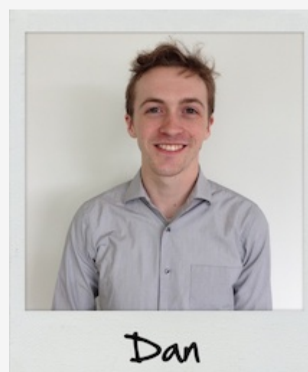
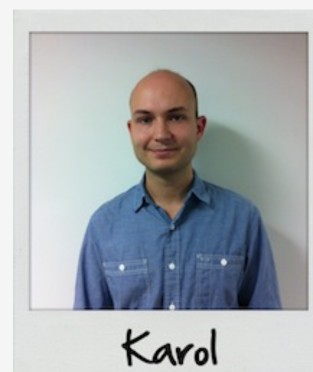
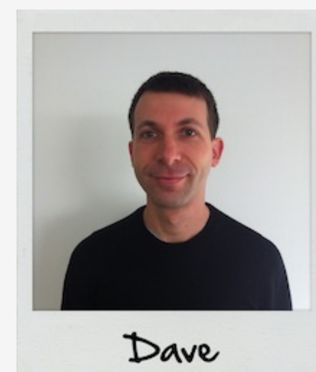
Universal Value Function Approximators

Tom Schaul, Dan Horgan, Karol Gregor, Dave Silver







Motivation

Forecasts about the environment

- = temporally abstract predictions (questions)
- not necessarily related to reward (unsupervised)
- conditioned on a behavior
- (aka GVFs, nexting)
- **many** of them

Why?

- better, richer representations (features)
- decomposition, modularity
- temporally abstract planning, long horizons

Example forecasts

- Hitting the wall
  - if the agent aims for the nearest wall
  - if the agent goes for the door
- Remaining time on battery
  - if the agent stands still
  - if the agent keeps moving
- Luminosity increase
  - if the agent presses the light switch
  - if the agent waits for sunrise

Concretely, for this work:

Subgoal forecasts

- Reaching any of a set of states, then
  - the episode terminates ($\gamma = 0$)
  - and a pseudo-reward of 1 is given
- Various time-horizons induced by γ
- Q-values are for the optimal policy that tries to reach the subgoal (alignment)

Neural networks as function approximators

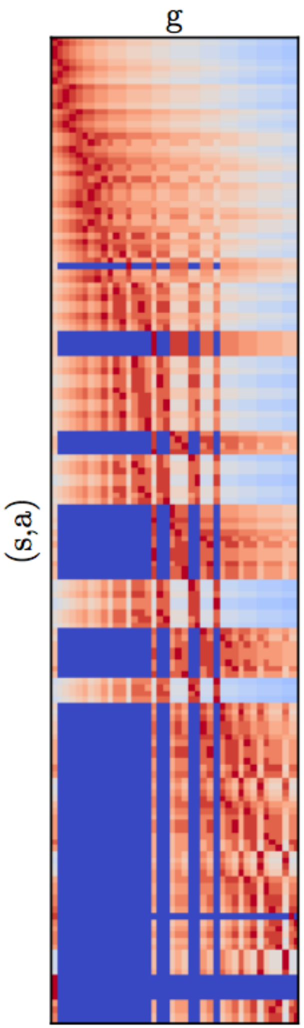
Combinatorial numbers of subgoals

Why?

- because the environment admits tons of predictions
- any of them could be useful for the task

How?

- efficiency
  - sub-linear cost in the number of subgoals
- exploit shared structure in value space
- generalize to similar subgoals

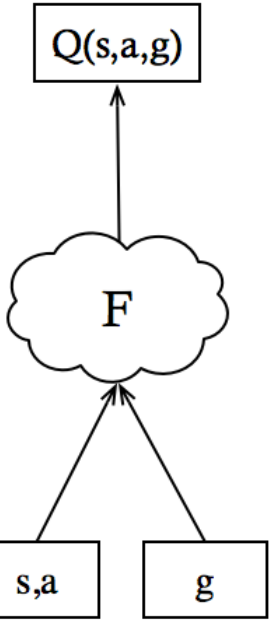


Outline

- Motivation
  - learn values for forecasts
  - efficiently for many subgoals
- Approach
  - new architecture
  - one neat trick
- Results

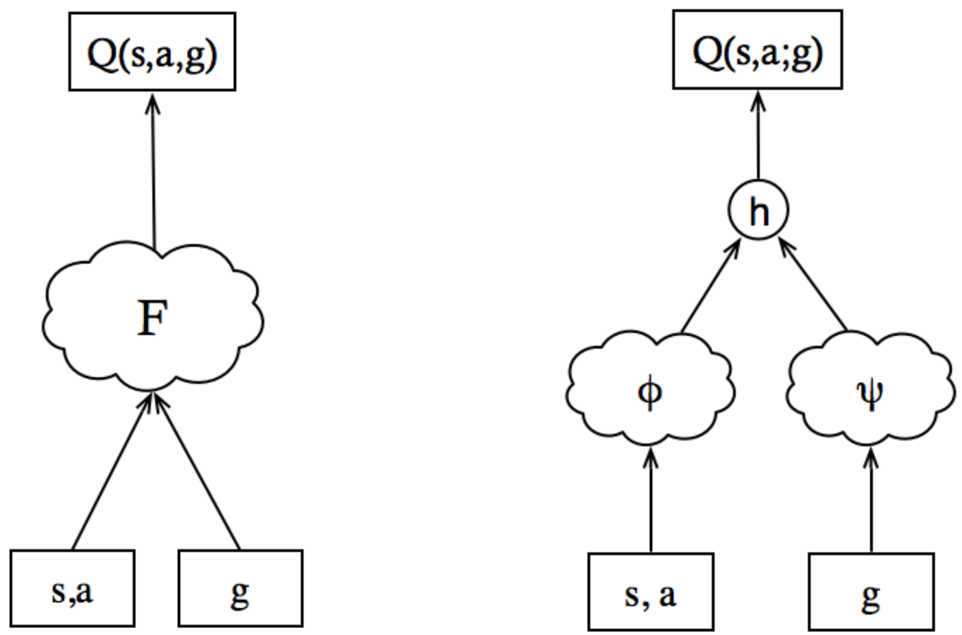
Universal Value Function Approximator

- a single neural network producing Q(s, a; g)
  - for many subgoals g
  - generalize between subgoals
  - compact
- UVFA (“you-fah”)



UVFA architectures

- Vanilla (monolithic)
- Two-stream
  - separate embeddings φ and ψ for states and subgoals
  - Q-values = dot-product of embeddings
  - (works better)



UVFA learning

- Method 1: bootstrapping

$$Q(s_t, a_t, g) \leftarrow \alpha \left(r_g + \gamma_g \max_{a'} Q(s_{t+1}, a', g) \right) + (1 - \alpha)\, Q(s_t, a_t, g)$$

  - some stability issues

- Method 2:
  - built training set of subgoal values
  - train with supervised objective
  - like neuro-fitted Q-learning
  - (works better)

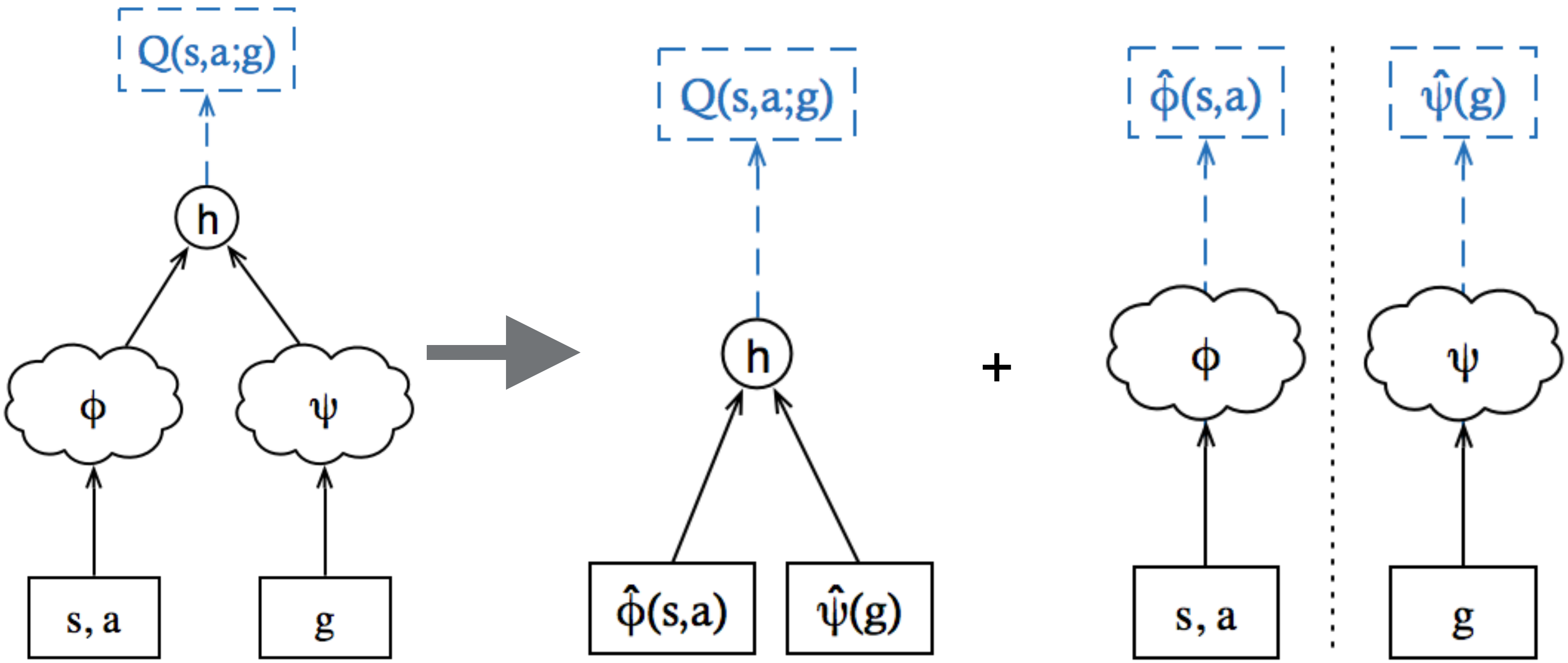
Outline

- Motivation
  - learn values for forecasts
  - efficiently for many subgoals
- Approach
  - new architecture: UVFA
  - one neat trick
- Results

Trick for supervised UVFA learning: FLE

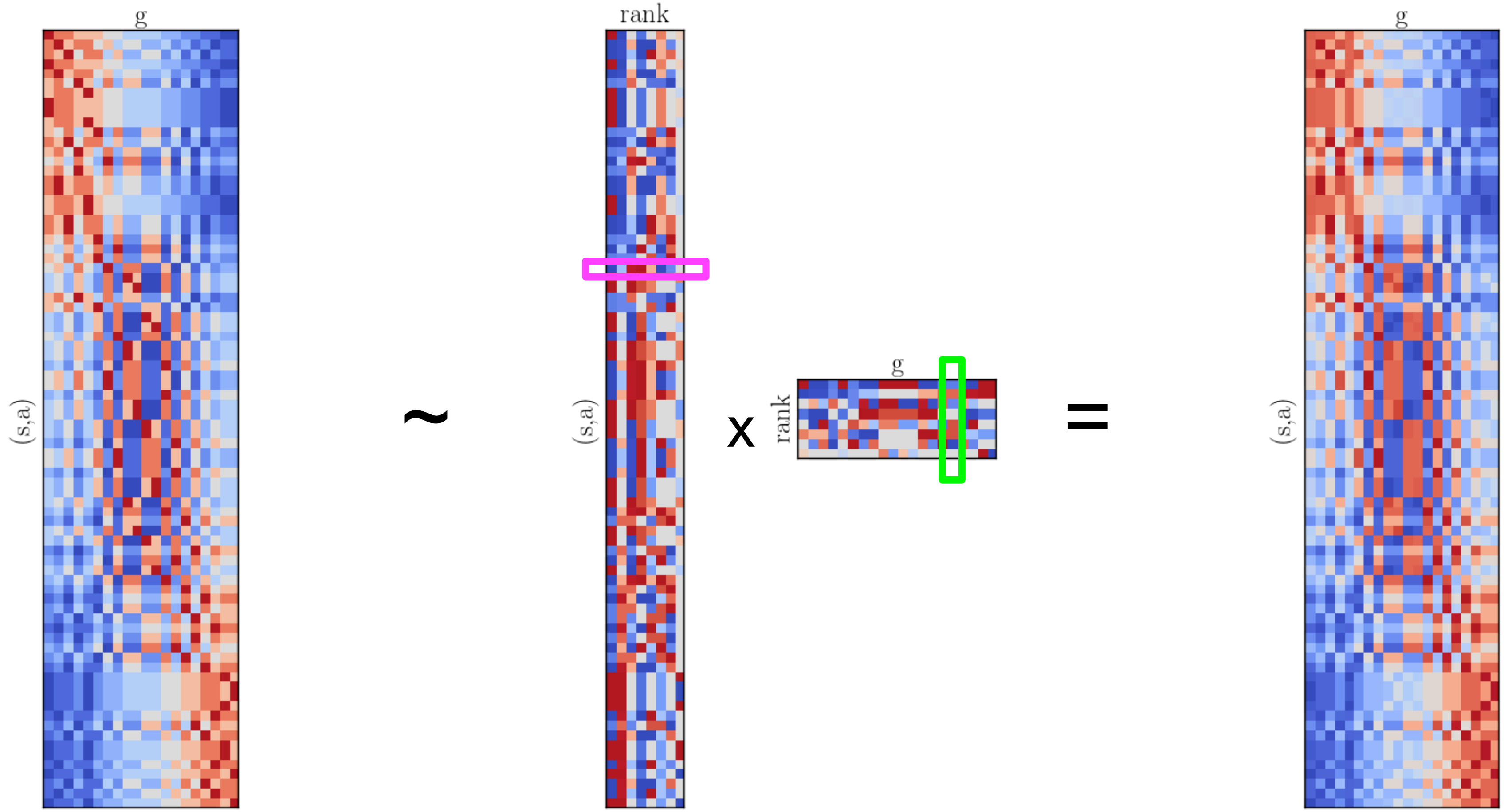
Stage 1: **F**actorize

Stage 2: **L**earn **E**mbeddings



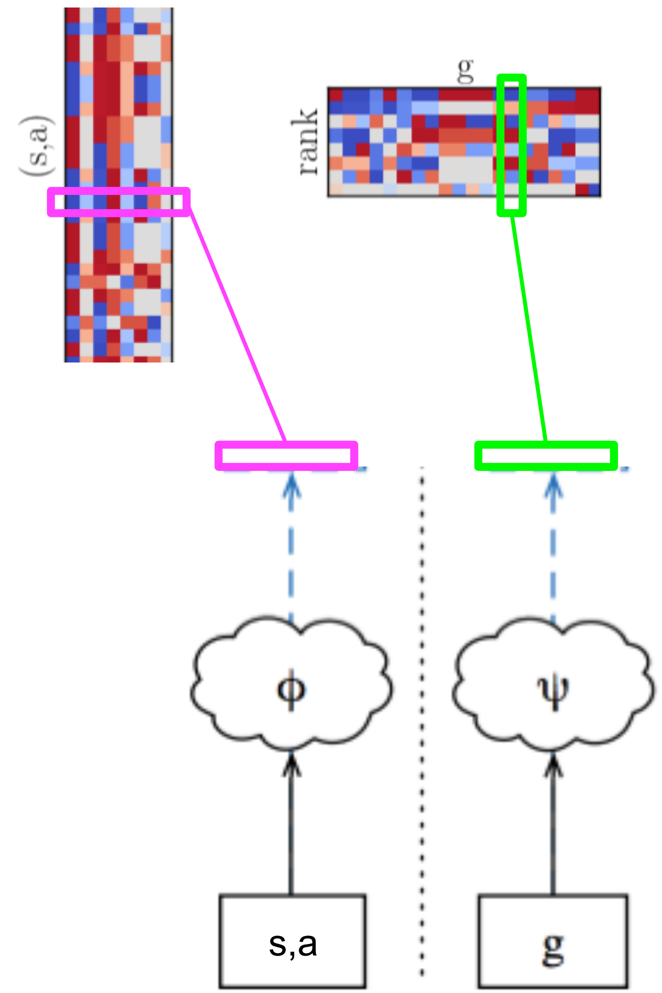
Stage 1: Factorize (low-rank)

- target embeddings for states and goals



Stage 2: Learn Embeddings

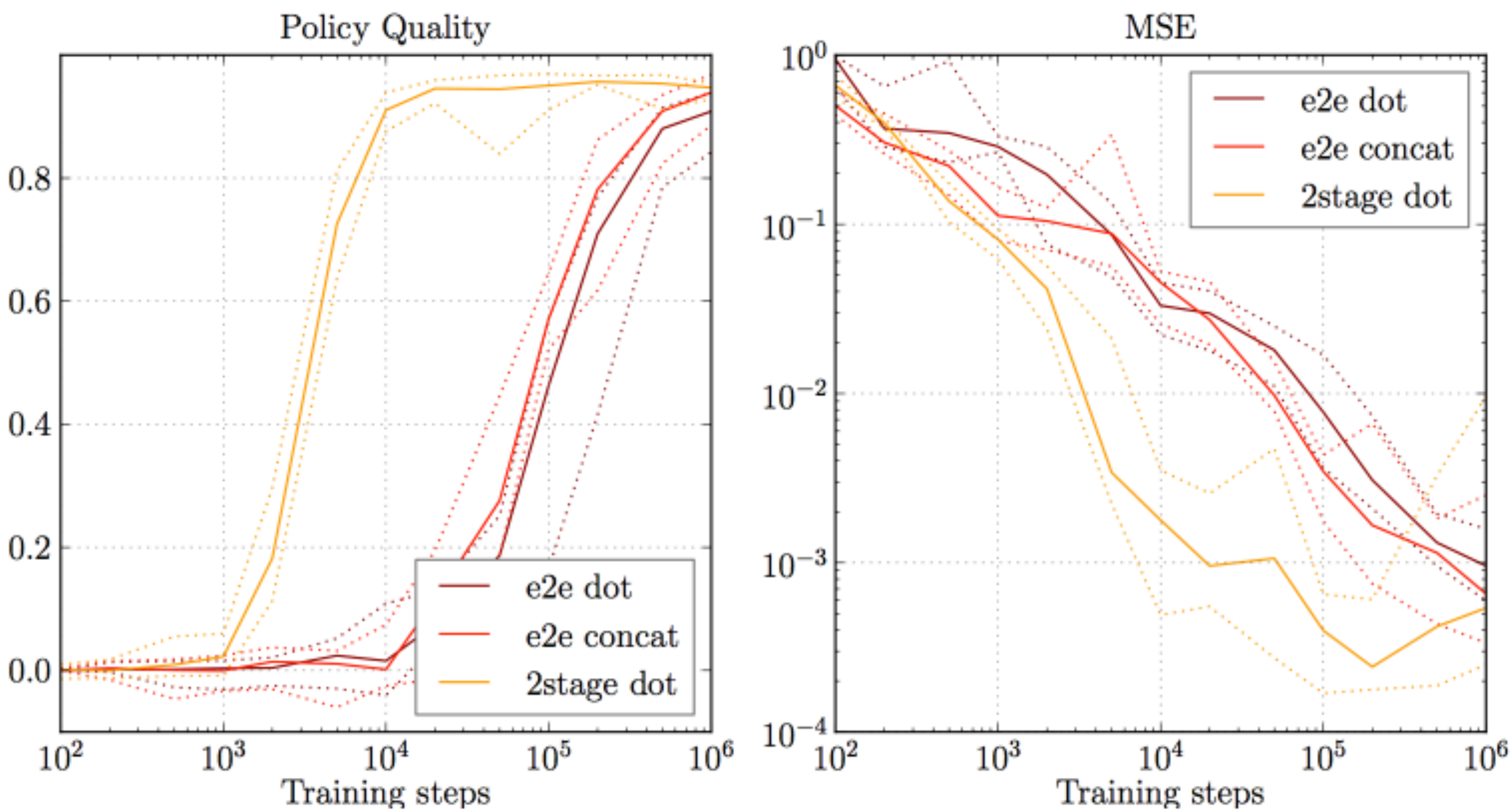
- regression from state/ subgoal features to target embeddings



(optional Stage 3): end-to-end fine-tuning

FLE vs end-to-end regression

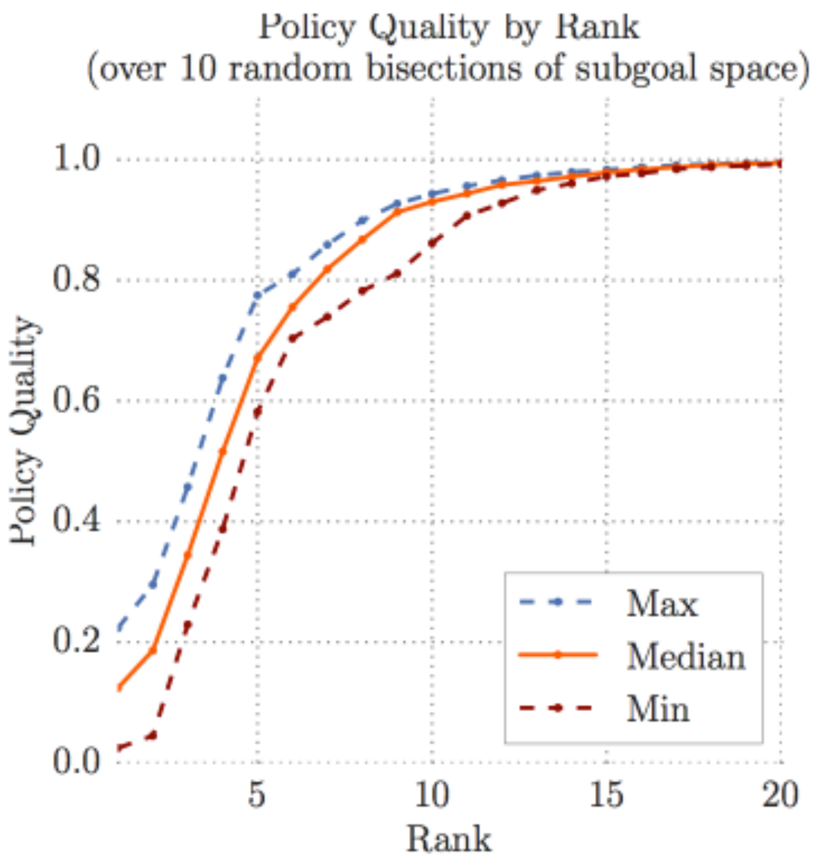
- between 10x and 100x faster

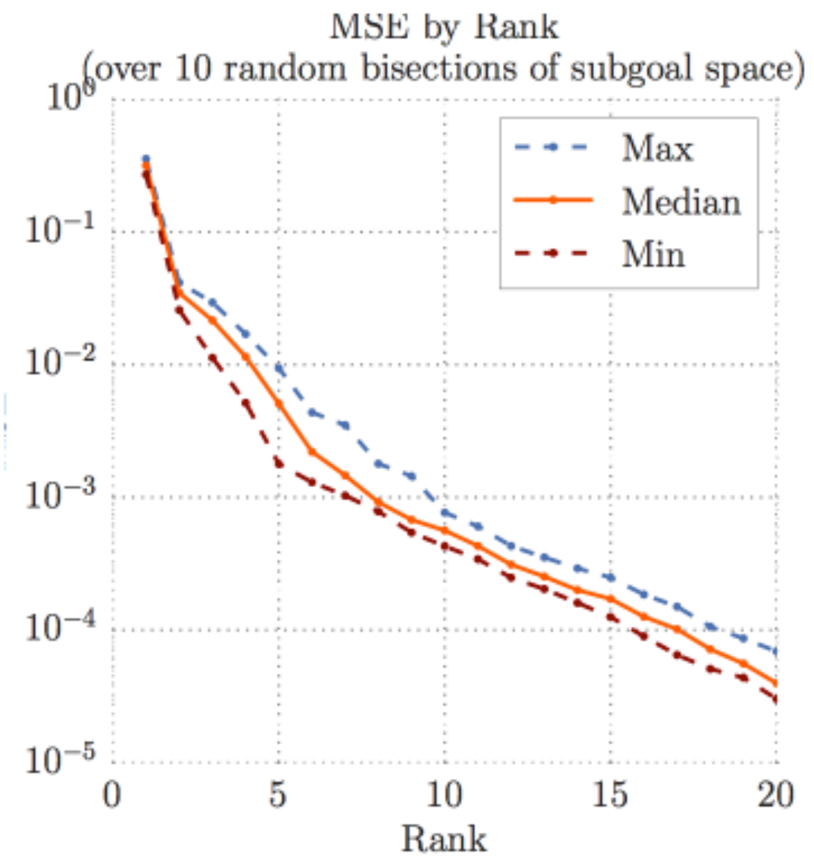


Outline

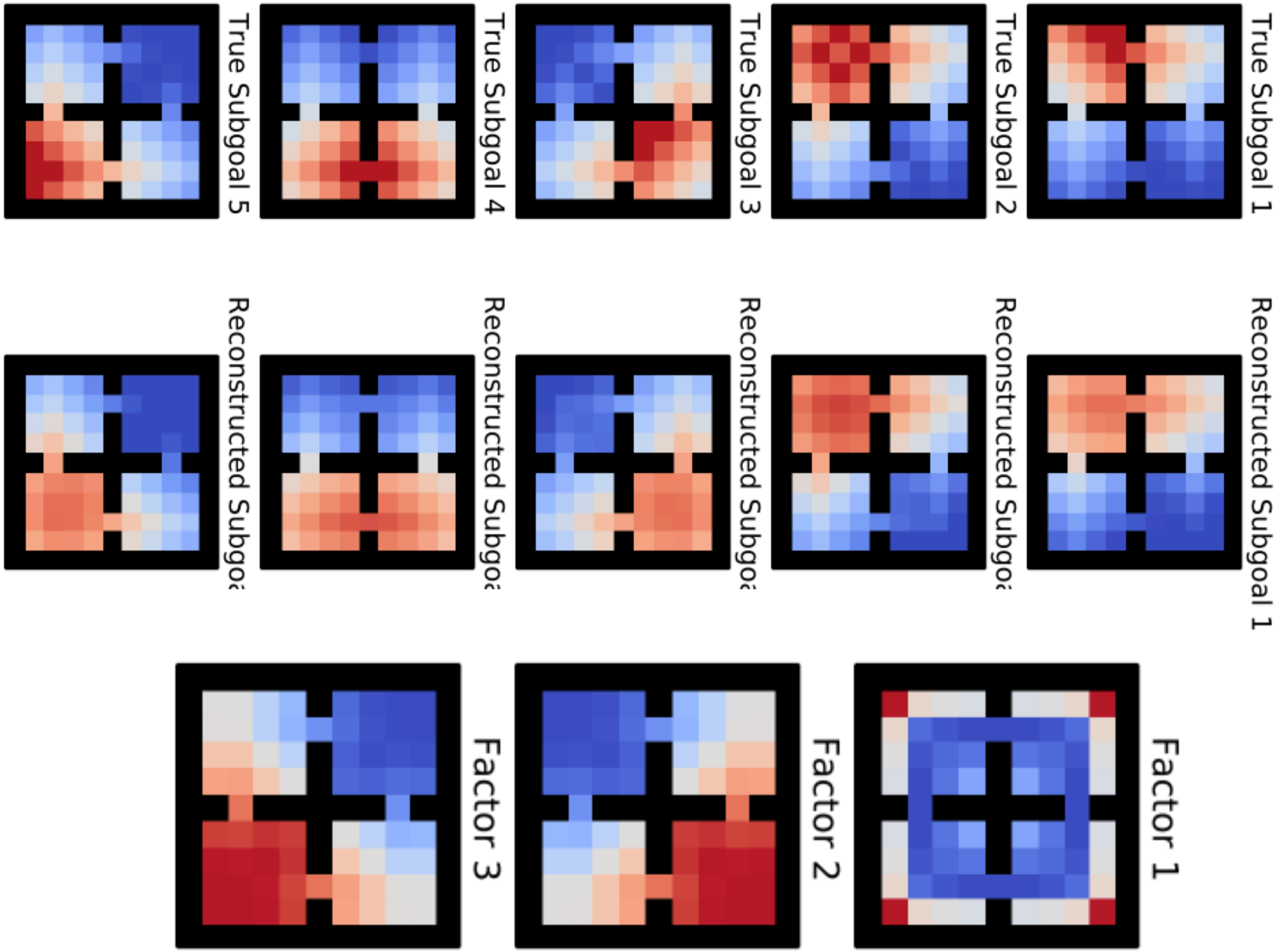
- Motivation
  - learn values for forecasts
  - efficiently for many subgoals
- Approach
  - new architecture: UVFA
  - one neat trick: FLE
- Results

Results: Low-rank is enough

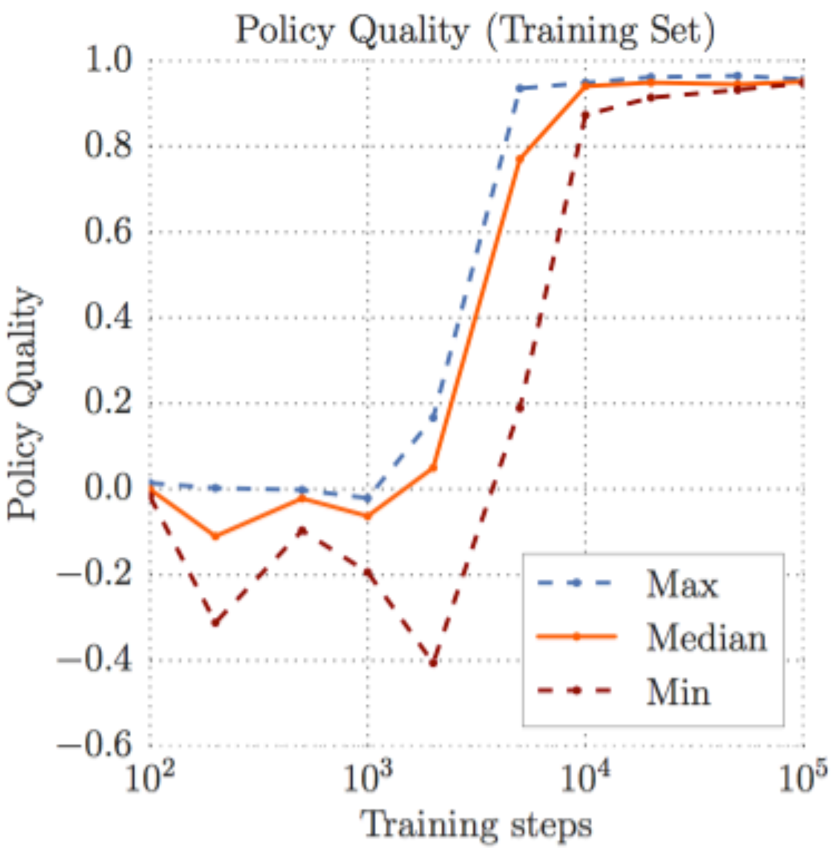


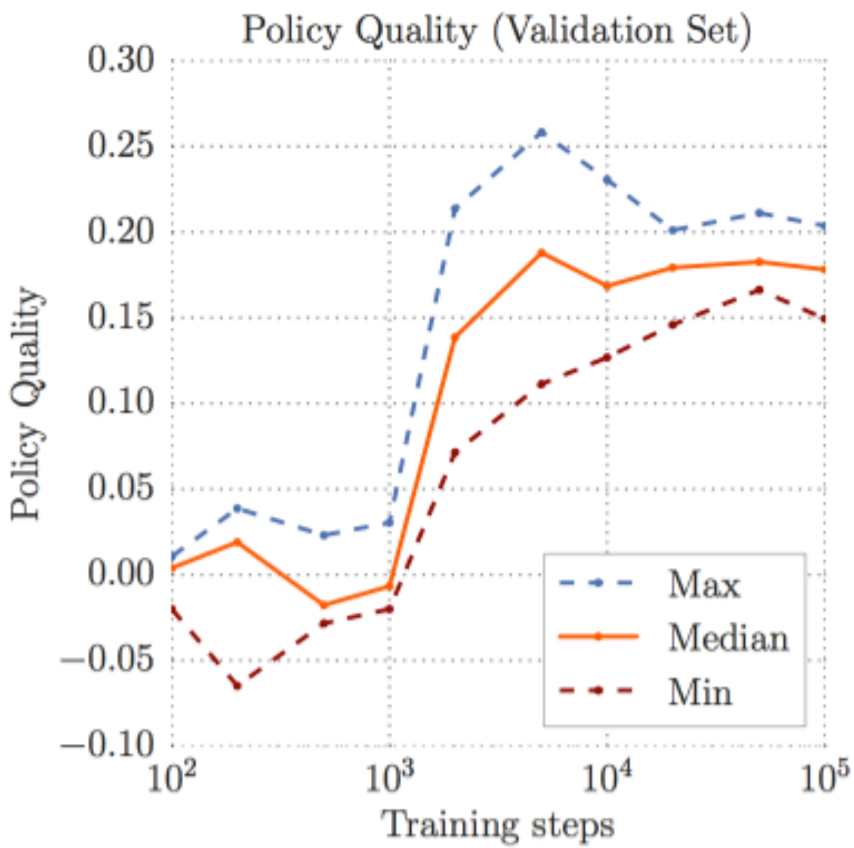


Results: Low-rank embeddings



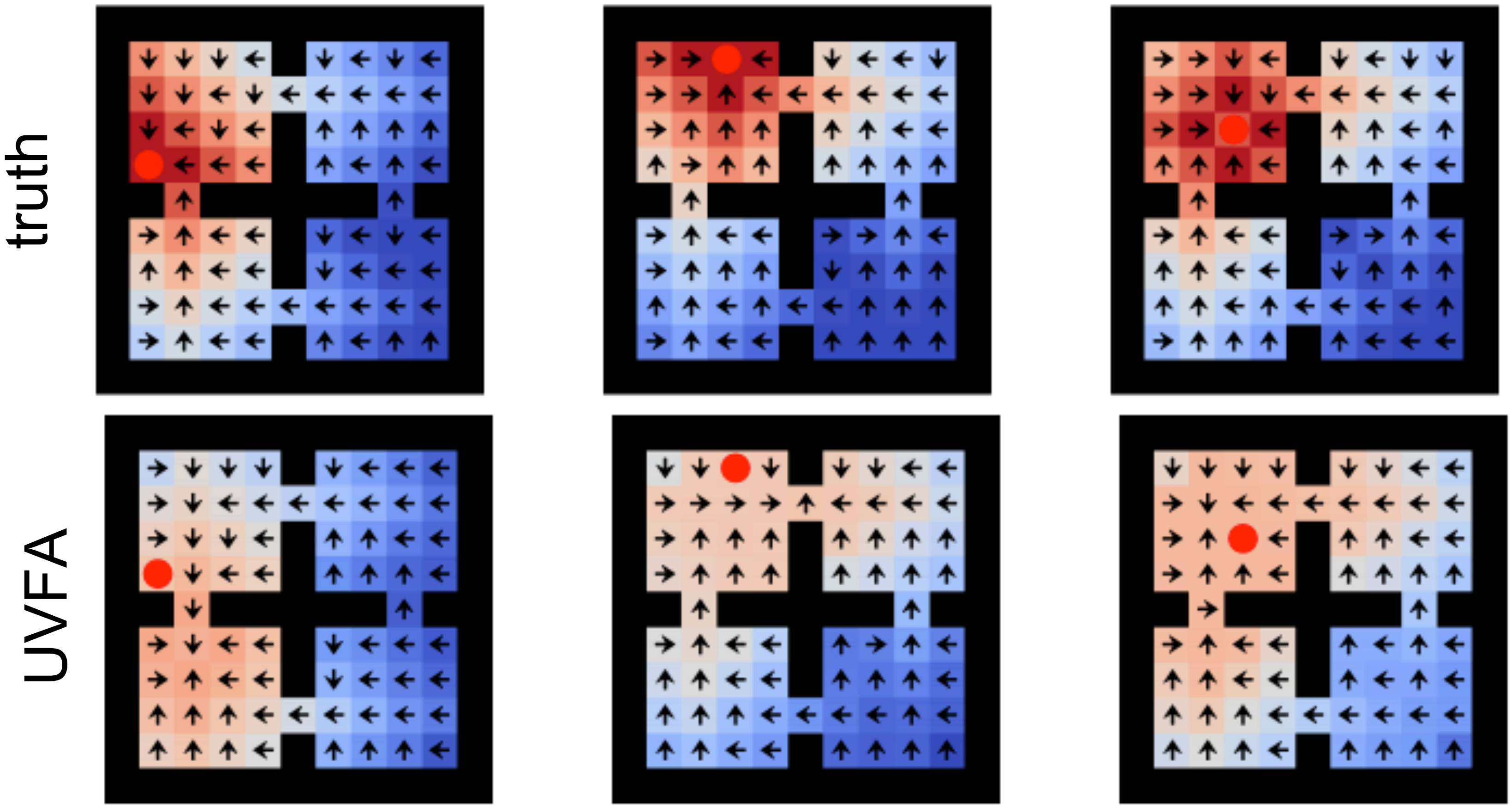
Results: Generalizing to new subgoals





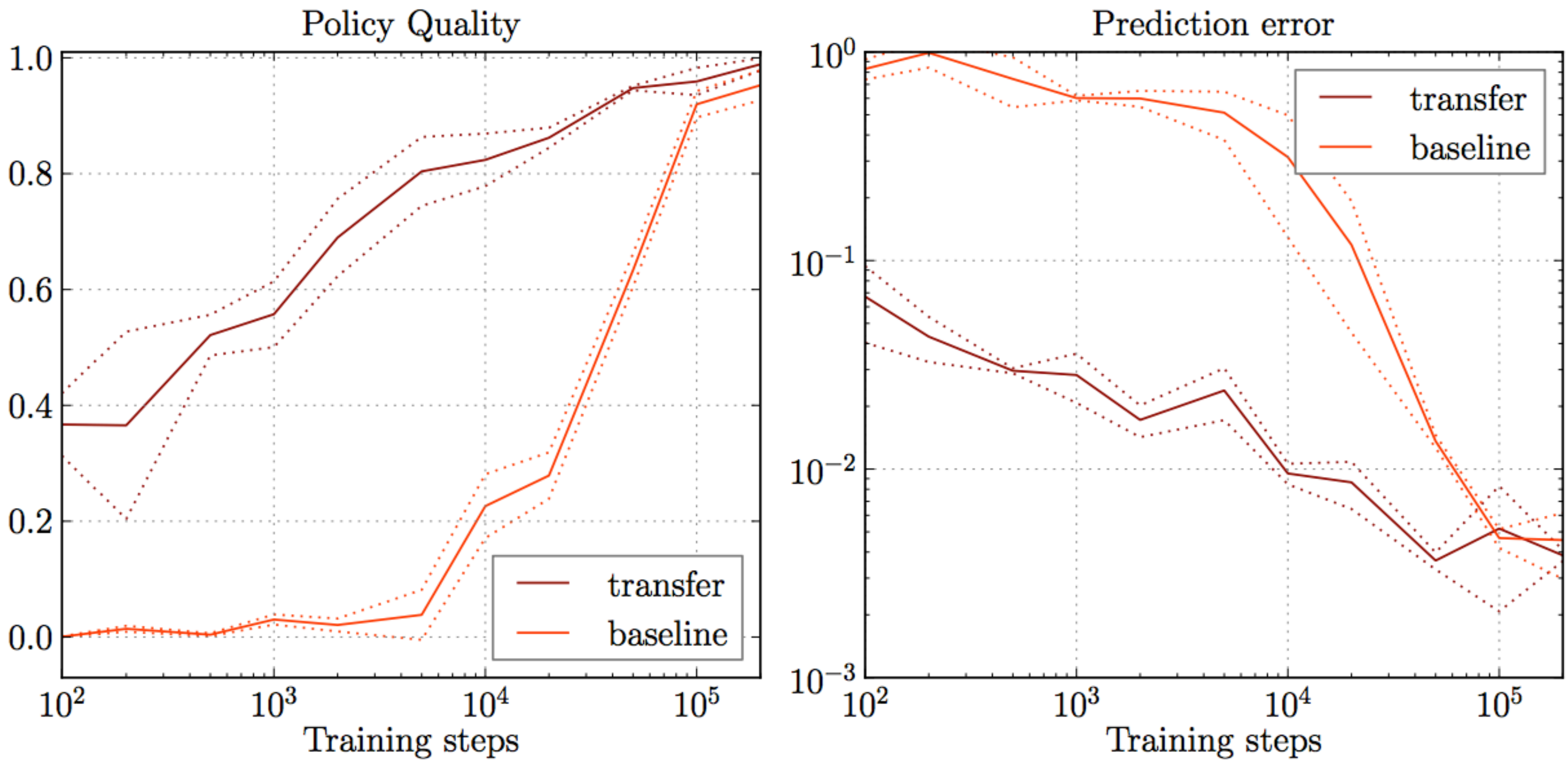
Results: Extrapolation

even to subgoals in unseen fourth room:



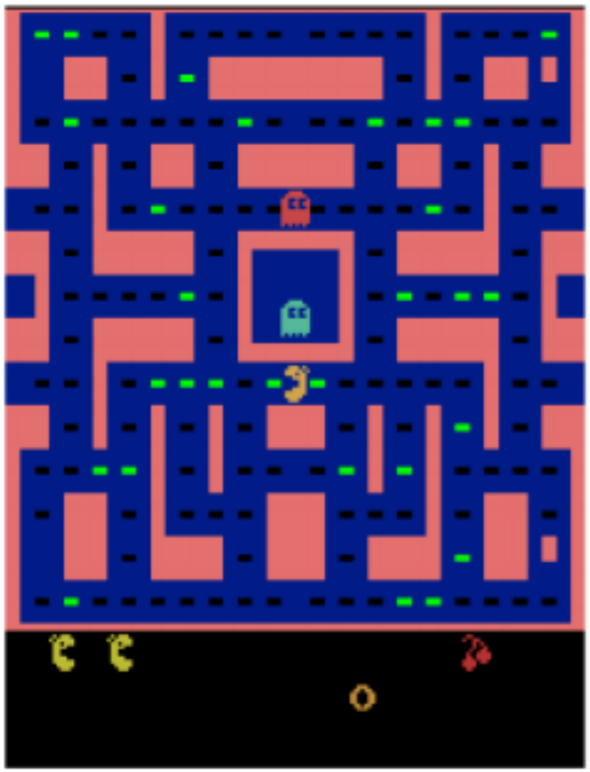
Results: Transfer to new subgoals

Refining UVFA is much faster than learning from scratch



Results: Pacman pellet subgoals

training set



test set

Results: pellet subgoal values (test set)

“truth”

UVFA generalization

Summary

- UVFA
  - compactly represent values for many subgoals
  - generalization, even extrapolation
  - transfer learning
- FLE
  - a trick for efficiently training UVFAs
  - side-effect: interesting embedding spaces
  - scales to complex domains (Pacman from raw vision)

Details: see our paper at ICML 2015