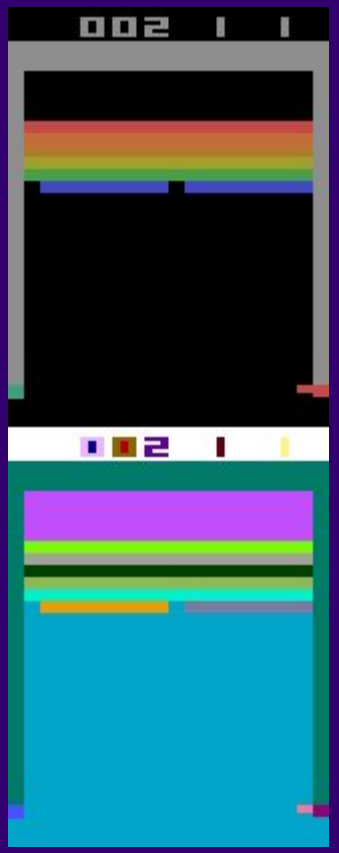
Object-Level Reinforcement Learning



William Agnew
Pedro Domingos

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY *of* WASHINGTON

Outline

- Motivation
  - Deep RL Algorithms are Incredibly Sample Inefficient
- Our Approach
  - Unsupervised Learning of Object-Level Representations
  - Object Dynamics Modelling
  - Estimating Reward and Value Function with a Linear Regressor
  - Planning with UCT
- Results
  - Atari Games
  - Interpretability
  - Transfer Learning

Motivation

- Deep RL algorithms achieve great performance
- But at the cost of many samples:
  - DQN[1] trained on 50 million frames, or 38 days of gameplay
- Samples can be very expensive; ex. robots
- Humans learn to play these games in mere minutes
- Can we do better?

Motivation

- How do humans learn and act in physical environments?
  - View the world in terms of objects
  - Model object dynamics
  - Use dynamics to plan
- Current state of the art: neural network outputs actions from current state
  - Reinventing objects, modelling, and planning all with a neural network
- Can we make a RL agent that explicitly does this?

Object-Level Reinforcement Learning

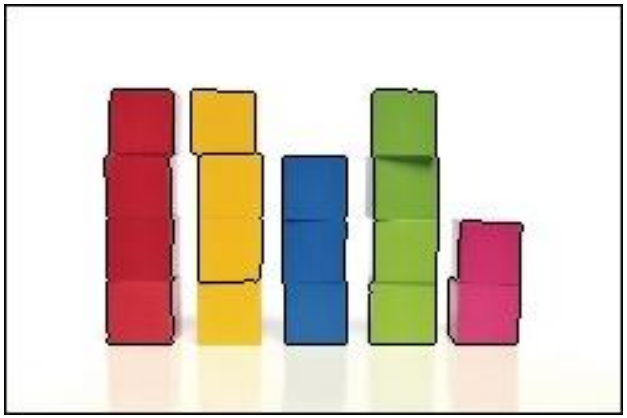
- We use a small number of powerful yet general principles to develop an object-level agent:
  - The world may be represented and modelled in terms of objects
  - All else equal, simpler representations are more accurate (Ockham's razor)

Object-Level Reinforcement Learning

- Our agent uses these principals to:
  - Learn an object-level world representation from pixels with no supervision
  - Learn predictive object dynamics models
  - Learn state-action value and rewards with a simple and interpretable approximator
  - Plan best actions using predictive models and value/reward approximators

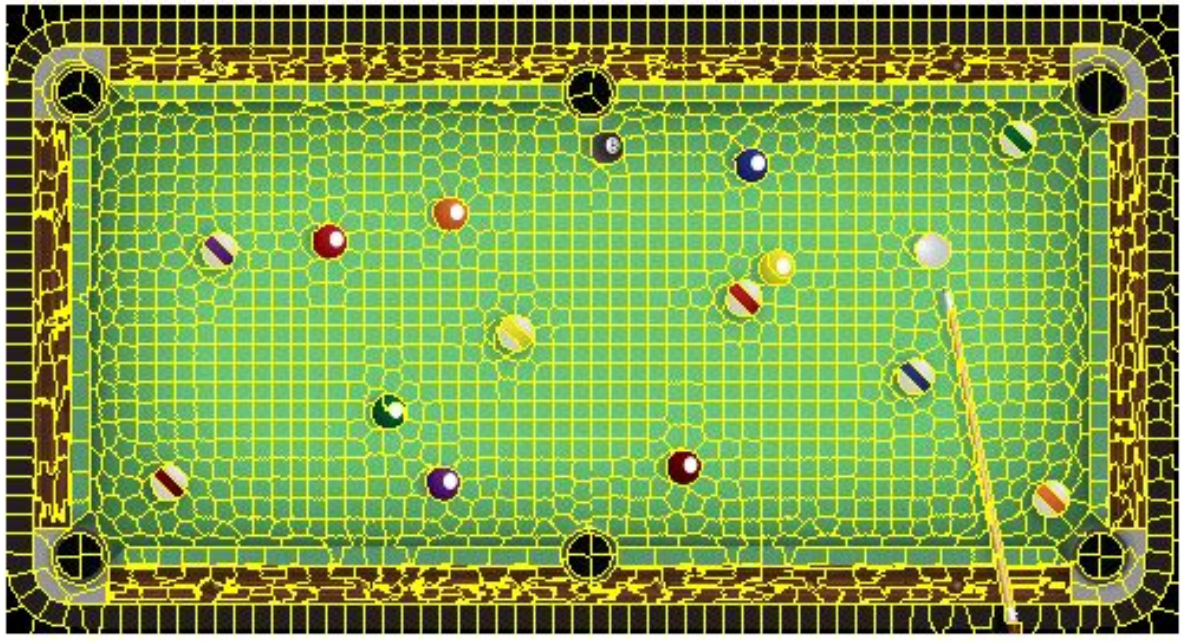
Unsupervised Learning of Objects

- How do we learn a mapping from pixels to objects with no supervision?
- Segmentation algorithms can over- or undersegment



Unsupervised Learning of Objects

- We can oversegment without losing information
- But the state representation may be very large

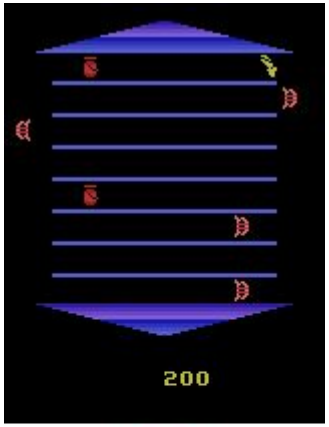


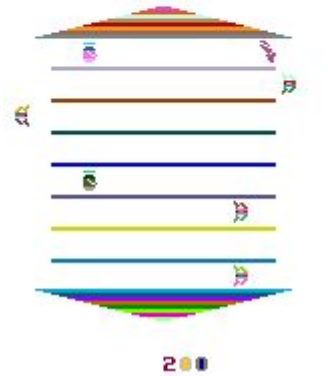
Unsupervised Learning of Objects

- How can well tell if two segments are part of the same object?
- Two segments are part of the same object if they behave in the same way

Unsupervised Learning of Objects

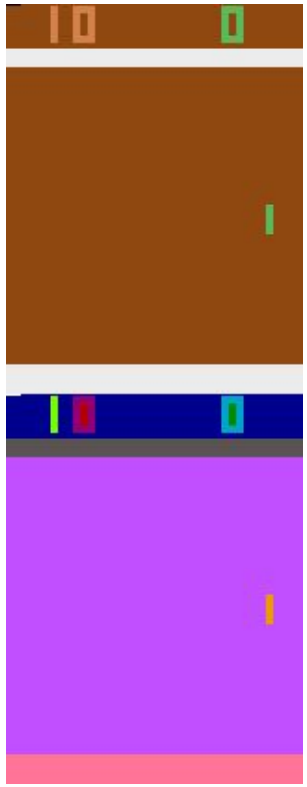
- Use a simple algorithm to produce an over segmentation
- Simple and fallible object tracking
- Train models on segment dynamics
- Combine segments
- Represent state as object absolute and relative positions, velocities, accelerations, and estimated contacts





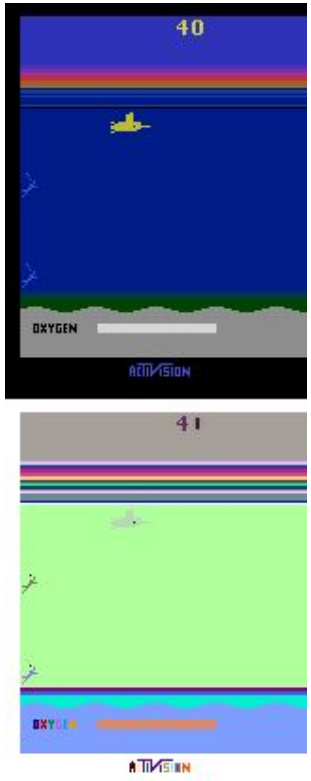
Modelling Object Dynamics

- Model object acceleration at each timestep
- One model per object, per dimension
- Discretize accelerations, use multiclass classifiers to output a probability distribution over possible accelerations
- Also learn when and where objects will appear/disappear



Estimating State-Action Value and Reward

- Monte Carlo estimation of state-action values from experiences
- Fit a linear regressor from object-level representation to state-action value or reward
- Value functions for most takes in physical world can be easily represented as a linear combination of collisions and relative positions
  - Go to an object
  - Touch an object
  - Avoid an object



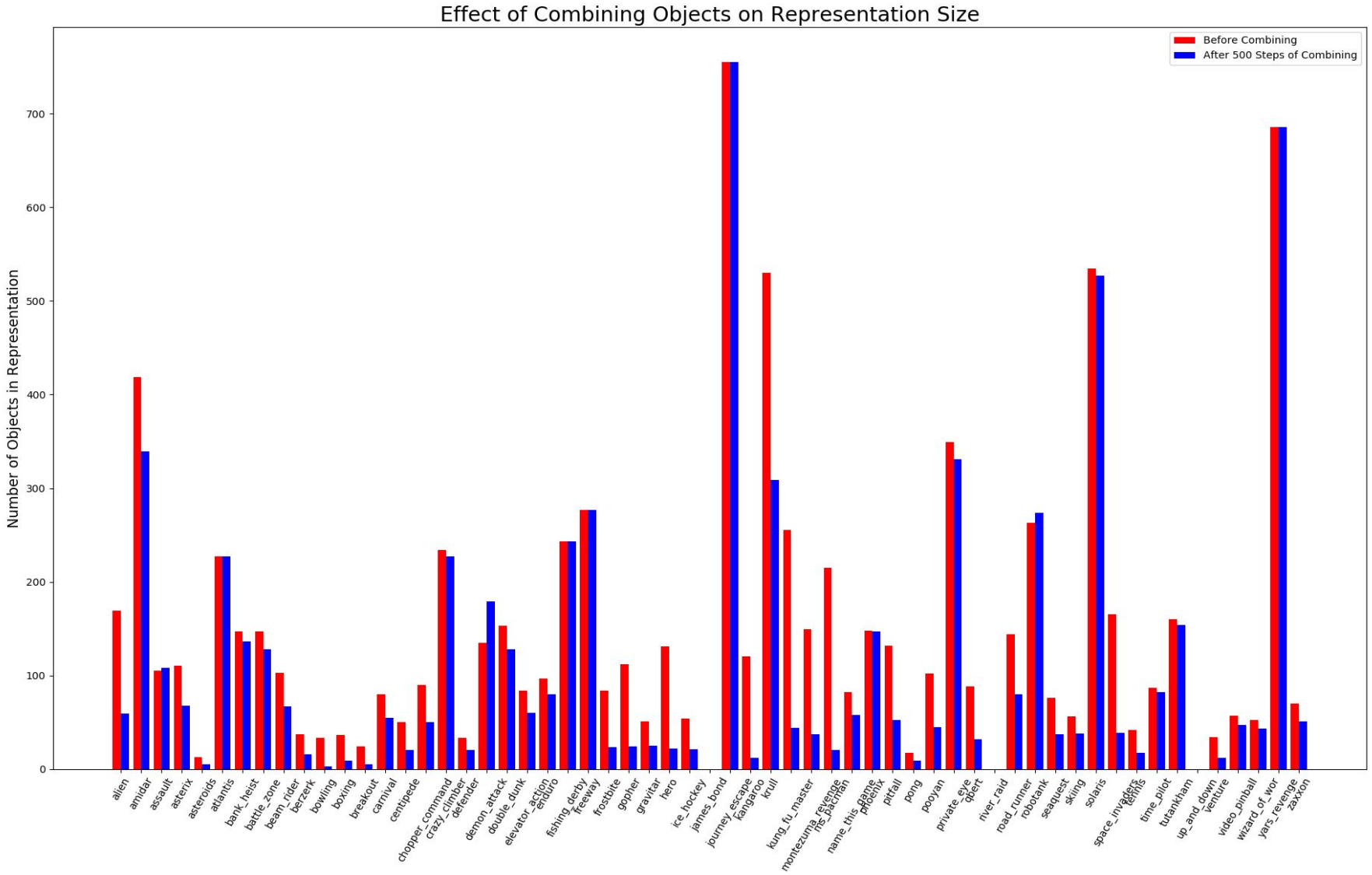
Estimating State-Action Value and Reward

- A linear regression on object-level features is readily interpretable-we can easily understand the learned policy
- Object-level representation is also easy to reason about
- Reinforcement learner won't have bad edge case behavior on known states (known objects), recognizes when it has encountered new states (new objects)

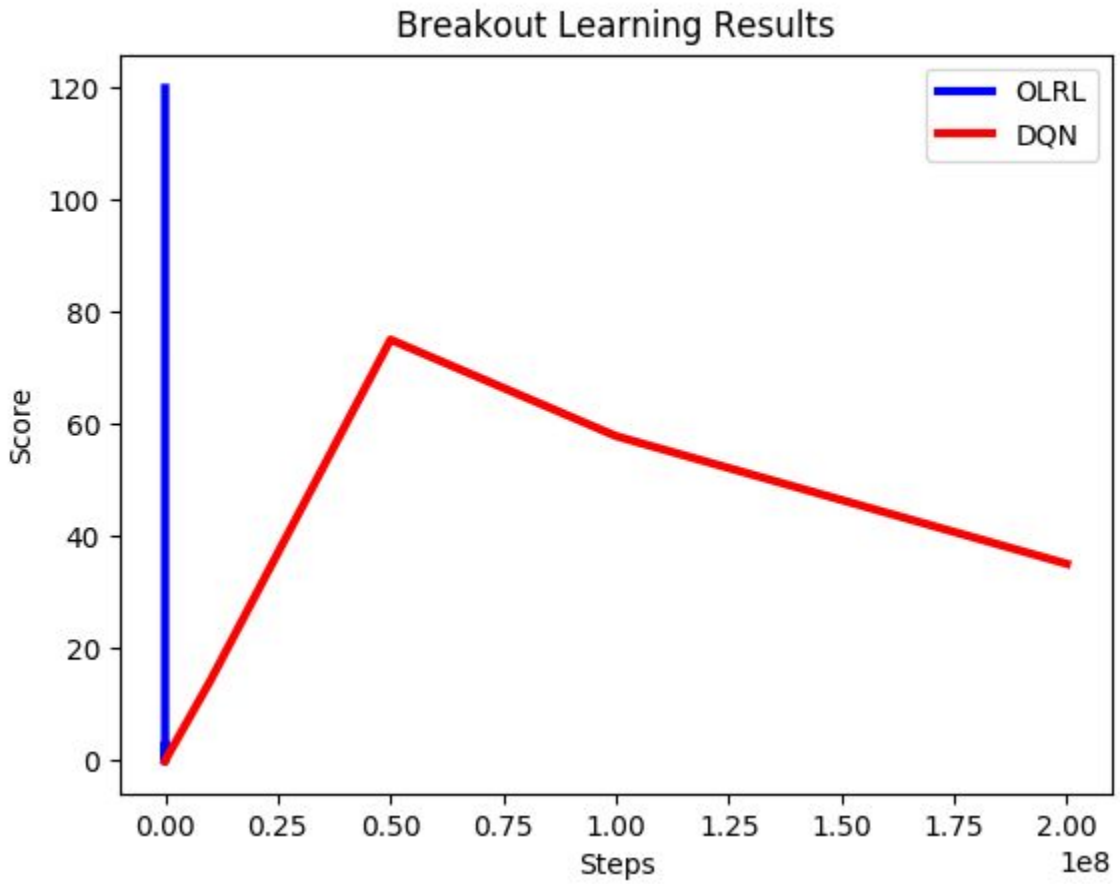
Planning

- Given a model of the world and an estimator for state-action value and reward, how do we choose the best next action to take?
- Use UCT to find the best future action, approximating playouts with the value estimator
- Similar approach as used in AlphaGo[2]

Results: Learning Object-Level Representations

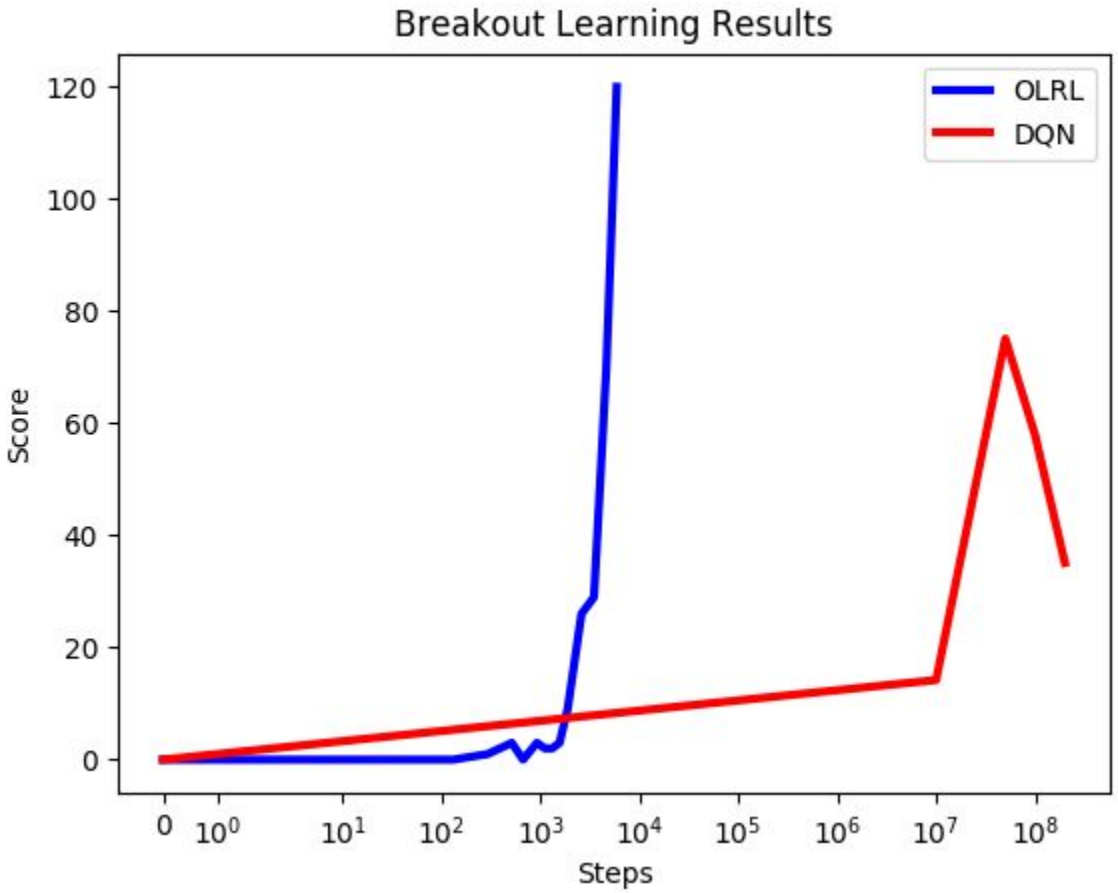


Results: Learning Curves



Evaluation methodology and DQN data from [3]

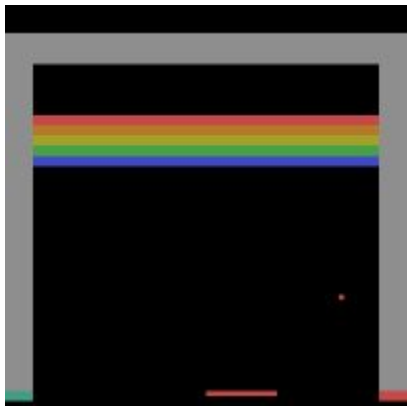
Results: Learning Curves



Results: Interpretability

Linear state value estimator weights on object pair relative distance

Object Pair	Ball-Paddle	Ball-Background	Paddle-Background
Vertical Dimension	0.020	-0.040	0.101
Horizontal Dimension	-0.057	0.003	0.029



Conclusion

- We developed a model-based reinforcement learner that is over two orders of magnitude more sample efficient than DQN while achieving comperable performance
- Our agent achieves vastly better results than DQN on transfer learning tasks
- Our agent is also interpretable
- Modelling and planning are well studied fields: we believe this paradigm will extend well to more complex domains

References

1. Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.
2. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), 484-489.
3. Machado, Marlos C., et al. "Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents." *arXiv preprint arXiv:1709.06009* (2017).