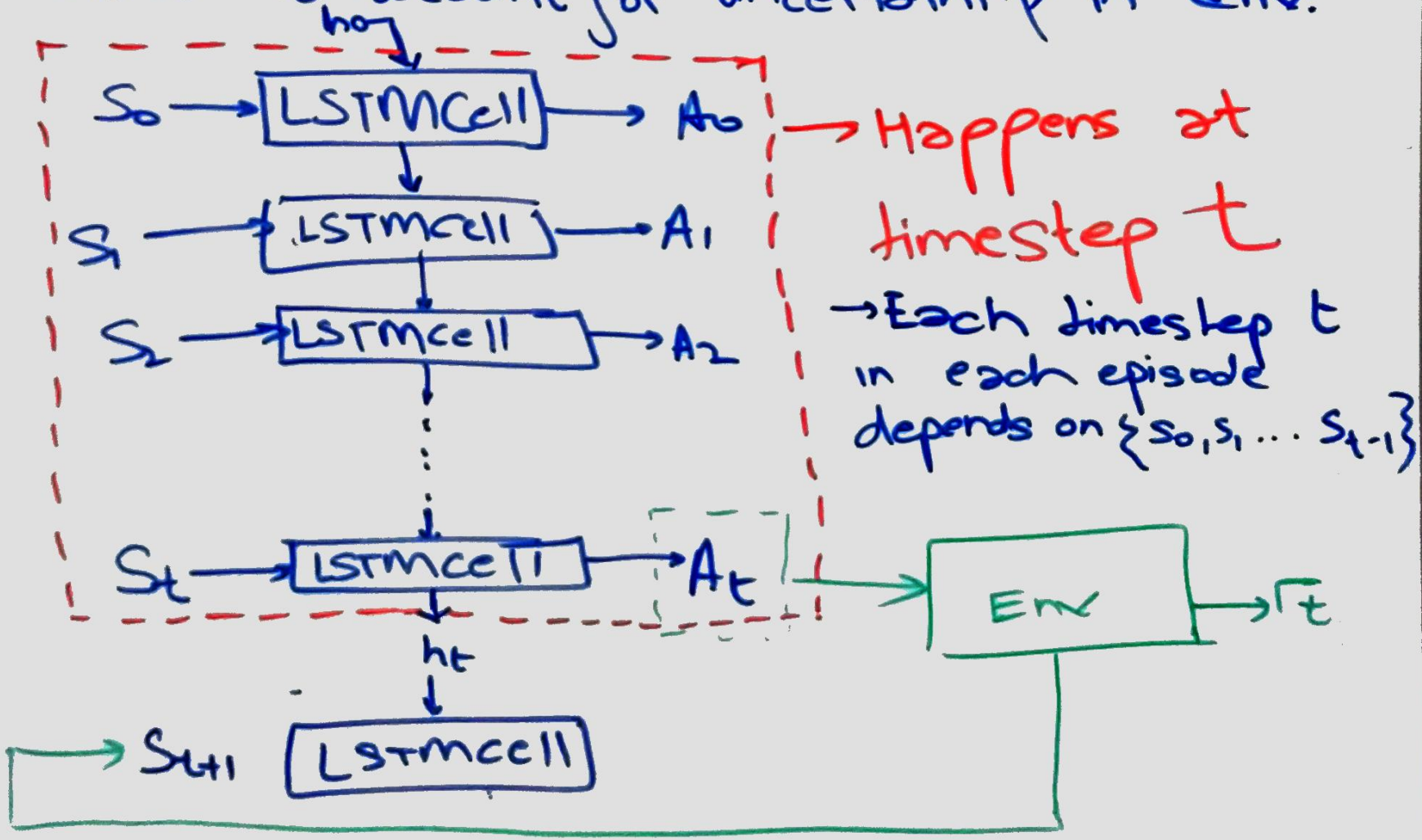
1. Will the oscillation state be considered a part of action? - osc state goes as input to the network operation for the next step.

2. Try using only complex MLP output. No mean & bias.

3. Impact of different types of experience replays and their biological significance.

4. Impact of how reward is calculated on learning & Impact of RNN actor/critic.

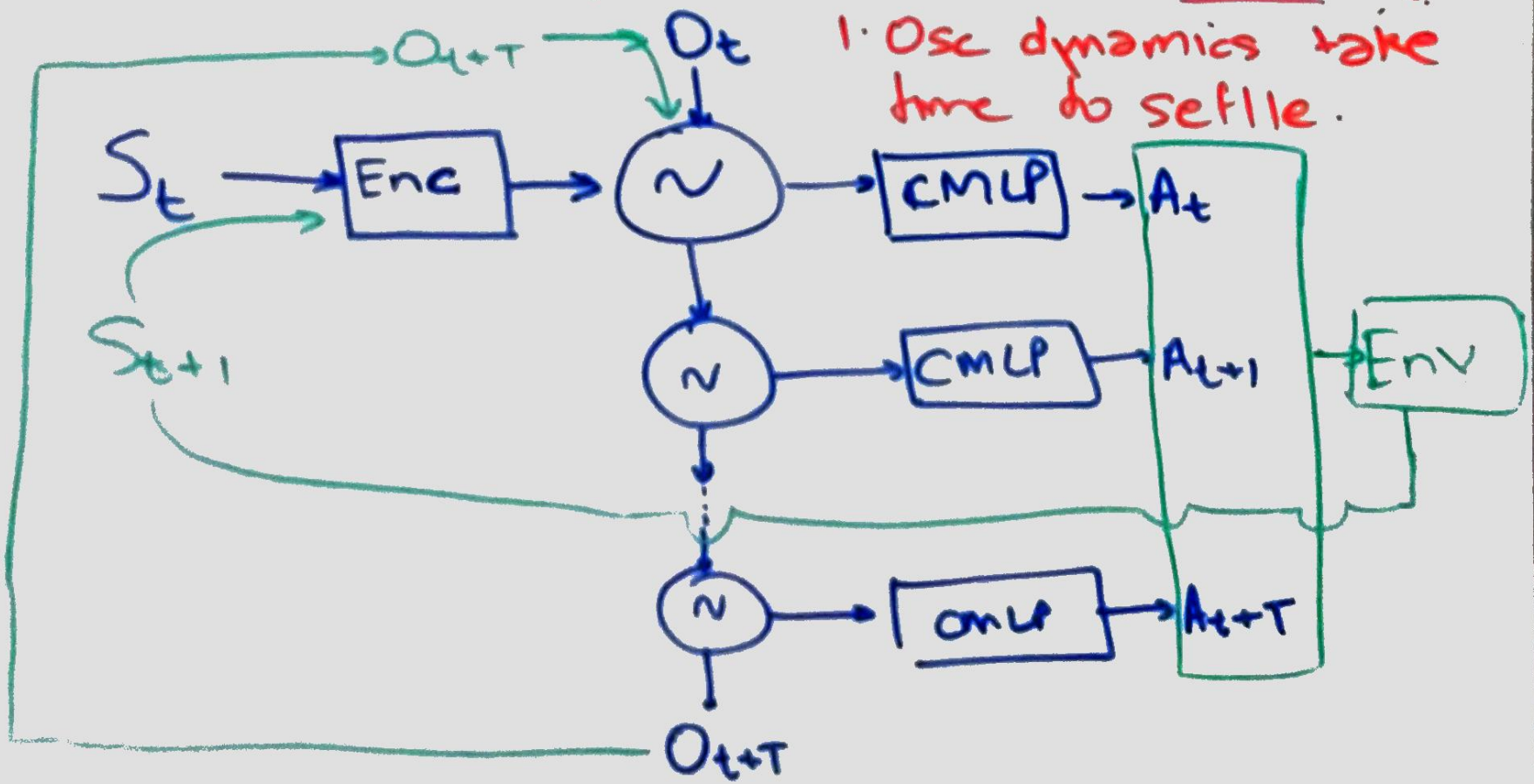
→ Use of history of states instead of a single state to account for uncertainity in Env.



→ Happens at timestep t

→ Each timestep t in each episode depends on $\{s_0, s_1 \dots s_{t-1}\}$

Our network has a recurrent state for oscillators. Can this state and oscillator emulate memory as seen in the RDPG??



1. Osc dynamics take time to settle.

→ The issue of oscillator transient time can be solved by hard assignment instead of updation by euler's method.

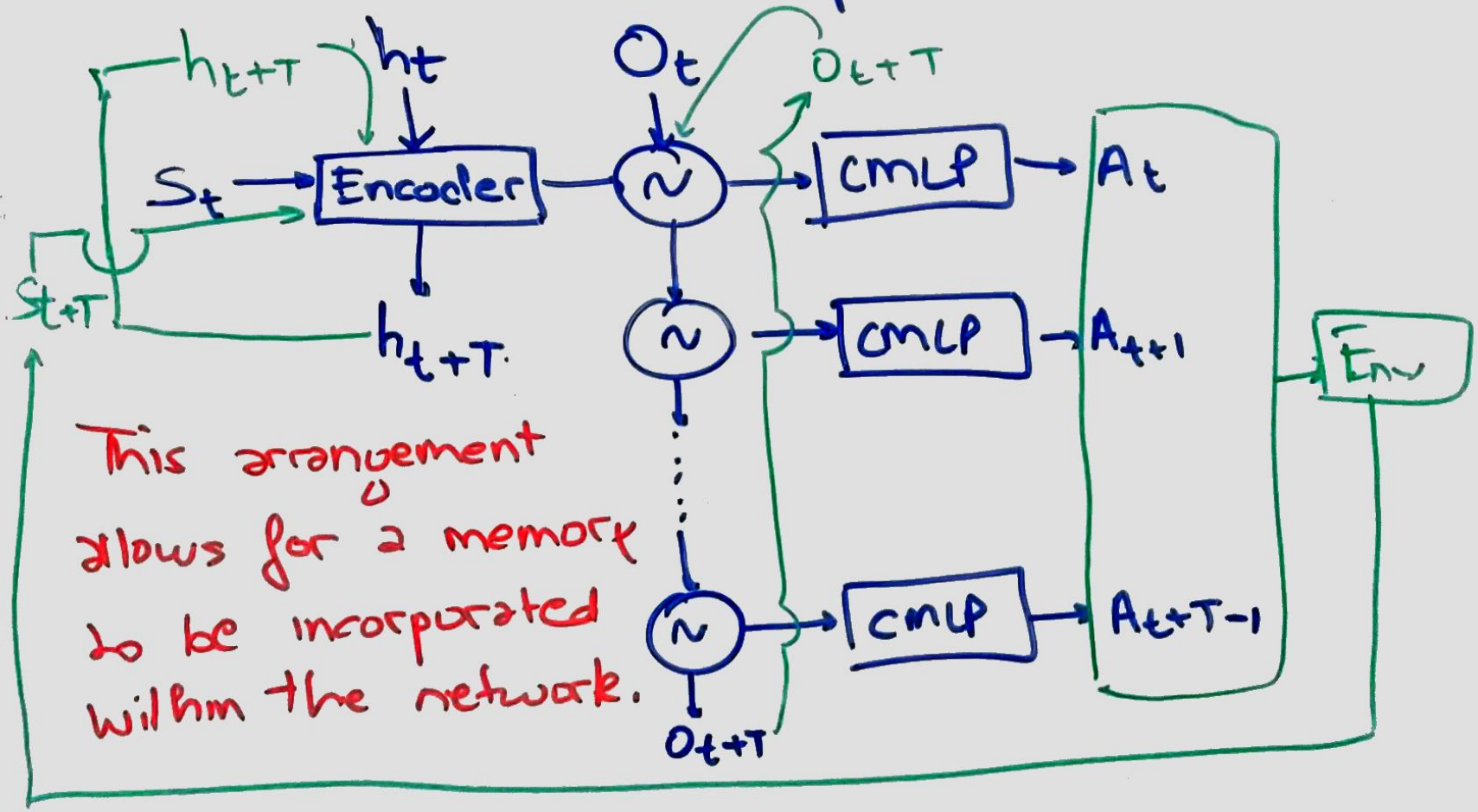
→ The oscillator state has no information about the environment or previous states. It is represents the current position of the oscillator variable values in the phase plane.

→ Having no means of looking at previous states makes the optimization problem equivalent to a random search (This may be evident from the similar performance of DDPG & ARS)

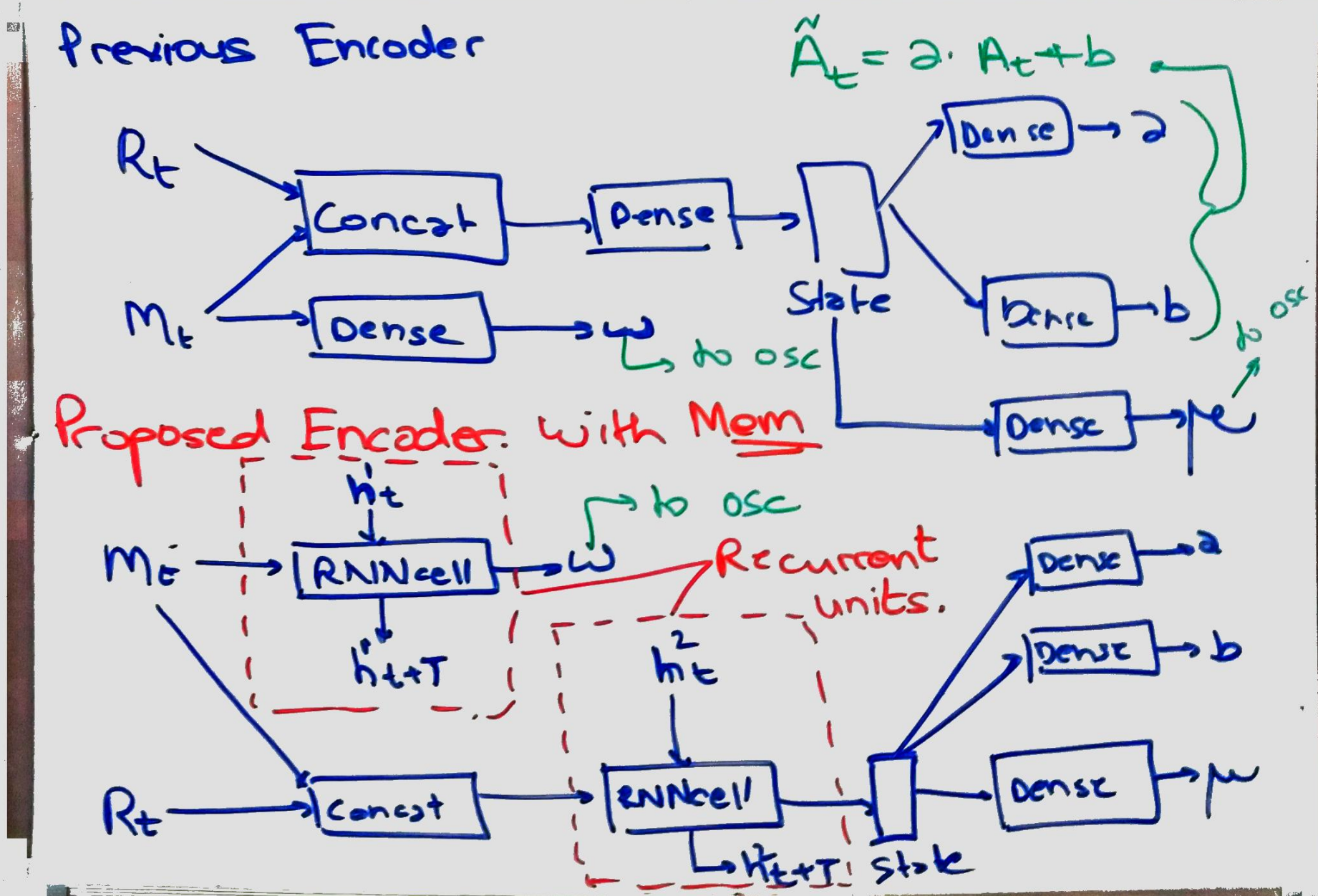
→ Relating the RDPH architecture to the Flip-Flop Osc Network from the previous mechno → The flipflop is a memory (recurrent cell), operating at a different timescale than the oscillator.

→ The flow of information from previous state to current state allow for the significant learning ability of the FFOsc network.

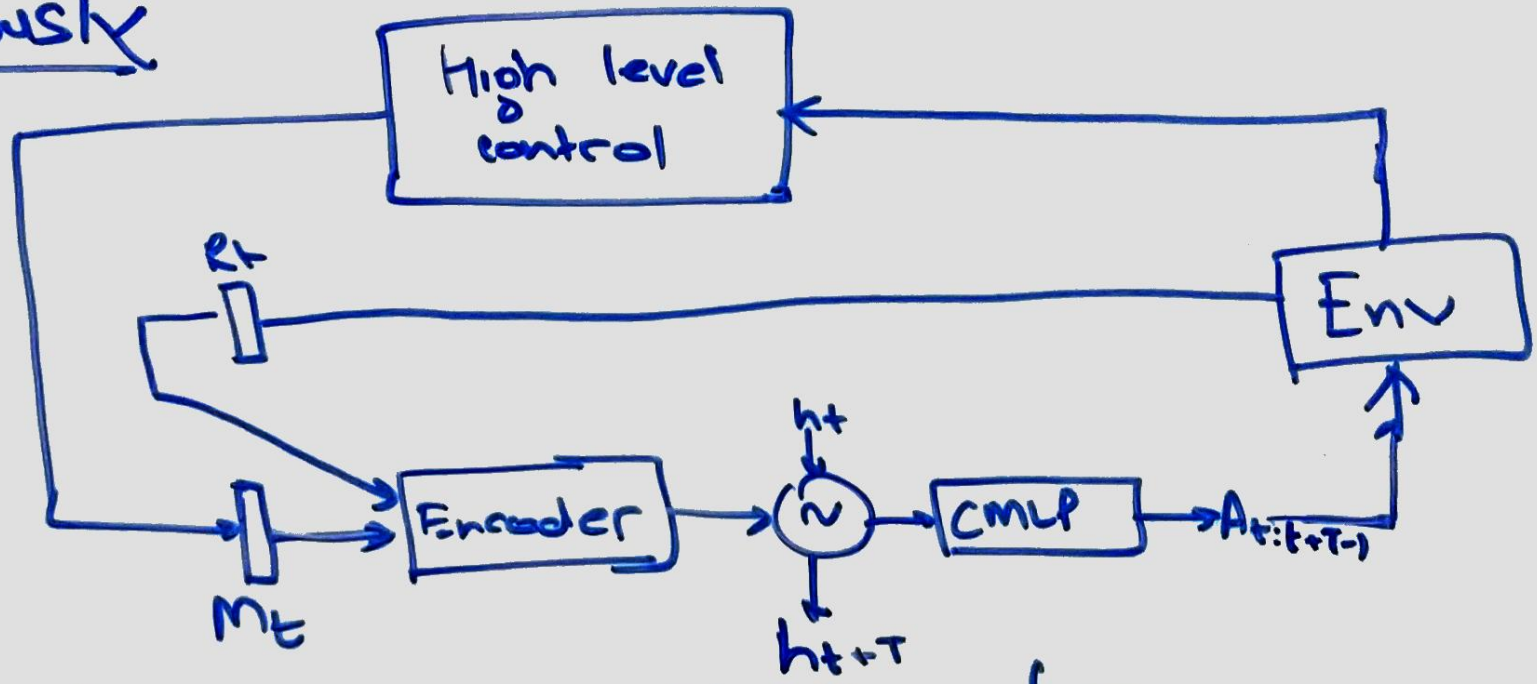
→ The use of the following architecture should alleviate our problem



This arrangement allows for a memory to be incorporated within the network.

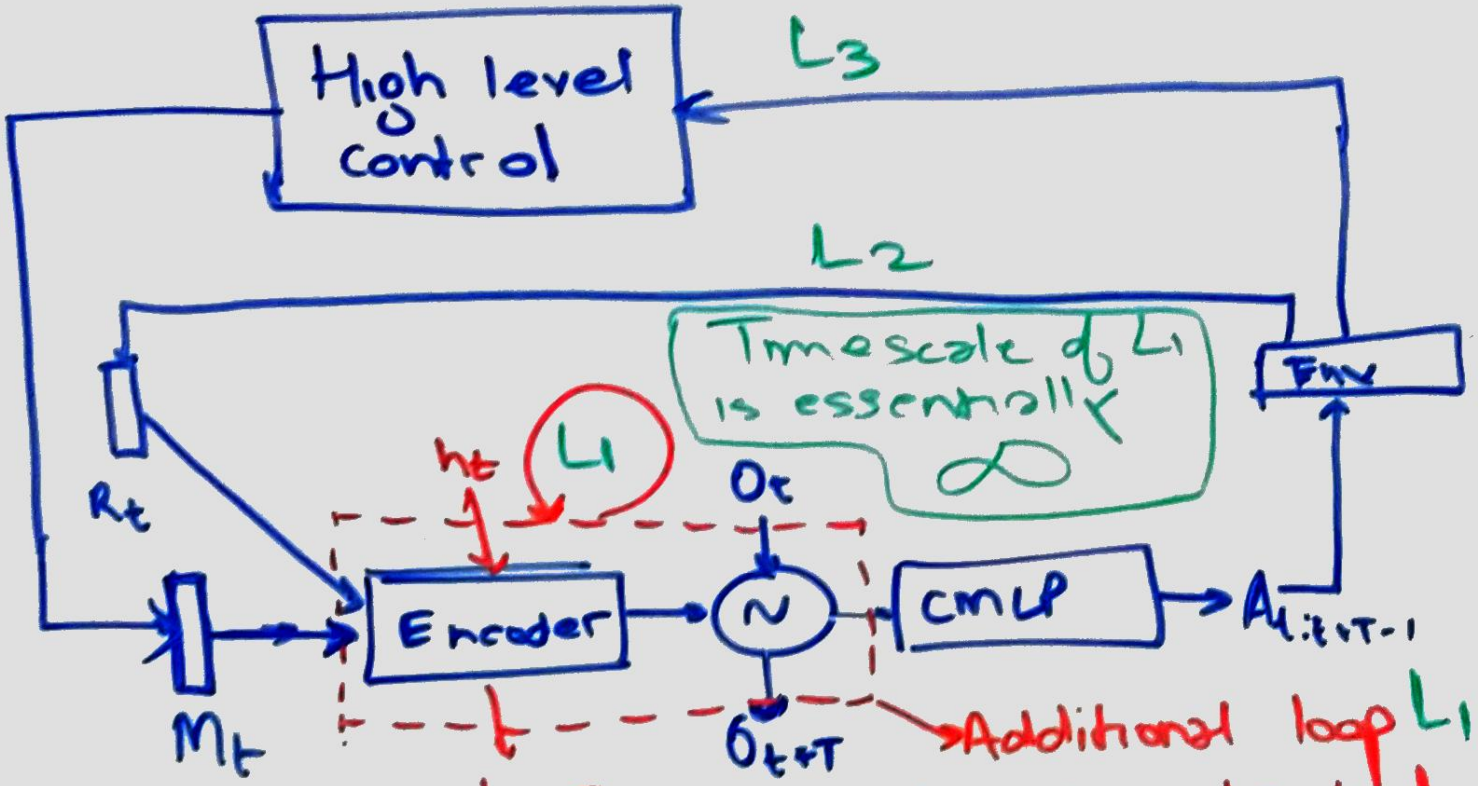

Previous Encoder
$\tilde{A}_t = a \cdot A_t + b$
R_t
M_t
Concat
Dense
Dense
State
Dense
a
Dense
b
Dense
w
to osc
Dense
e
to osc
Proposed Encoder with Mem
h'_t
M_t
RNNcell
w
to osc
Recurrent units.
h'_{t+T}
h^2_t
R_t
Concat
RNNcell
h^2_{t+1}
State
Dense
a
Dense
b
Dense
μ

Previously



High level control would allow for sophisticated behaviour like obstacle avoidance, object capture etc.

RDPG



→ L_2 keeps track of motion within steps

→ L_3 takes care of sophisticated behaviour.