1703.01161 - FeUdal Networks for Hierarchical Reinforcement Learning

- **Yunqiu Xu**

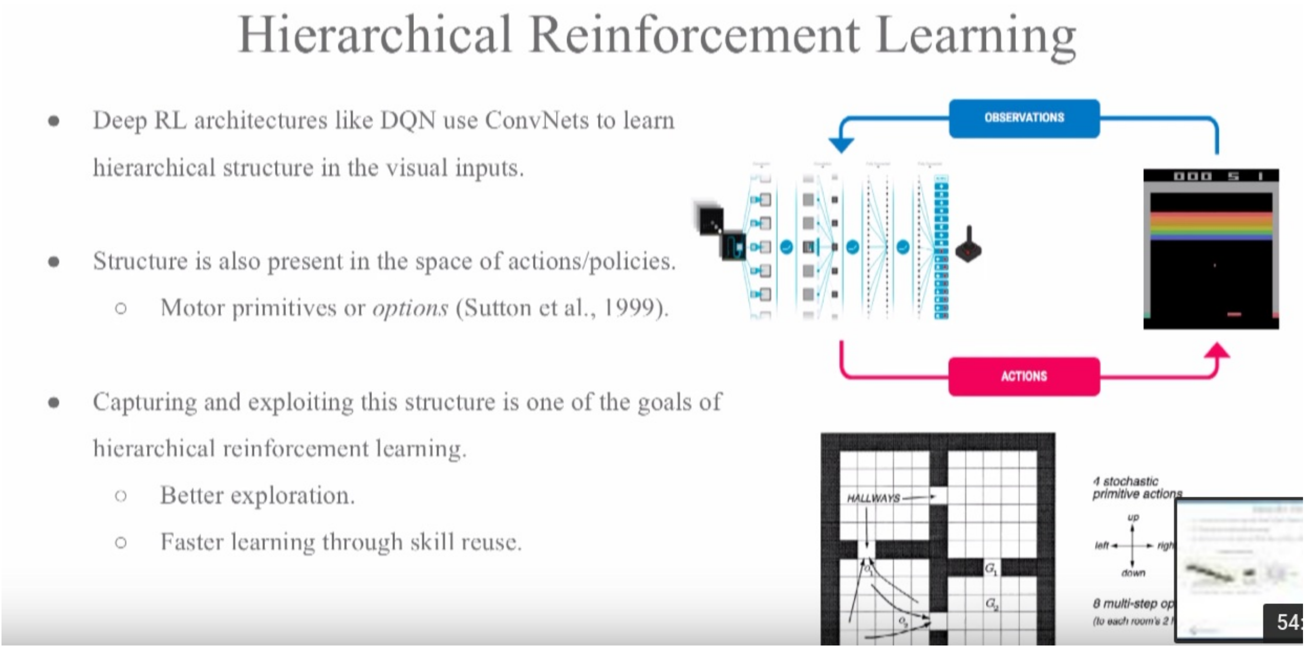
---

1. Introduction

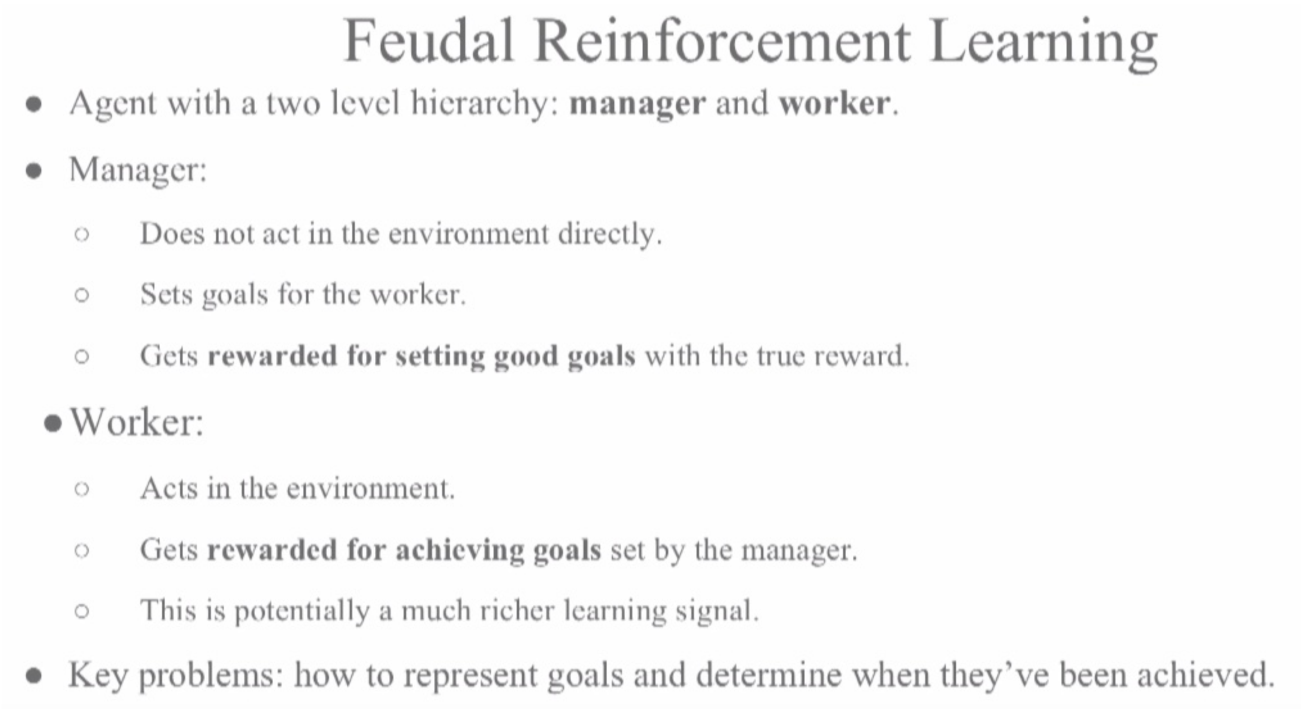
- Challenges:
  - Long-term credit assignment
  - Sparse reward: another solution can be found in 1707.05300 - Reverse Curriculum Generation for Reinforcement Learning
- Our work
  - Get insight from Feudal reinforcement learning (1993) , generalize its principle
  - End-to-end differentiable neural network with two levels of hierarchy: Manager and Worker
  - **Manager network** :
    - operates at a lower temporal resolution
    - produces a meaningful and explicit goal from a latent state-space
    - select latent goals for Worker, try to maximise **extrinsic reward**
  - **Worker network** :
    - operates at a higher temporal resolution
    - follow the goals by an intrinsic reward
    - produces primitive actions, try to maximise **intrinsic reward**
  - No gradients are propagated between Manager and Worker $\rightarrow$ **Manager receives learning signal from the environment alone**
- Advantage:
  - Facilitate very long timescale credit assignment
  - Encourage the emergence of sub-policies associated with different goals set by the Manager

2. Related Work

- Hierarchical RL:



- Feudal RL by Dayan and Hinton, 1993: treat Worker as sub-policy



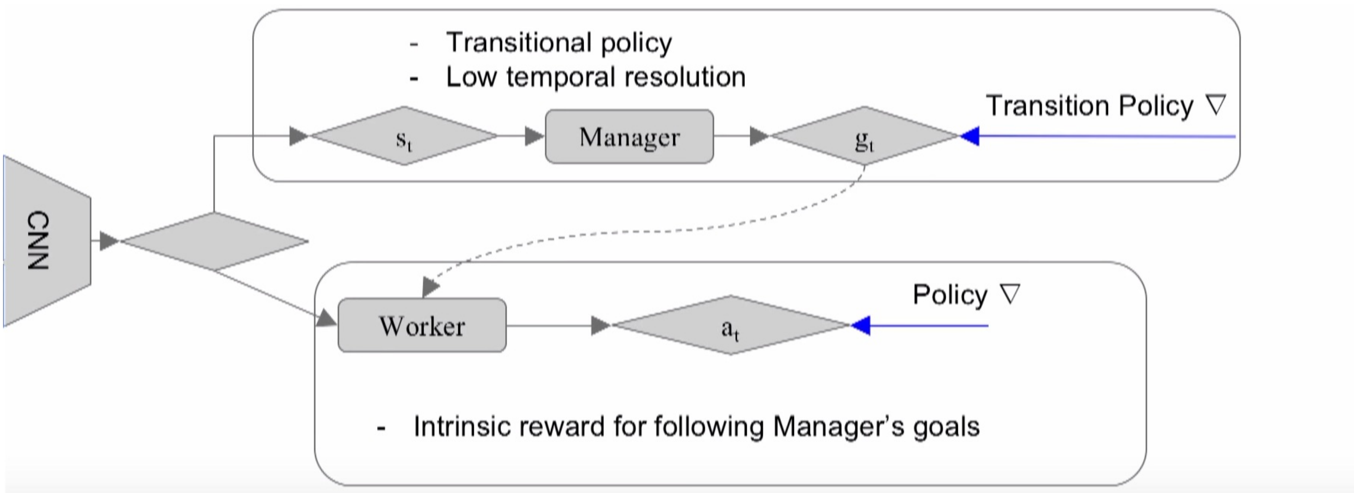
- Combine DL with predefined sub-goals:
  - 1604.07255 - A Deep Hierarchical Approach to Lifelong Learning in Minecraft
  - 1604.06057 - Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation

- However sub-goal discovery was not addressed
- Some non-hierarchical state-of-the-art on Montezuma’ s Revenge: orthogonal to H-DRL, can be combined together
  - 1606.01868 - Unifying Count-Based Exploration and Intrinsic Motivation
  - 1611.05397 - Reinforcement Learning with Unsupervised Auxiliary Tasks

3. The Model

3.1 Overview of Forward dynamics

- Both Manager and Worker are recurrent
  - Manager:
    - Receieve state(transformed by CNN) from environment
    - Compute latent state s_t
    - Output a goal g_t
  - How to train Manager to get g_t : transition policy gradient
  - Worker:
    - Receieve both state from environment and goal set by the Manager
    - Produce actions
  - How to train Worker : intrinsic reward to produce actions that cause these goal directions to be achieved



- $Eq. 1$:

- Manager and worker share a perceptual module
- Take an observation from env x_t
- Compute a shared intermediate representation z_t
- $f^{percept}$: CNN

- $Eq. 2$:
  - compute the implicit states for Manager to compute goals
  - f^{Mspace} : FC layer
- $Eq. 3$:
  - Compute the internal states h^M and goals for Manager
  - f^{Mrnn} : dilated LSTM
    - Operates at lower temporal resolution than the data stream
    - More details : [Yu & Koltun, 2015, Multi-Scale Context Aggregation by Dilated Convolutions]
- $Eq. 4$:
  - Goal embedding
  - $w_t \in R^k$ is embedding vector mapped from g_t via a linear projection ϕ
  - During implementation:
    - $k = 16$
    - ϵ : prob at each step to emit a random goal
- $Eq. 5$:
  - h^W : internal states for Worker
  - $U_t \in R^{|a| \times k}$ is the output of worker, an embedding for action
  - f^{Wrnn} : standard LSTM
- $Eq.6$: Policy π_t is computed from the combination of w_t and U_t

$$z_t = f^{\text{percept}}(x_t) \quad (1)$$

$$s_t = f^{Mspace}(z_t) \quad (2)$$

$$h_t^M, \hat{g}_t = f^{Mrnn}(s_t, h_{t-1}^M); g_t = \hat{g}_t/||\hat{g}_t||; \quad (3)$$

$$w_t = \phi(\sum_{i=t-c}^{t} g_i) \quad (4)$$

$$h^W, U_t = f^{Wrnn}(z_t, h_{t-1}^W) \quad (5)$$

$$\pi_t = SoftMax(U_t w_t) \quad (6)$$

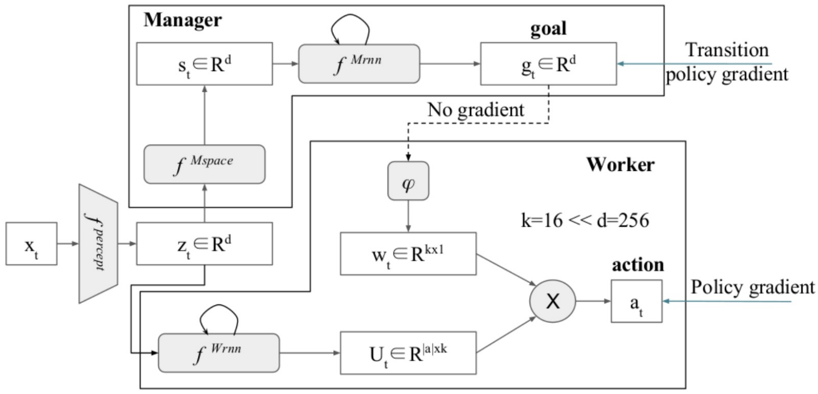


Figure 1. The schematic illustration of FuN (section 3)

- **Why the output of Manager (goal) always influence the final policy**
  - ϕ has no bias $\rightarrow$ never produce constant non-zero vector
  - So the setup will never ignore Manager's input

3.2 Learning $\rightarrow$ Train Worker

- FuN is fully differentiable $\rightarrow$ we can train it end-to-end using pg operating on actions taken by Worker
- **Why we do not propagate gradient between Manager and Worker**
  - g_t need to have semantic meaning $\rightarrow$ define the temporal resolution of the Manager
  - If we train Manager by gradients coming from the Worker
  - Manager's goals g will not have **semantic meaning** but internal latent variables
  - **注意如果这里 g 不存在 semantic meaning 的话在后面计算 Worker 的 intrinsic reward 的时候就会有问题**
- So what we do instead:
  - Independently train Manager to predict advantageous directions (transitions) in state space
  - Then intrinsically reward the Worker to follow these directions
- Thus the update rule of Manager can be:

$$\nabla_{g_t} = A_t^M \nabla_\theta d_{cos}(s_{t+c} - s_t, g_t(\theta)) \quad (7)$$

- $A_t^M = R_t - V_t^M(x_t, \theta)$: Manager's advantage function
- $d_{cos}(s_{t+c} - s_t, g_t(\theta))$: the cosine similarity of $s_{t+c} - s_t$ and $g_t(\theta)$
- **The dependence of s on θ is ignored here to avoid trival solutions**
- ∇_{g_t} can be seen as "advantageous direction"

- The intrinsic reward of Worker can be:

$$r_t^I = \frac{1}{c}\sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_{t-i}) \quad (8)$$

  - So here we need to give g_{t-i} semantic meaning
  - **此处存疑, semantic meaning of g_t 到底啥意思**

- **My understanding** :
  - 这里 g_t 不仅仅是一个position或者 reward value, 我们将其 理解为到达目标的方向
  - 两个state相减即为agent的前进方向, 因此我们要尽可能最大化其与 g_t 的余弦相似度
  - 即让你当前走的方向和到达目标的方向尽可能一致

- Compared with old version (Dayan & Hinton 1993), we add an intrinsic reward for following the goals, but retaining the environment reward as well
  - This is similar to **regularization**
  - Worker is trained to maximize $R_t + \alpha R_t^I$
  - Method to train Worker : A2C

$$\nabla_{\pi_t} = A_t^D \nabla_\theta log\pi(a_t|x_t;\theta) \quad (9)$$

  - Here advantage function can be transformed as

$$A_t^D = R_t + \alpha R_t^I - V_t^D(x_t;\theta)$$

3.3 Transition Policy Gradients → Train Manager

- The update of Manager is with respect to a model of Worker's behavior
- **Assumption : sub-policies are fixed duration behaviors**
- $o_t = \mu(s_t, \theta)$: Master need to learn high level policy to select which subpolicy to use
- $\pi^{TP}(s_{t+c}|s_t) = p(s_{t+c}|s_t, o_t)$: each sub-policy can be represented as transition distribution, here s_{t+c} means end states of this sub-policy
- So transition policy can be seen as the distribution over end states given start states

$$\pi^{TP}(s_{t+c}|s_t) = p(s_{t+c}|s_t, \mu(s_t, \theta))$$

- Then we can use PG to train π^{TP}

$$\nabla_\theta \pi_t^{TP} = E[(R_t - V(s_t))\nabla_\theta logp(s_{t+c}|s_t, \mu(s_t, \theta))] \quad (10)$$

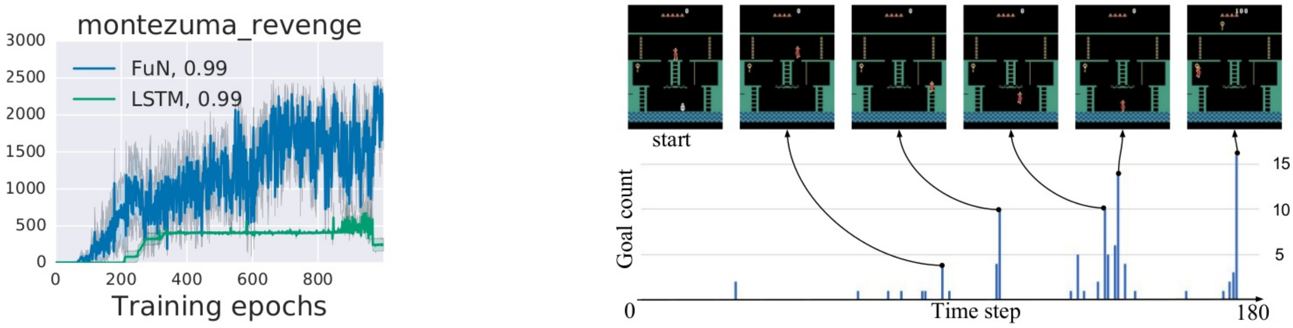
- **Why we just need to use end state distribution of sub-policies**
  - Worker may follow a complex trajectory, and it's hard to compute PG by learning from these trajectories
  - If we know the end states of trajectories, we can skip Worker's behavior, and just follow the PG or predicted transition

4. Experiment

- Goal:
  - Check whether FuN learns non-trival, helpful and interpretable subpolicies and subgoals
  - Validate components of the architecture

4.1 Montezuma's Revenge

- Try to get the key to go out the first room
- For each timestamp, compute latent state s_t and goal g_t
- Then try to find a future state s_f to maximize $d_{cos}(s_f - s_t, g_t) \rightarrow$ make them more similar
- From (a) we can see that FuN needs less states to maximize the goal
- **From (b) FuN learns semantically meaningful sub-goals: we can interpret the tall bar as useful "milestones" (e.g. turning right then going down)**



(a) (b)

4.2 Other Atari Games

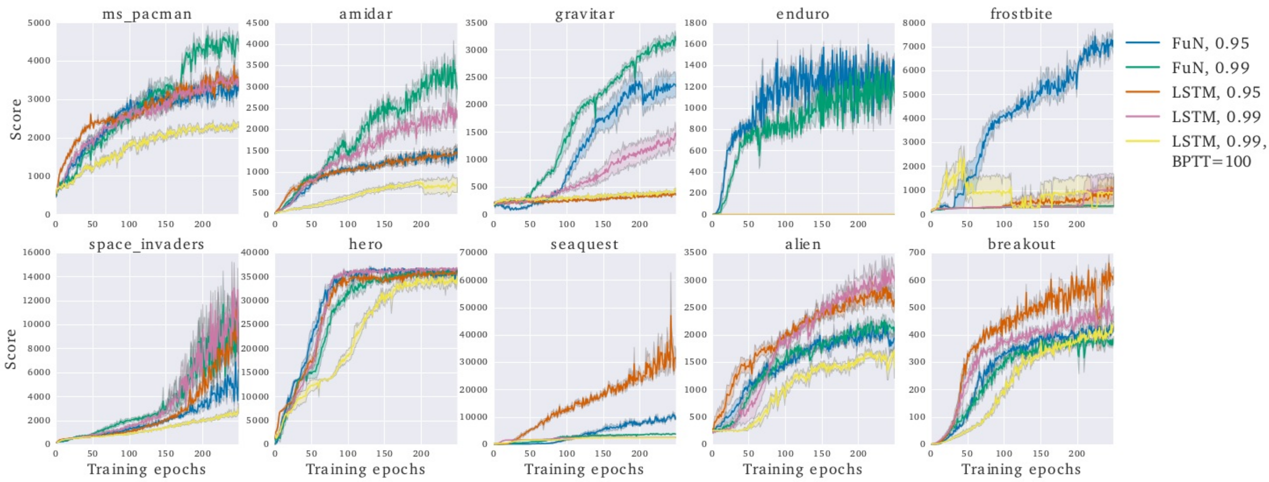
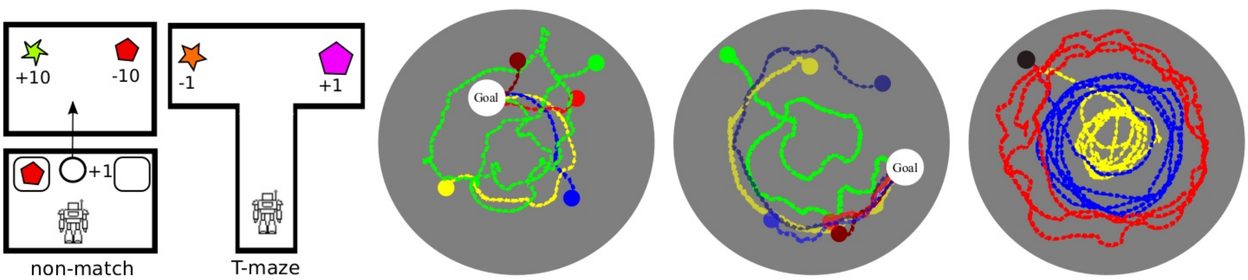
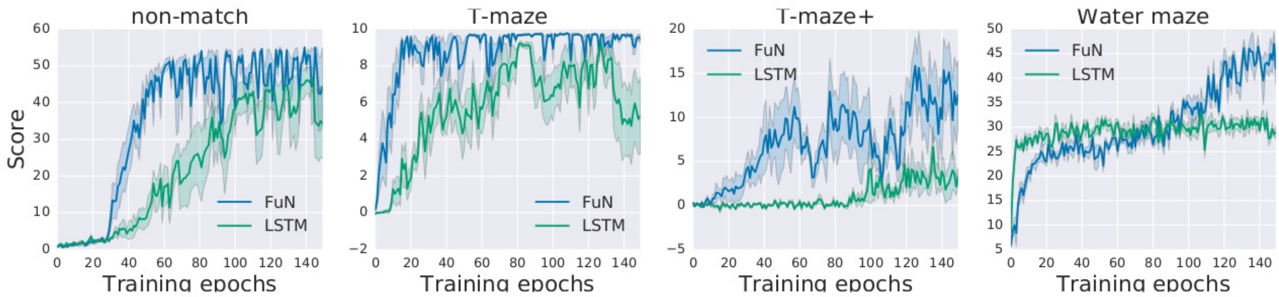


Figure 4. ATARI training curves. Epochs corresponds to a million training steps of an agent. The value is the average per episode score of top 5 agents, according to the final score. We used two different discount factors 0.95 and 0.99.

4.3 Visual memorisation tasks in 3D environment



(a) (b)



4.4 Ablative Analysis

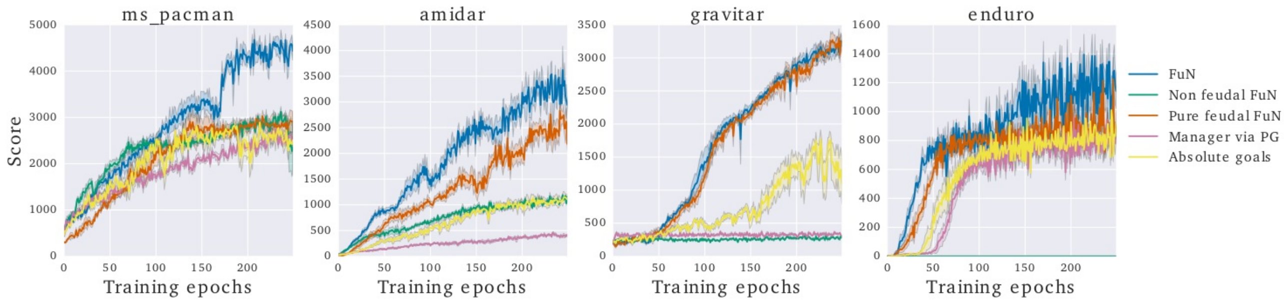


Figure 8. Ablative analysis

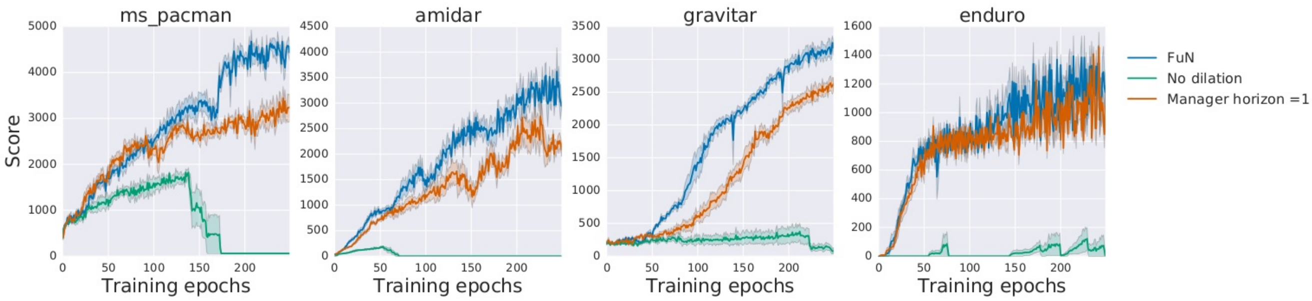


Figure 10. Learning curves for ablations of FuN that investigate influence of dLSTM in the Manager and Managers prediction horizon c. No dilation – FuN trained with a regular LSTM in the Manager; Manager horizon =1 – FuN trained with $c = 1$.

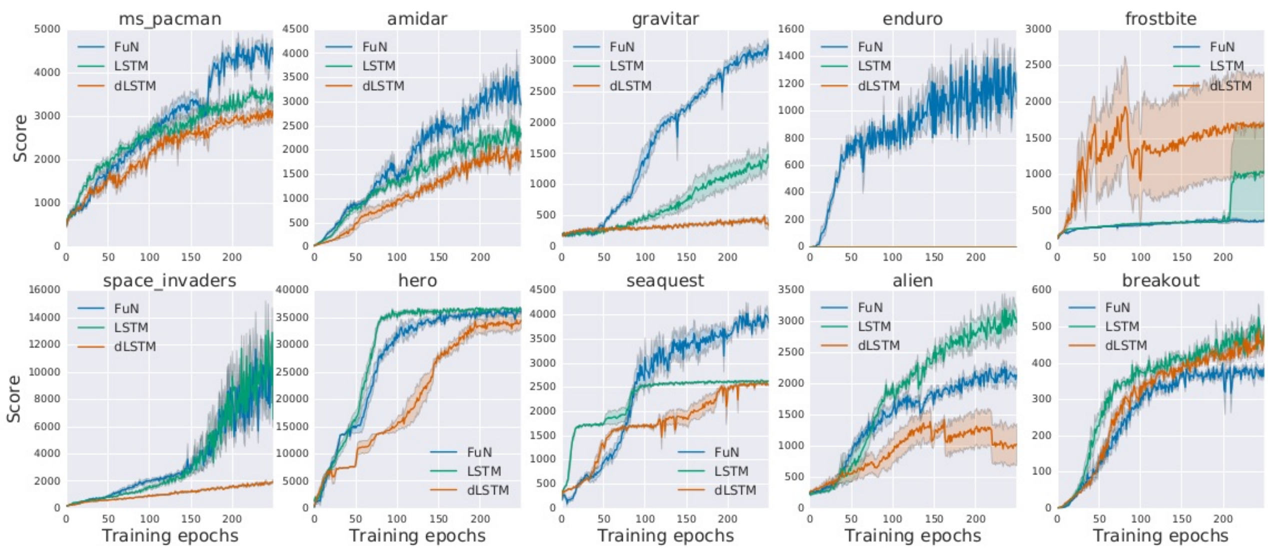
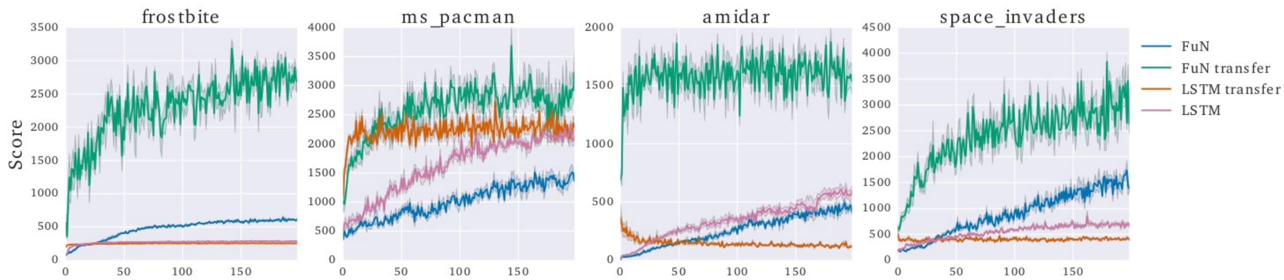


Figure 12. Learning curves for dLSTM based agent with LSTM and FuN for comparison.

- Action Repeat Transfer



5. Discussion and Future Work

- How we formulate sub-goals
  - Set sub-goals as directions in latent state space
  - If followed, sub-goals will be translated as meaningful behavioral primitives
- Future work:
  - Deeper hierarchies: 这个可以看下 DDO 和 DDCO
  - Transfer / multi-task Learning: 这个可以结合下 MIL 和 MLSH, 用 meta-learning 训练合适的子任务, 然后对于新的任务只需要训练 master (i.e. 在合适的时间选择合适的子任务进行执行)