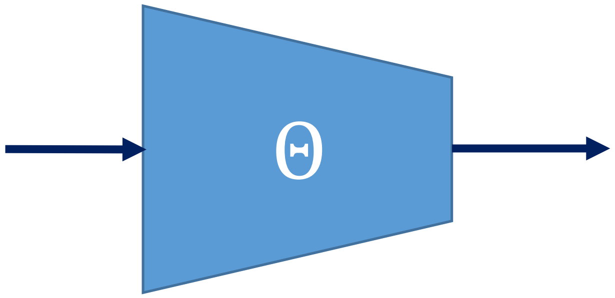
# NN to play 3D ball balancing

2. NN with policy interacts with 3D Ball (MLAgent 10).ipynb

$\Theta$: neural network weights and biases

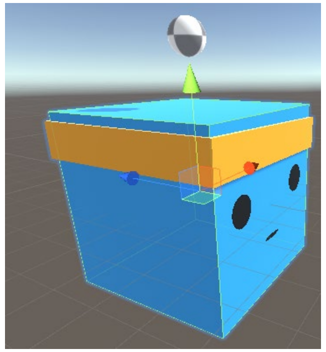
$$s = (\theta_x, \theta_z, \Delta_x, \Delta_y, \Delta_z, v_x, v_x, v_x)$$




8-128-128-2

$$p_\Theta(\Delta\theta_x|s)$$

$$p_\Theta(\Delta\theta_z|s)$$



# NN interacts with training environment

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_{2,} \cdots s_T, a_T)$$

$$p_\Theta(\tau) = p(s_1)p_\Theta(a_1|s_1)p(s_2|s_1, a_1)p_\Theta(a_2|s_2)p(s_3|s_2, a_2)\cdots$$

$$\mathrm{R}(\tau) = \sum_{t=1}^{T} \mathrm{r}_t$$

$$\bar{R}_\Theta = \sum_{\tau} R(\tau)p_\Theta(\tau) = E_{\tau\sim p_\Theta(\tau)}[R(\tau)]$$

Ref: 李弘毅 DRL lecture [https://youtu.be/z95ZYgPgXOY](https://youtu.be/z95ZYgPgXOY)

# Find NN parameters that maximize $E[\bar{R}_\Theta]$

$$\max_{\Theta} E[\bar{R}_\Theta] \qquad \bar{R}_\Theta = \sum_{\tau} R(\tau) p_\Theta(\tau)$$

$$\nabla \bar{R}_\Theta = \sum_{\tau} R(\tau) \nabla p_\Theta(\tau) \qquad \nabla f(x) = f(x) \nabla \log f(x)$$

$$= \sum_{\tau} R(\tau) p_\Theta(\tau) \nabla \log p_\Theta(\tau)$$

$$= E_{\tau \sim p_\Theta(\tau)} [R(\tau) \nabla \log p_\Theta(\tau)]$$

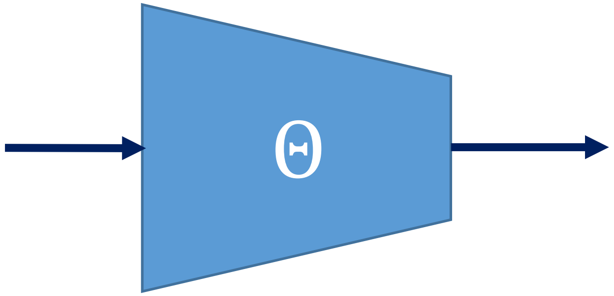
$$\approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla \log p_\Theta(\tau^n)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p_\Theta(a_t^n | s_t^n)$$

Ref: 李弘毅 DRL lecture https://youtu.be/z95ZYgPgXOY

# Use $\nabla \bar{R}_{\Theta}$ to update policy network

$$s = (\theta_x, \theta_z, \Delta_x, \Delta_y, \Delta_z, v_x, v_x, v_x)$$




$$p_{\Theta}(\Delta\theta_x|s)$$

$$p_{\Theta}(\Delta\theta_z|s)$$

$$\Theta^{\pi'} \leftarrow \Theta^{\pi} + \eta\nabla\bar{R}_{\Theta}$$

# Tips to reduce bias and variance in estimating $\nabla\bar{R}_{\Theta}$

$$\nabla\bar{R}_{\theta} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla\log p_{\theta}(a_t^n|s_t^n)$$

Add a baseline to calculate the reward

$$\nabla\bar{R}_{\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} (R(\tau^n) - b)\nabla\log p_{\theta}(a_t^n|s_t^n), \qquad b \approx E[R(\tau)]$$

$$\nabla\bar{R}_{\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} \left(\sum_{t'}^{T_n} r_{t'}^n - b\right)\nabla\log p_{\theta}(a_t^n|s_t^n)$$

# Tips to reduce bias and variance in estimating $\nabla\bar{R}_{\Theta}$

$$\nabla\bar{R}_{\theta} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla \log p_{\theta}(a_t^n|s_t^n)$$

$$\nabla\bar{R}_{\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'}^{T_n} r_{t'}^n - b\right)\nabla \log p_{\theta}(a_t^n|s_t^n)$$

Assign suitable time delayed credit

$$\nabla\bar{R}_{\theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'}^{T_n} \gamma^{t'-t} r_{t'}^n - b\right)\nabla \log p_{\theta}(a_t^n|s_t^n), \gamma < 1$$

$$A^{\theta}(s_t, a_t) = \left(\sum_{t'}^{T_n} \gamma^{t'-t} r_{t'}^n - b\right)$$

# Interact with training environment to collect training data



**3. NN with policy interacts with 3D Ball to collect training data (MLAgent_10).ipynb**

while frame < max_frames
  while (episodes < buffer_size)
    get $s_1$
    for step in range(time_horizon):
      $(s_1, a_1, r_1, s_2), v_1, \log p_1$
      $(s_2, a_2, r_2, s_3), v_2, \log p_2$
      …
      $(s_N, a_N, r_N, s_{N+1}), v_N, \log p_N$
  calculate GAE
  use GAE to update NN

```
hyperparameters:
  batch_size: 2048
  buffer_size: 20480
  learning_rate: 0.0003
```

```
keep_checkpoints: 5
max_steps: 5000000
time_horizon: 1000
summary_freq: 30000
threaded: true
```

Time_horizon: This parameter trades off between a less biased, but higher variance estimate (long time horizon) and more biased, but less varied estimate (short time horizon). In cases where there are frequent rewards within an episode, or episodes are prohibitively large, a smaller number can be more ideal. This number should be large enough to capture all the important behavior within a sequence of an agent's actions.

Buffer size: larger value corresponds to more stable training updates

# Calculate GAE

Δ = reward of this step + expected reward of next step

gae = Δ + accumulated gae

Return = gae + v

$$\Delta_{20} = r_{20} + \gamma * v_{21} * mask_{20} - v_{20}$$
$$gae_{20} = \Delta_{20} + \gamma * \tau * mask_{20} * gae_{initial}$$
$$return_{20} = gae_{20} + v_{20}$$

$$\Delta_{19} = r_{19} + \gamma * v_{20} * mask_{19} - v_{19}$$
$$gae_{19\sim 20} = \Delta_{19} + \gamma * \tau * mask_{19} * gae_{20}$$
$$return_{19} = gae_{19\sim 20} + v_{19}$$

...

$$\Delta_{1} = r_{1} + \gamma * v_{2} * mask_{1} - v_{1}$$
$$gae_{1\sim 20} = \Delta_{1} + \gamma * \tau * mask_{1} * gae_{2\sim 20}$$
$$return_{1} = gae_{1\sim 20} + v_{1}$$

```
hyperparameters:
  batch_size: 2048
  buffer_size: 20480
  learning_rate: 0.0003
  beta: 0.005
  epsilon: 0.2
  lambd: 0.95
```

$\tau$

```
reward_signals:
  extrinsic:
    gamma: 0.995
    strength: 1.0
```

$\tau$: low values correspond to relying more on the current value estimate (which can be high bias), and high values correspond to relying more on the actual rewards received in the environment (which can be high variance).

$\gamma$ : In situations when the agent should be acting in the present in order to prepare for rewards in the distant future, this value should be large. In cases when rewards are more immediate, it can be smaller. Must be strictly smaller than 1.