FeUdal Networks for Hierarchical Reinforcement Learning

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, Koray Kavukcuoglu: DeepMind

Rene Bidart (rbbidart@uwaterloo.ca)

CS885

June 22, 2018

Why do we care about HRL?

Reinforcement Learning is hard!

- Long time horizons and sparse rewards are problematic for current methods
- Many of the methods we use are not intuitively appealing
- Look to human decision process for inspiration

Hierarchies

How do we make decisions?

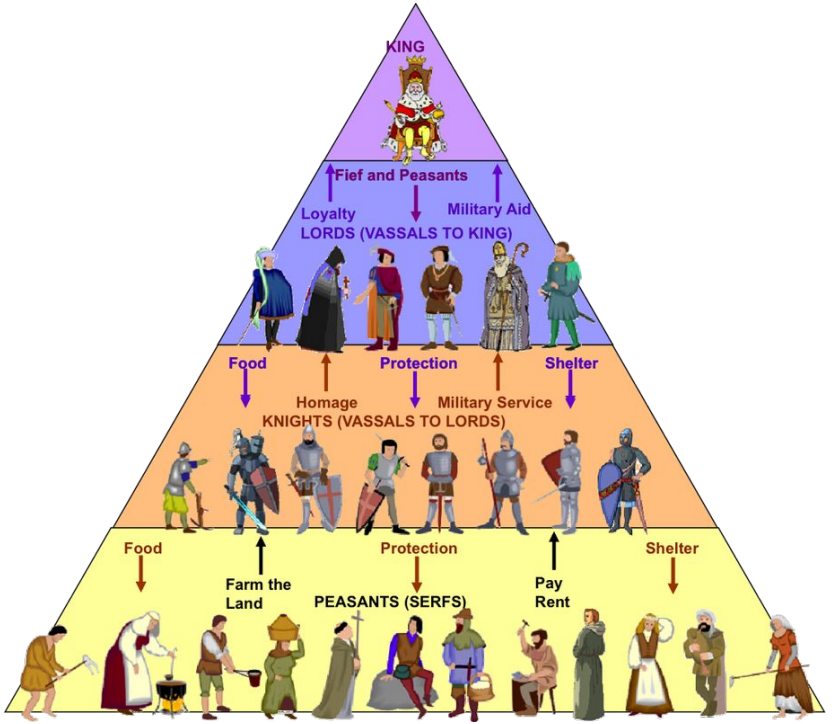
- If we are hungry, do we reason in terms of small muscle movements?
- To play the guitar, do we randomly jitter our fingers until we play a song?

No, we reason using **hierarchies of abstraction.**

- We already use conv nets to learn hierarchical structure in images. Why not use hierarchical structure in policies?

Feudalism

- Governance system in Europe in middle ages
- Extremely hierarchical, based on ownership of property
- Higher level people have control over the lower levels, but not over people many layers lower



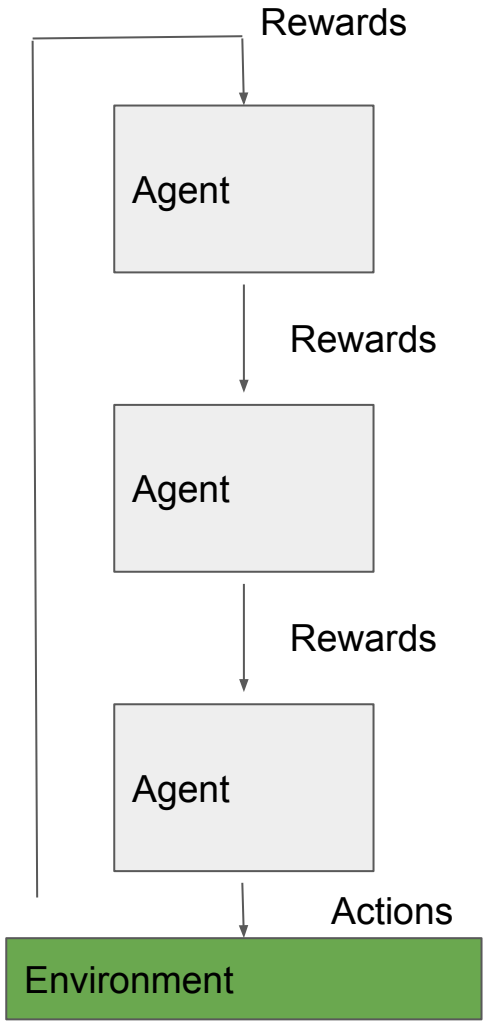
Feudal RL (1993)

Reward Hiding:

- Managers reward sub-managers for satisfying their commands, not through an external reward
- Managers have absolute control

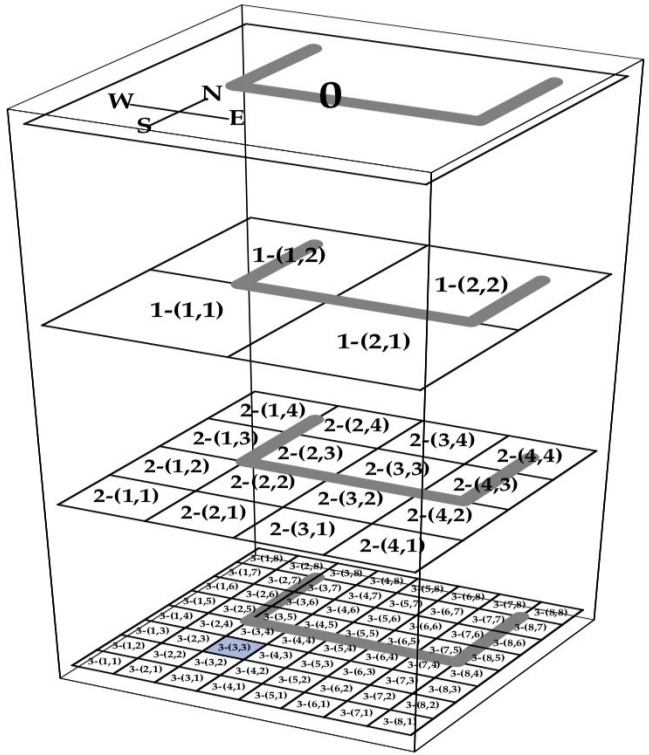
Information Hiding

- Observe world at different resolutions
- Managers don’t know what happens at a other levels of the hierarchy



Feudal RL (1993)

- Q-learning
- Used to solve a simple maze task
- Didn’t generate good results on more complex or less obviously hierarchical problems



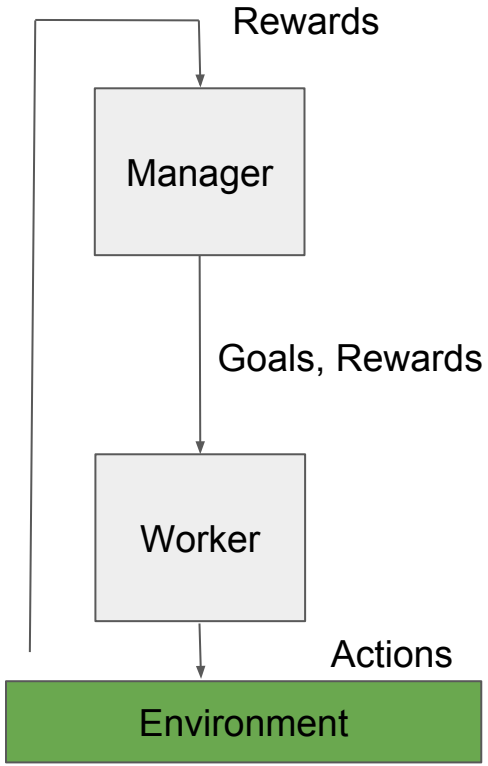
FeUdal Networks (2017): Overview

Manager

- Sets directional goals for the worker
- Rewarded by environment
- Does not directly act in environment

Worker

- Higher temporal resolution
- Reward for achieving manager's goals
- Produces primitive actions in environment



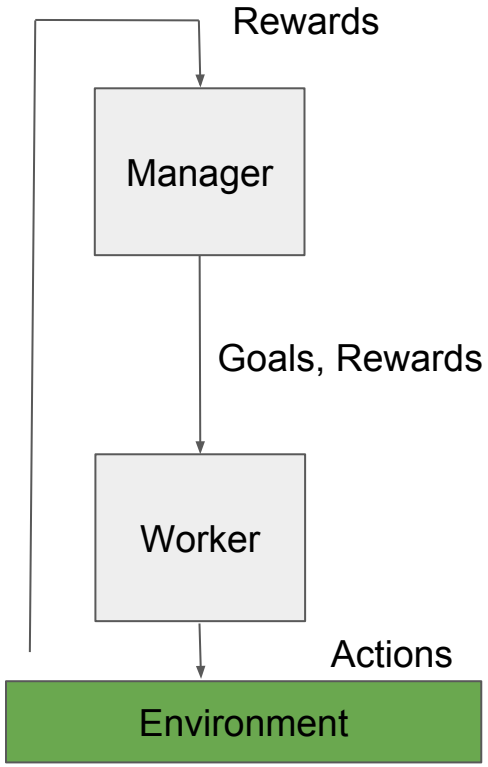
FeUdal Networks (2017): Overview

Architecture

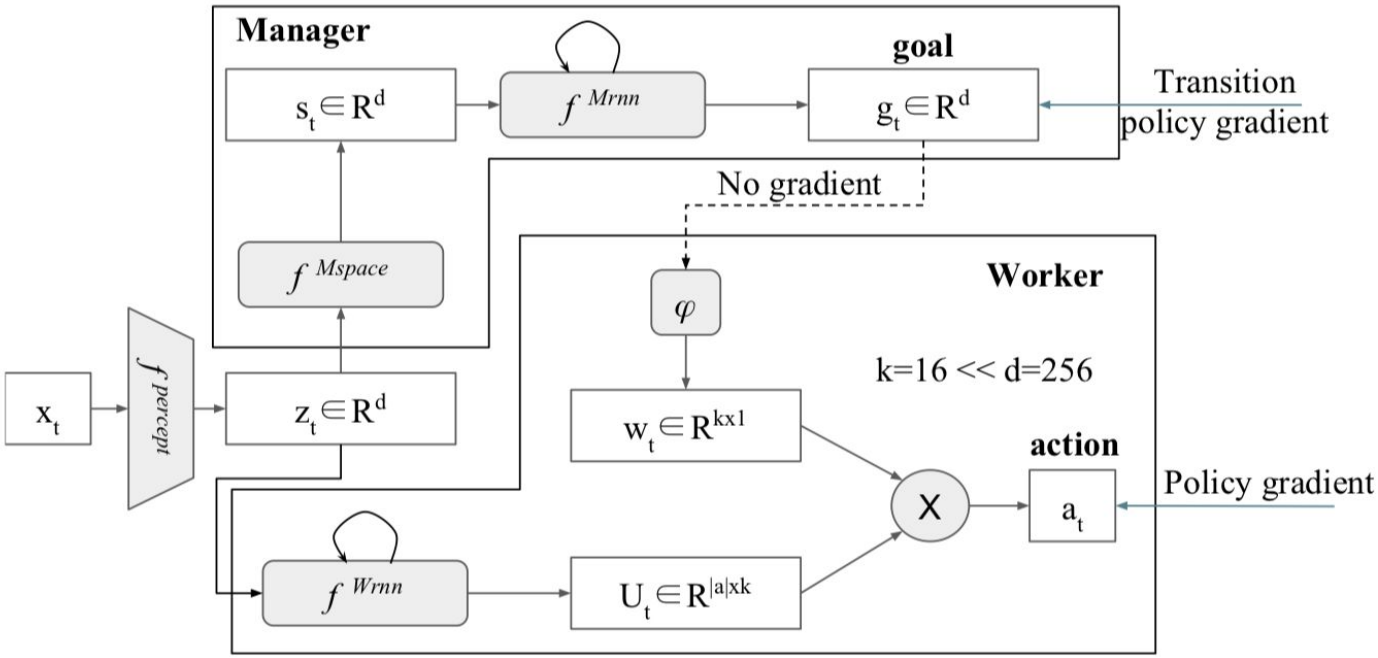
- Both worker and manager share a state embedding
- Both worker and manager use RNNs

Goals

- Manager produces directional goals for worker in latent space
- Trained using novel transition policy gradient



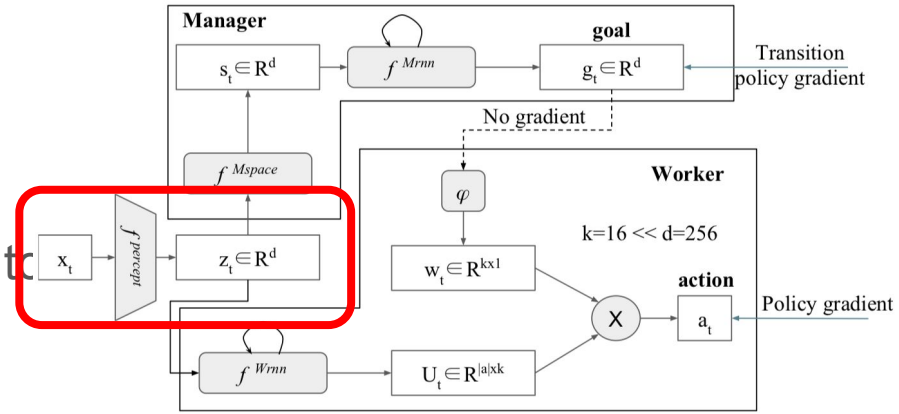
FeUdal Network: Details



FeUdal Network

Shared Dense Embedding

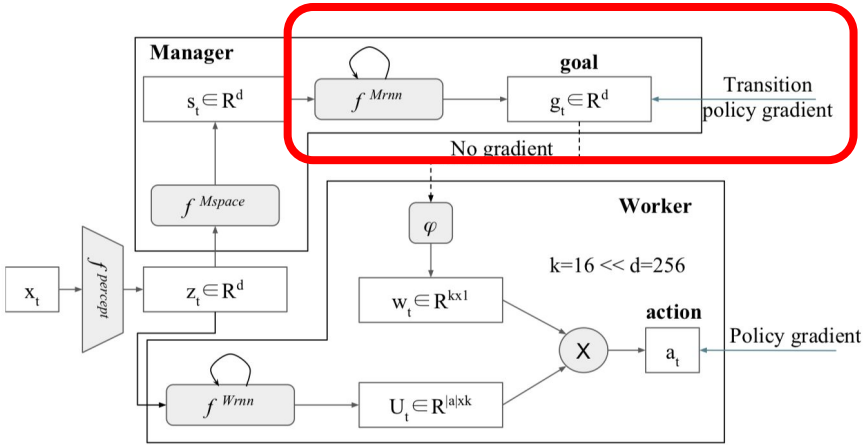
- Embedding of input state
- Used by both worker and manager t produce goal and action
- CNN
  - 16 8x8 filters
  - 32 4x4 filters
  - 256 fully connected
  - ReLU



FeUdal Network

Manager: Goal embedding

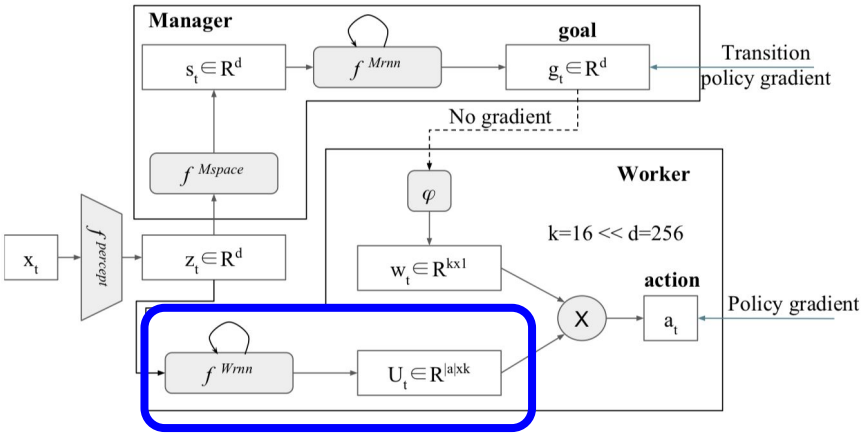
- Lower Temporal Resolution, goals summed over last 10 time steps
- Uses dilated LSTM
- Goal is in low-dimensional space, not environment
- Trained using transition policy gradient



FeUdal Network

Worker: Action Embedding

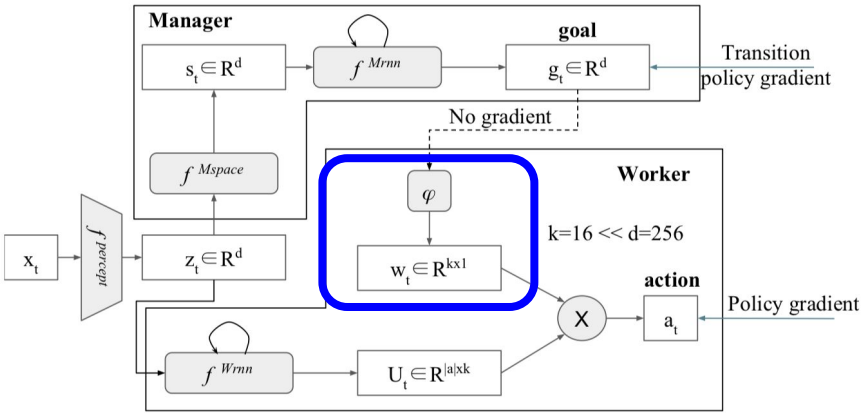
- LSTM on shared embedding
- Embedding U matrix:
  - Rows: actions [a]
  - Columns : embedding dimension [k]



FeUdal Network

Goal embedding: Worker

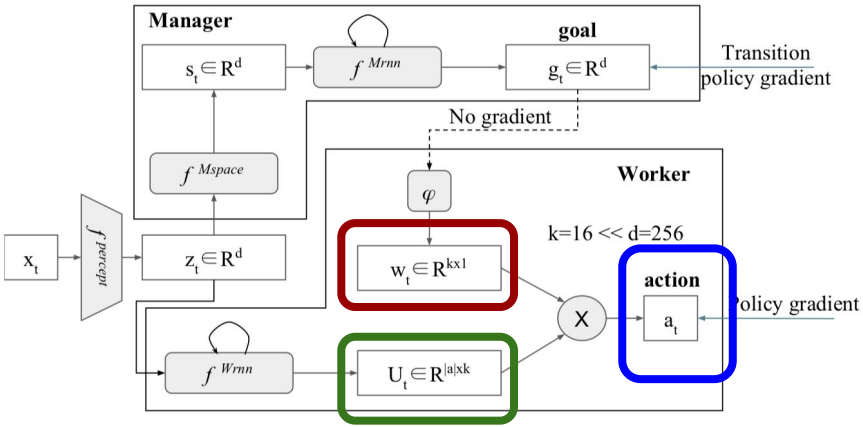
- Compress manager's goal to dim k using linear transformation - ϕ
- Same dim as action embedding
- Linear transformation with no bias
  - Can't produce a 0 vector
  - Can't ignore the manager's input, so manager's goal will influence final policy



FeUdal Network

Action: Worker

- Product of action embedding matrix (U) with goal embedding (w)
- Produces a distribution over actions
- Action = softmax(U*w)



Training Manager: Transition Policy Gradient

- Worker's goal is directional, rather than absolute
- Instead of increasing the probability of an action, we shift the direction of the goal

Actor-critic:

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

Value function from internal critic:

$$A_t^M = R_t - V_t^M(x_t, \theta)$$

Training: Worker's Intrinsic Reward

- Intrinsic reward is based on if the worker follows the correct direction

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

Training: Worker's Intrinsic Reward

Actor-Critic:

$$\nabla \pi_t = A_t^D \nabla_\theta \log \pi(a_t|x_t;\theta)$$

Reward isn't truly hierarchical

- They use weighted sum of intrinsic reward, and environment reward

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

Why directional goals?

Feasibility

- Worker can more easily cause directional shifts, rather than reaching a new location in state space

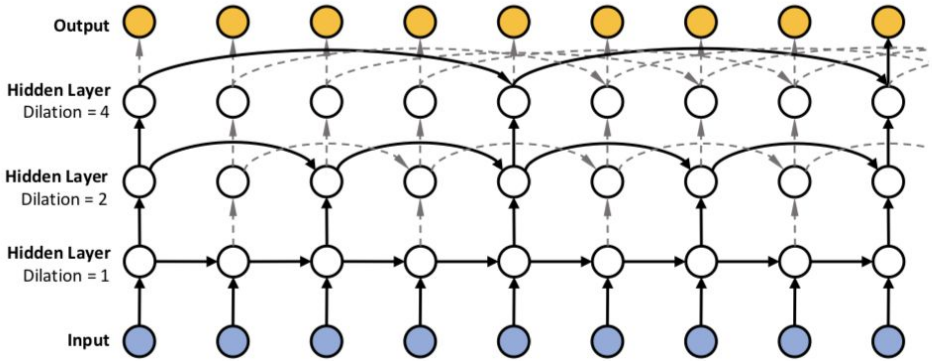
Structural Generalization

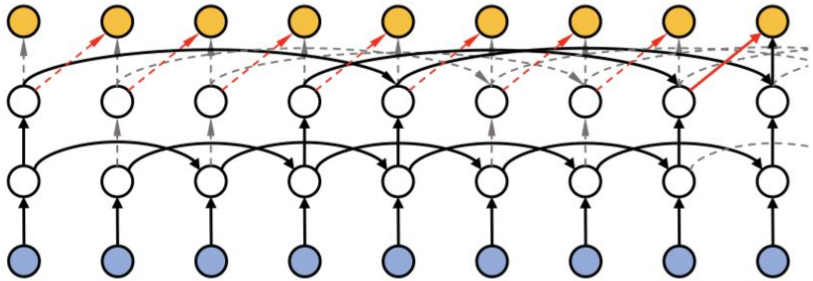
- A single sub goal (direction) can be useful in many different locations in the state space

More Details: Dilated LSTM

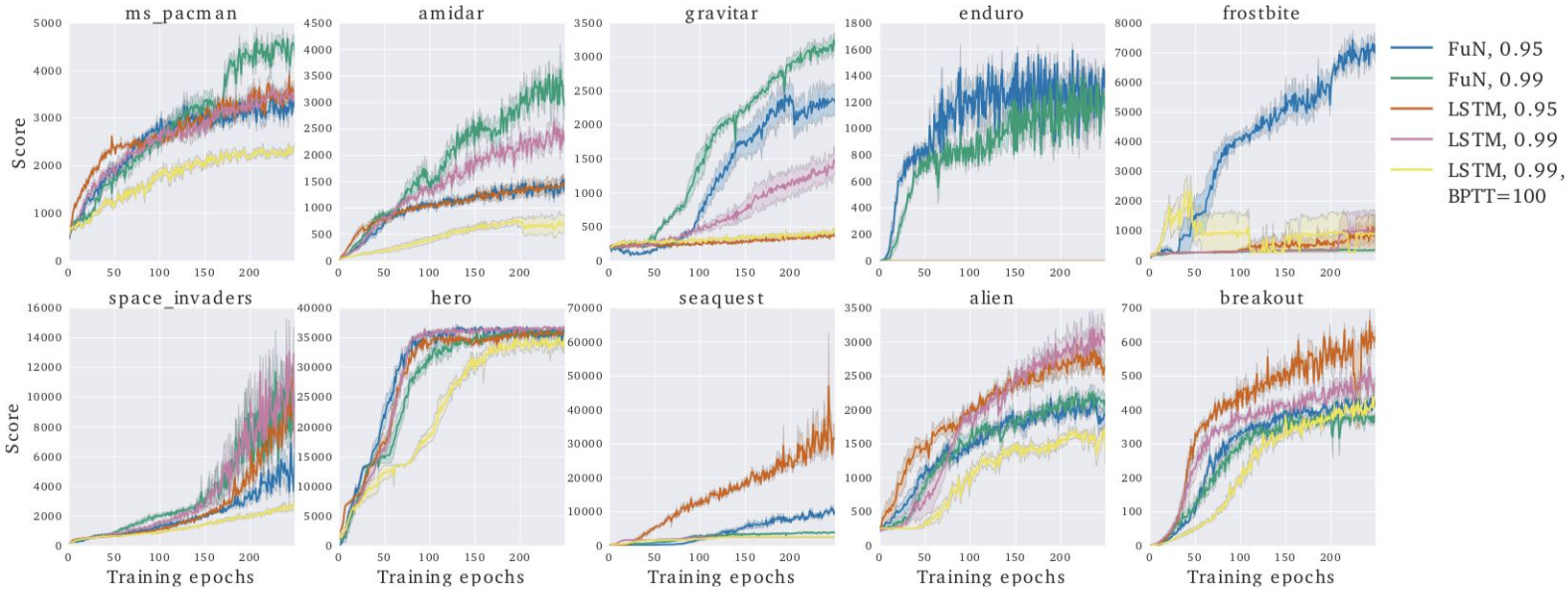
- Better able to preserve memories over long periods
- Output is summed over previous 10 steps
- Specific type of Dilated RNN

Dilated RNN [Chang et al. 2017]:





Results: Atari

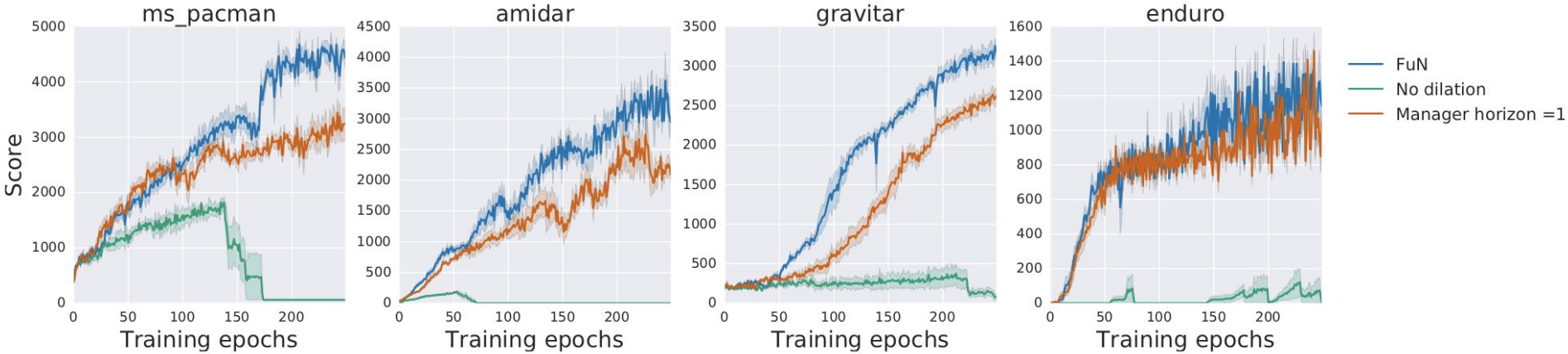


- Outperforms LSTM baseline whenever there are more delayed rewards

Results: Water Maze

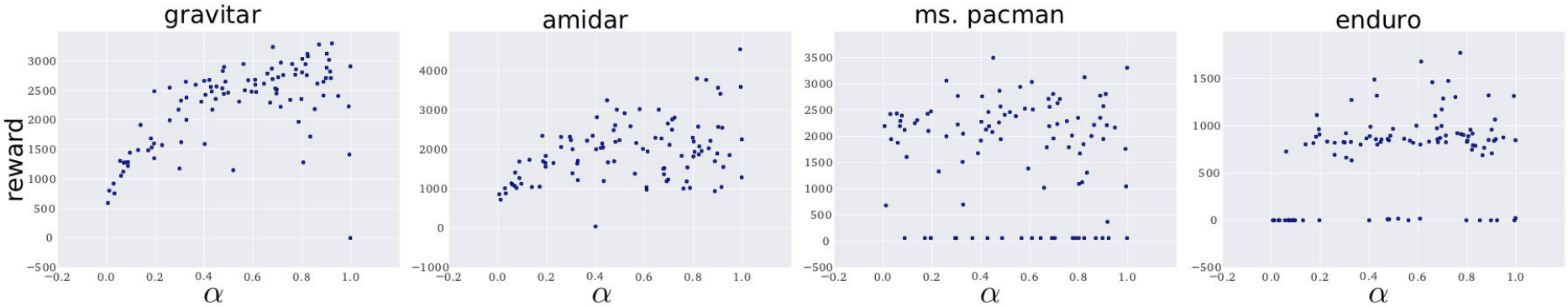
- Circular space with invisible goal, agent must find goal
- Next episode put in a random location, and agent must find goal again
- Left are individual episodes, right visualizes the sub-policies
- Agent learns meaningful sub goals

Results: Temporal Resolution Ablations



- Removing dilations from the LSTM or using full temporal time scale for manager is significantly worse

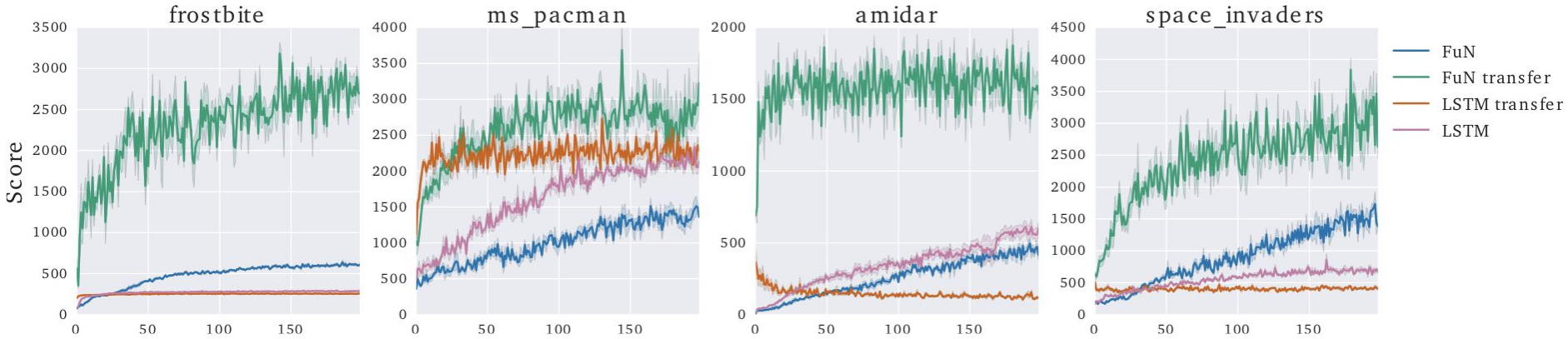
Results: Intrinsic Reward Ablations



- Using only intrinsic reward at right
- Environment reward is not necessary for good performance

Results: Atari Action Repeat Transfer

- One of the goals of HRL was better transfer learning
- Transfer learning with different number of action repeats
- Manager's policy does not depend on how the worker achieves these goals



Summary

- Directional rather than absolute goals are useful
- Dilated LSTM is crucial for high performance
- Improves long-term credit assignment over baselines
- Improves transfer across different action repeats
- Manager’s goals are meaningful low-level behaviors from the worker

Thoughts

- Ablation studies were crucial to get a better idea what is going on - something missing in a lot of DL papers.

- Why does worker produce goal and action embedding, rather than just feeding it into a fully connected network?

Questions?