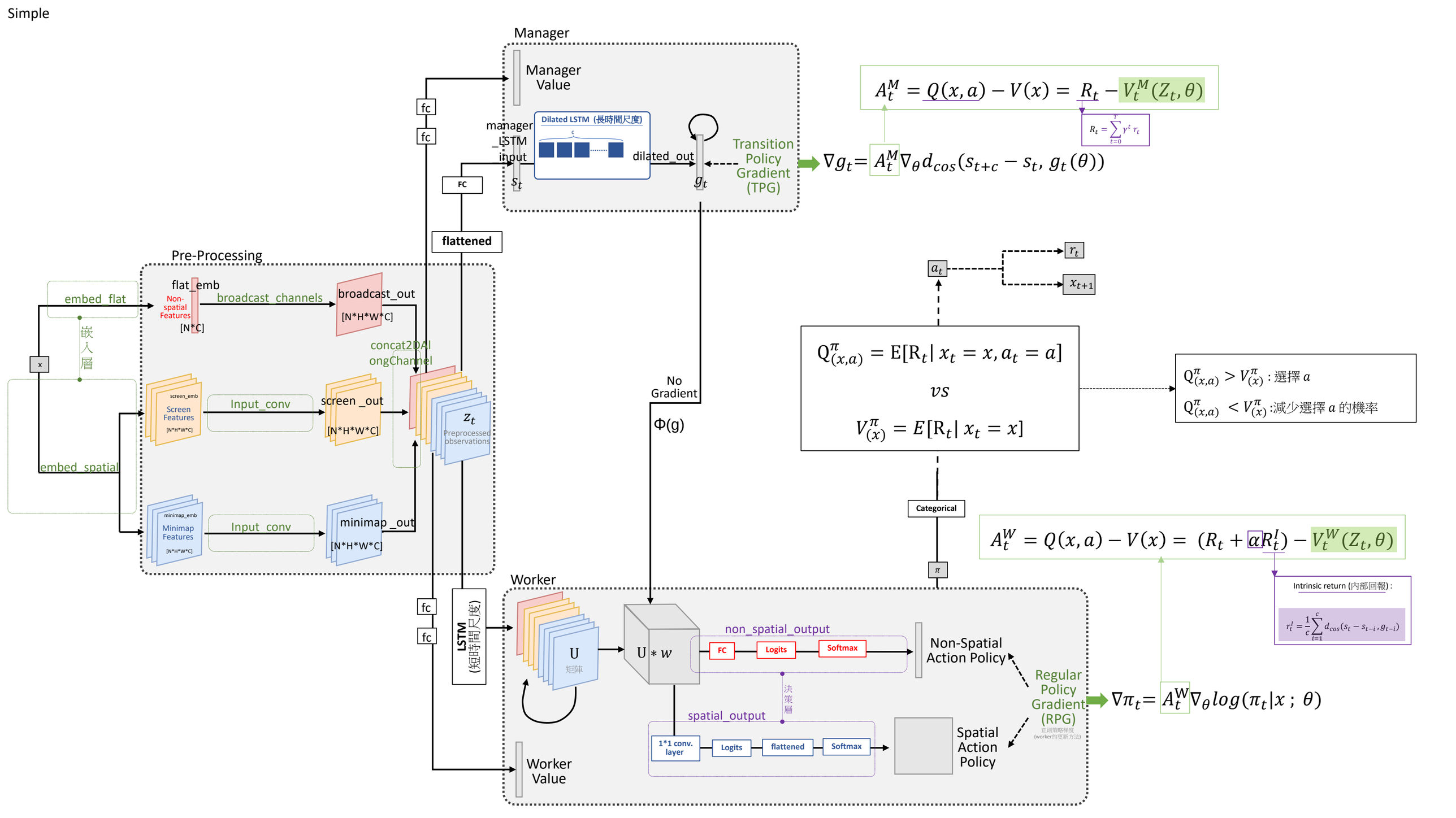
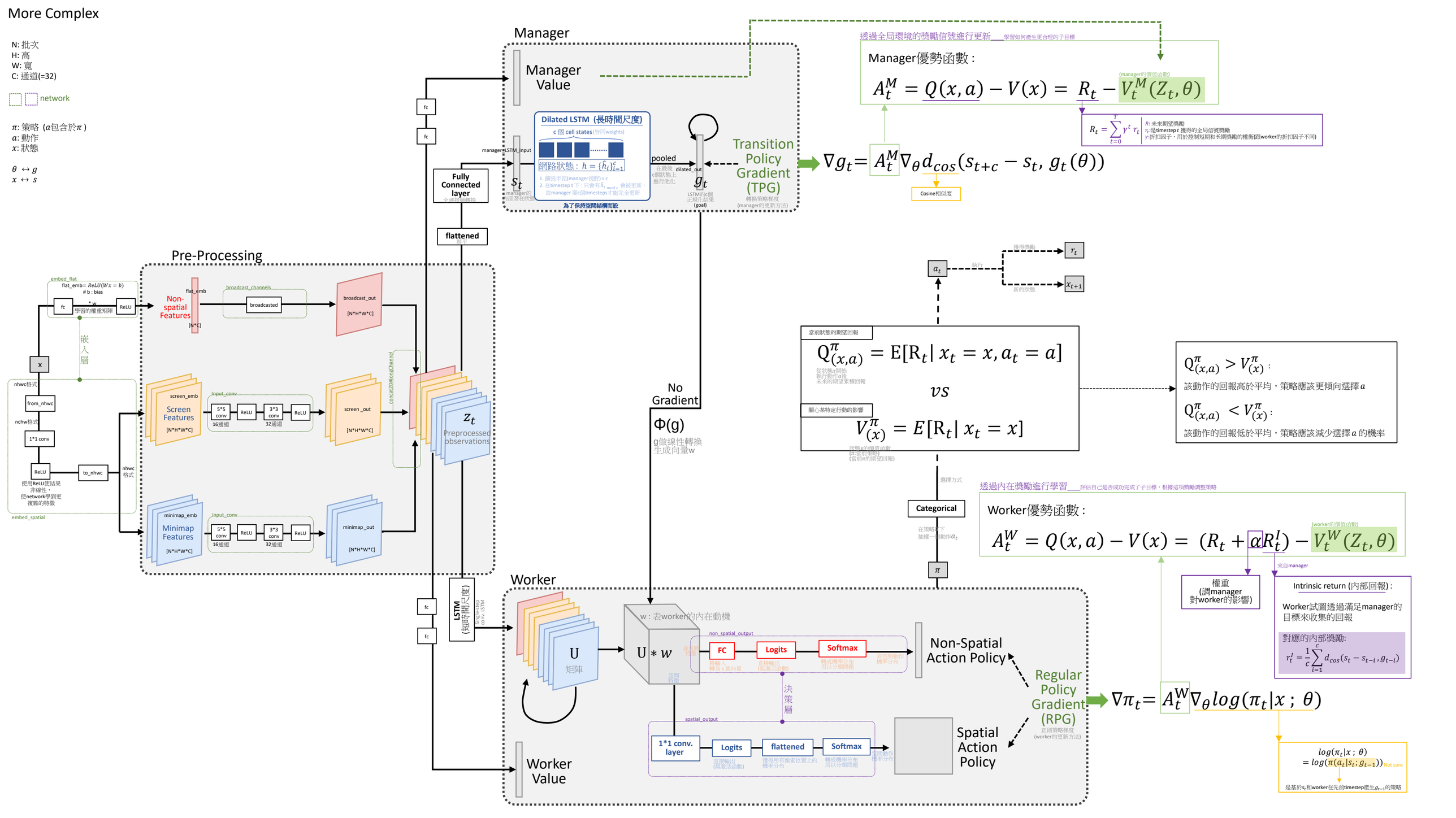

Simple
Manager
Manager Value
fc
fc
FC
manager _LSTM input
Dilated LSTM (長時間尺度)
c
dilated_out
s_t
g_t
Transition Policy Gradient (TPG)
$A_t^M = Q(x,a) - V(x) = R_t - V_t^M(Z_t,\theta)$
$R_t = \sum_{t=0}^{T} \gamma^t r_t$
$\nabla g_t = A_t^M \nabla_\theta d_{cos}(s_{t+c} - s_t, g_t(\theta))$
flattened
Pre-Processing
embed_flat
嵌入層
x
flat_emb
Non-spatial Features
[N*C]
broadcast_channels
broadcast_out
[N*H*W*C]
concat2DAlongChannel
screen_emb
Screen Features
[N*H*W*C]
Input_conv
screen _out
[N*H*W*C]
z_t
Preprocessed observations
embed_spatial
minimap_emb
Minimap Features
[N*H*W*C]
Input_conv
minimap _out
[N*H*W*C]
No Gradient
Φ(g)
a_t
r_t
x_{t+1}
$Q^\pi_{(x,a)} = E[R_t | x_t = x, a_t = a]$
vs
$V^\pi_{(x)} = E[R_t | x_t = x]$
$Q^\pi_{(x,a)} > V^\pi_{(x)}$: 選擇 a
$Q^\pi_{(x,a)} < V^\pi_{(x)}$:減少選擇 a 的機率
Categorical
π
$A_t^W = Q(x,a) - V(x) = (R_t + \alpha R_t^I) - V_t^W(Z_t,\theta)$
Intrinsic return (內部回報) :
$r_t^I = \frac{1}{c}\sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_{t-i})$
Worker
fc
fc
LSTM (短時間尺度)
U
矩陣
U * w
non_spatial_output
FC
Logits
Softmax
Non-Spatial Action Policy
決策層
spatial_output
1*1 conv. layer
Logits
flattened
Softmax
Spatial Action Policy
Regular Policy Gradient (RPG)
正則策略梯度 (worker的更新方法)
$\nabla \pi_t = A_t^W \nabla_\theta log(\pi_t | x\ ;\ \theta)$
Worker Value


More Complex
N: 批次
H: 高
W: 寬
C: 通道(=32)
network
π: 策略 (a包含於π)
a: 動作
x: 狀態
θ ↔ g
x ↔ s
Manager
Manager Value
Dilated LSTM (長時間尺度)
c 個 cell states(皆同weights)
網路狀態： h = {ĥ_i}_{i=1}^c
1. 擴張半徑(manager視野) = c
2. 在timestep t 下：只會有ĥ_{t mod c} 會被更新，故manager 要c個timesteps才能完全更新
為了保持空間結構而設
manager LSTM_input
manager的全部潛在狀態
s_t
pooled
在最後c個狀態上進行池化
dilated_out
g_t
LSTM的c個正規化結果(goal)
Transition Policy Gradient (TPG)
轉換策略梯度(manager的更新方法)
fc
fc
Fully Connected layer
全連接層轉換
flattened
降維
Pre-Processing
embed_flat
flat_emb= ReLU(Wx = b)
# b : bias
* w
學習的權重矩陣
fc
ReLU
嵌入層
X
nhwc格式
from_nhwc
nchw格式
1*1 conv
ReLU
使用ReLU使結果非線性，使network學到更複雜的特徵
to_nhwc
nhwc格式
embed_spatial
Non-spatial Features
[N*C]
flat_emb
broadcast_channels
broadcasted
broadcast_out
[N*H*W*C]
screen_emb
Screen Features
[N*H*W*C]
input_conv
5*5 conv
16通道
ReLU
3*3 conv
32通道
ReLU
screen _out
[N*H*W*C]
concat2DAlongChannel
minimap_emb
Minimap Features
[N*H*W*C]
input_conv
5*5 conv
16通道
ReLU
3*3 conv
32通道
ReLU
minimap_out
[N*H*W*C]
z_t
Preprocessed observations
No Gradient
Φ(g)
g做線性轉換生成向量w
fc
fc
LSTM (短時間尺度)
Single-step conv LSTM
Worker
U
矩陣
U * w
w : 表worker的內在動機
空間轉置
non_spatial_output
FC
Logits
Softmax
決策層
spatial_output
1*1 conv. layer
Logits
flattened
Softmax
Worker Value
Non-Spatial Action Policy
Spatial Action Policy
Regular Policy Gradient (RPG)
正則策略梯度(worker的更新方法)
透過全局環境的獎勵信號進行更新___學習如何產生更合理的子目標
Manager優勢函數 :
A_t^M = Q(x,a) − V(x) = R_t − V_t^M(Z_t, θ)
(manager的價值函數)
R_t = Σ_{t=0}^{T} γ^t r_t
R: 未來期望獎勵
r_t:是timestep t 獲得的全局信號獎勵
γ:折扣因子，用於控制短期和長期獎勵的權衡(跟worker的折扣因子不同)
∇g_t = A_t^M ∇_θ d_cos(s_{t+c} − s_t, g_t(θ))
Cosine相似度
a_t
執行
獲得獎勵
r_t
新的狀態
x_{t+1}
當前狀態的期望回報
Q^π_(x,a) = E[R_t | x_t = x, a_t = a]
從狀態x開始執行動作a後未來的期望累積回報
vs
關心某特定行動的影響
V^π_(x) = E[R_t | x_t = x]
狀態x的價值函數(π:當前策略)(當前π的期望回報)
Q^π_(x,a) > V^π_(x) :
該動作的回報高於平均，策略應該更傾向選擇 a
Q^π_(x,a) < V^π_(x) :
該動作的回報低於平均，策略應該減少選擇 a 的機率
選擇方式
Categorical
在策略π下抽樣一個動作a_t
π
透過內在獎勵進行學習___評估自己是否成功完成了子目標，根據這項獎勵調整策略
Worker優勢函數 :
A_t^W = Q(x,a) − V(x) = (R_t + αR_t^I) − V_t^W(Z_t, θ)
(worker的價值函數)
權重 (調manager對worker的影響)
來自manager
Intrinsic return (內部回報) :
Worker試圖透過滿足manager的目標來收集的回報
對應的內部獎勵:
r_t^I = (1/c) Σ_{i=1}^{c} d_cos(s_t − s_{t−i}, g_{t−i})
∇π_t = A_t^W ∇_θ log(π_t|x ; θ)
log(π_t|x ; θ) = log(π(a_t|s_t; g_{t−1}))
Not sure
是基於s_t和worker在先前timestep產生g_{t−1}的策略

More Complex(更新)

N: 批次
H: 高
W: 寬
C: 通道(=32)

network

agent

π: 策略 (a包含於π)
a: 動作
x: 狀態

$\theta \leftrightarrow g$
$x \leftrightarrow s$

Manager

Manager Value

Dilated LSTM (長時間尺度)

c 個 cell states(皆同weights)

網路狀態： $h = \{\hat{h}_i\}_{i=1}^{c}$

1. 擷選半徑(manager視野) = c
2. 在timestep t 下：只會有$h_{t \bmod c}$ 會被更新，故manager 要c個timesteps才能完全更新

為了保持空間結構而設

manager_LSTM_input

s_t manager的目前潛在狀態

pooled 在最後c個狀態上進行池化

dilated_out

g_t LSTM的c個正規化結果 (goal)

Transition Policy Gradient (TPG) 轉換策略梯度 (manager的更新方法)

fc

fc

Fully Connected layer 全連接層轉換

flattened 拉平

$g_t = g_{t-1} + \Delta g$

g_t: manager在timestep t 設定的goal
Δg: goal的變化量，由manager network產生

No Gradient

Φ(g)

g做線性轉換生成向量w

Pre-Processing

embed_flat

flat_emb= $ReLU(Wx = b)$
# b : bias
* w

fc 學習的權重矩陣 ReLU

嵌入層

X

nhwc格式

from_nhwc

nchw格式

1*1 conv

ReLU 使用ReLU使結果非線性，使network學到更複雜的特徵

to_nhwc

nhwc格式

embed_spatial

Non-spatial Features [N*C]

flat_emb

broadcast_channels

broadcasted

broadcast_out [N*H*W*C]

screen_emb

Screen Features [N*H*W*C]

input_conv

5*5 conv 16通道 ReLU 3*3 conv 32通道 ReLU

screen _out [N*H*W*C]

concat2DAlongChannel

minimap_emb

Minimap Features [N*H*W*C]

input_conv

5*5 conv 16通道 ReLU 3*3 conv 32通道 ReLU

minimap_out [N*H*W*C]

z_t Preprocessed observations

Worker

LSTM (短時間尺度) Single-step conv LSTM

fc

fc

U 矩陣

w : 表worker的內在動機

U * w 空間轉置

non_spatial_output

FC 非空間特徵 將輸入轉為 c 維向量

Logits 直接輸出 (無激活函數)

Softmax 轉成機率分布 用以分類問題

非空間動作機率分布

Non-Spatial Action Policy

決策層

spatial_output

1*1 conv. layer

Logits 直接輸出 (無激活函數)

flattened 獲得所有像素位置上的機率分布

Softmax 轉成機率分布 用以分類問題

空間動作機率分布

Spatial Action Policy

Worker Value

a_t 動作

獲得獎勵 r_t

新的狀態 x_{t+1}

當前狀態的期望回報

$Q^{\pi}_{(x,a)} = E[R_t | x_t = x, a_t = a]$

從狀態x開始 執行動作a後 未來的期望累積回報

vs

關心某特定行動的影響

$V^{\pi}_{(x)} = E[R_t | x_t = x]$

狀態x的價值函數 (π:當前策略) (當前π的期望回報)

$Q^{\pi}_{(x,a)} > V^{\pi}_{(x)}$:
該動作的回報高於平均，策略應該更傾向選擇a

$Q^{\pi}_{(x,a)} < V^{\pi}_{(x)}$:
該動作的回報低於平均，策略應該減少選擇a 的機率

選擇方式

Categorical 在策略π下抽樣一個動作a_t

π

透過全局環境的獎勵信號進行更新

學習如何產生更合理的子目標

Manager_Value_Loss =

$$A^M_t = \underbrace{Q(x,a)}_{\text{概念}} - V(x) = G_t - \underbrace{V^M_t(Z_t,\theta)}_{\text{論文}} = \underbrace{\frac{1}{2}E\left[[G_t - V(s_t)]^2\right]}_{\text{程式}}$$

(manager的值函數)

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

G_t: timestep t 的累積獎勵
R_t:當前 timestep t 獲得的全局信號獎勵
γ:折扣因子，用於控制範圍在(0,1，用來控制長期獎勵的影響(=0.99)

$V(s_t)$:狀態 s_t 的 值函數 (Value Function)

$$\nabla\theta_m = -E[log\pi_{\theta_m}(g_t|s_t)A^M_t]$$

manager_loss

透過內在獎勵進行學習

評估自己是否成功完成了子目標，根據這項獎勵調整策略

Worker_Value_Loss =

$$A^W_t = Q(x,a) - V(x) = (R_t + \alpha R^I_t) - \underbrace{V^W_t(Z_t,\theta)}_{\text{論文}} = \underbrace{\frac{1}{2}E\left[[R_I - V_{worker}(s_t)]^2\right]}_{\text{程式}}$$

(worker的價值函數)

概念

權重 (調manager對worker的影響)

來自manager

Intrinsic return (內部回報) :

Worker試圖透過滿足manager的目標來收集的回報

對應的內部獎勵:

$$r^I_t = \frac{1}{c}\sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_t - i)$$

$$R_I = g_t \cdot \Delta s_t$$

g_t : Manager給的goal
Δs_t : Worker 在環境中的位移
· :表示dot，確保 Worker 的action與goal對齊

$V_{worker}(s_t)$: worker預測的價值函數

$$\nabla\theta_w = -E[\nabla_\theta log\pi_{\theta_w}(a_t|s_t)A^W_t]$$

worker_loss

$\pi_\theta(a_t|s_t)$: 策略網路（Policy Network）的機率分佈

Regular Policy Gradient (RPG) 正則策略梯度 (worker的更新方法)

```
python "C:\Users\yuhsu\pysc2-rl-agents\run.py"
experiment_03
--agent feudal
--map MoveToBeacon
--envs 4
--res 32
--steps_per_batch 32
--iters 100000
--lr 0.0005
--entropy_weight 0.01
--save_iters 2000
--summary_iters 50
--vis
--value_loss_weight 0.7
--discount 0.99
```

學習率 α 使用指數衰減：

$$\alpha_t = \alpha_0 \cdot 0.94^{\frac{t}{10000}}$$

- 初始學習率 $\alpha_0 = 0.0005$
- 每 **10,000** 步，學習率乘以 **0.94**

$$L_{\text{entropy, manager}} = -\mathbb{E}\left[\sum_g \pi_{\theta_m}(g|s)\log\pi_{\theta_m}(g|s)\right]$$

$$L_{\text{entropy, worker}} = -\mathbb{E}\left[\sum_a \pi_{\theta_w}(a|s)\log\pi_{\theta_w}(a|s)\right]$$

manager_entropy + worker_entropy

$\lambda_1 = 0.7$ $\lambda_2 = 0.01$

(manager_loss + worker_loss) + Value_loss_weight * (manager_value_loss + worker_value_loss) + entropy_weight * (entropy)

策略損失 (Policy Loss)

來自 Actor-Critic 方法
目標是最大化累積獎勵

值函數損失 (Value Loss)

用於減少 Agent 預測的價值
和 實際累積獎勵 之間的誤差

熵損失 (Entropy Loss)

熵損失 H(π) 用來增加策略的隨機性
以鼓勵探索