



“All the hundreds of millions of people who, in their time, believed the Earth was flat never succeeded in unrounding it by an inch”

Isaac Asimov

CMPUT 365

Introduction to RL

Reminder

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks.

You **need to check, every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

The deadlines in the public session **do not align** with the deadlines in Coursera.

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

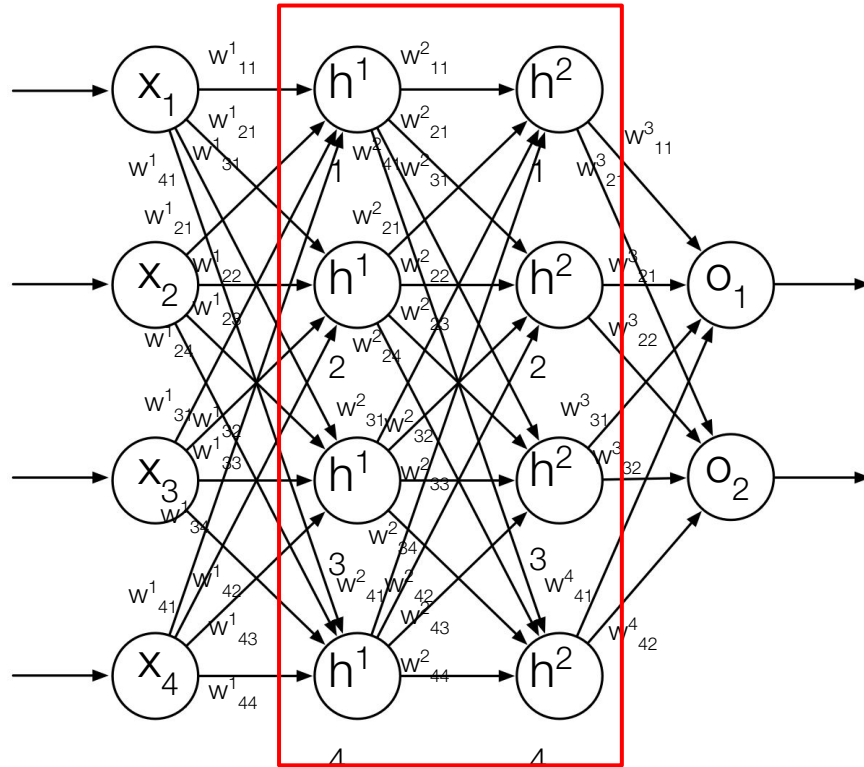
Reminders and Notes

- The practice quiz for Control with FA (new week) is due on Monday (we are almost there).
- We won't cover Module 4: A Complete RL System (Capstone) in this course.
- Rich Sutton will give a guest lecture Dec 9th, Monday. Spread the word.
- A note on the final exam:
 - The required reading from the syllabus does not mean that's what will be covered in the final exam. There are some mismatches. Anything we discussed in class is fair game, including Maximization Bias and Double Learning (Section 6.7), and Nonlinear Function Approximation: Artificial Neural Networks (Section 9.7).

Please, interrupt me at any time!



Last Class: Neural Networks



**Representation
(Learned features)**

$$\mathbf{h}^1 = \text{act}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$

$$\text{s.t. } h^1_1 = x_1 w^1_{11} + x_2 w^1_{21} + x_3 w^1_{31} + x_4 w^1_{41} + B^1$$

$$\mathbf{h}^2 = \text{act}(\mathbf{h}^1\mathbf{W}^2 + \mathbf{b}^2)$$

$$\text{s.t. } h^2_1 = h^1_1 w^2_{11} + h^1_2 w^2_{21} + h^1_3 w^2_{31} + h^1_4 w^2_{41} + B^2$$

$$\mathbf{o} = \text{act}(\mathbf{h}^2\mathbf{W}^3 + \mathbf{b}^3)$$

$$\text{s.t. } o_1 = h^2_1 w^3_{11} + h^2_2 w^3_{21} + h^2_3 w^3_{31} + h^2_4 w^3_{41} + B^2$$

$$\mathbf{o} = \text{act}(\text{act}(\text{act}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2)\mathbf{W}^3 + \mathbf{b}^3)$$



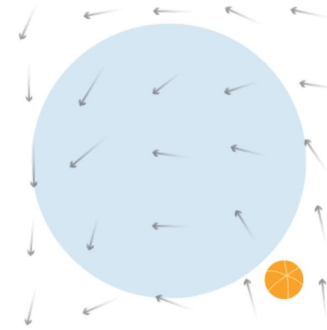
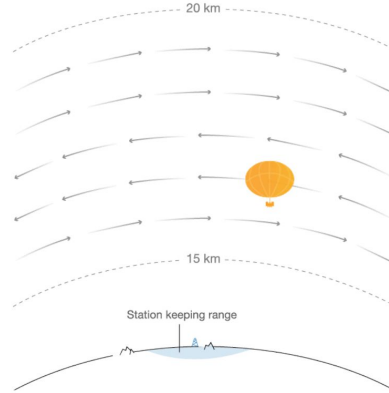
A Note from the Textbook

The backpropagation algorithm can produce good results for shallow networks having 1 or 2 hidden layers, but it may not work well for deeper ANNs. In fact, training a network with $k + 1$ hidden layers can actually result in poorer performance than training a network with k hidden layers, even though the deeper network can represent all the functions that the shallower network can (Bengio, 2009). Explaining results like these is not easy, but several factors are important. First, the large number of weights in a typical deep ANN makes it difficult to avoid the problem of overfitting, that is, the problem of failing to generalize correctly to cases on which the network has not been trained. Second, backpropagation does not work well for deep ANNs because the partial derivatives computed by its backward passes either decay rapidly toward the input side of the network, making learning by deep layers extremely slow, or the partial derivatives grow rapidly toward the input side of the network, making learning unstable. Methods

Things change quickly...

What we feed to a neural network still matters (a lot!)

<https://www.nature.com/articles/s41586-020-2939-8>

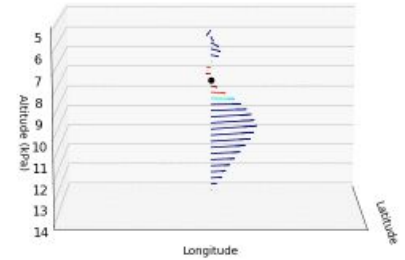


Article
Autonomous navigation of stratospheric balloons using reinforcement learning

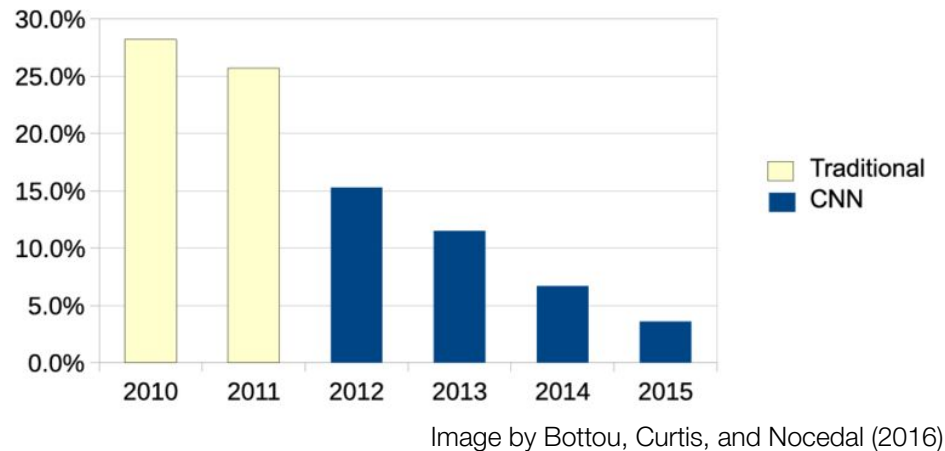
<https://doi.org/10.1038/s41586-020-2939-8> **Mari G. Bellemare^{1*}, Salvatore Candido^{1†}, Pablo Samuel Castro², Jun Gong³, Marlos C. Machado³, Subhodeep Moitra³, Simeone D. Ponda³ & Piya Wang³**

Accepted: 29 September 2020
 Published online: 2 December 2020
 Check for updates

Efficiently navigating a superpressure balloon in the stratosphere¹ requires the integration of a multitude of cues, such as wind speed and solar elevation, and the process is complicated by forecast errors and sparse wind measurements. Coupled with the need to make decisions in real time, these factors rule out the use of conventional control techniques^{2–4}. Here we describe the use of reinforcement learning^{5,6} to create a high-performing flight controller. Our algorithm uses data augmentation⁷ and a self-correcting design to overcome the key technical challenge of reinforcement learning from imperfect data, which has proved to be a major obstacle to its application to physical systems⁸. We deployed our controller to station Loon superpressure balloons at multiple locations across the globe, including a 39-day controlled experiment over the Pacific Ocean. Analyses show that the controller outperforms Loon's previous algorithm and is robust to the natural diversity in stratospheric winds. These results demonstrate that reinforcement learning is an effective solution to real-world autonomous control problems in which



Are we improving anything? From features to architecture...



Deep Convolutional Network

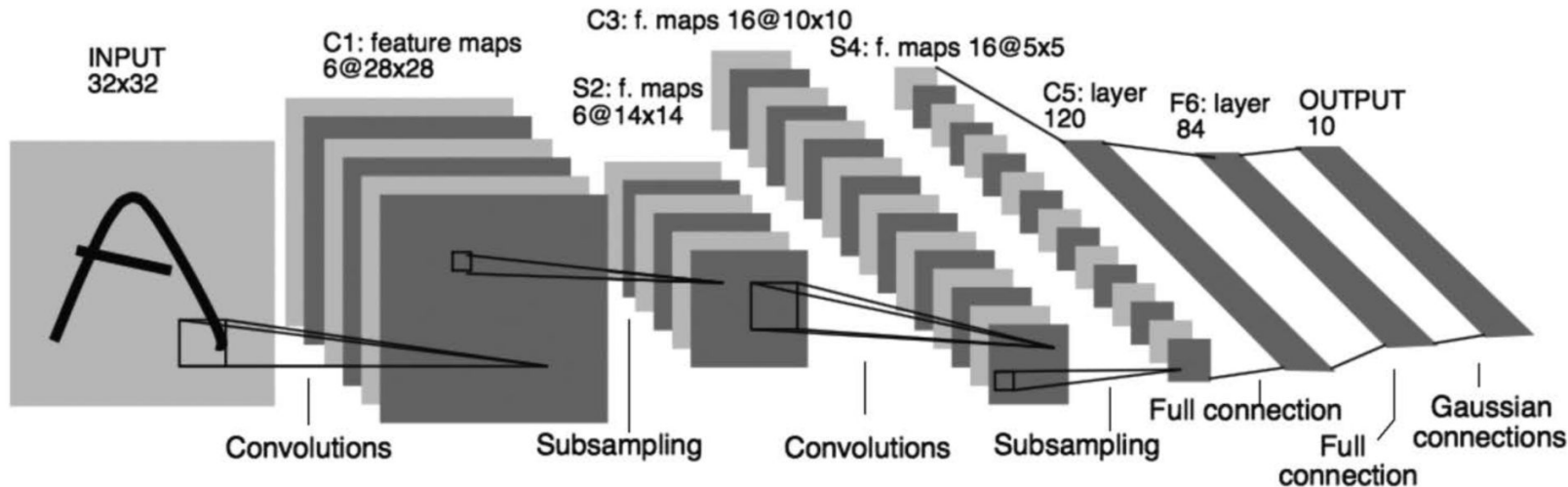
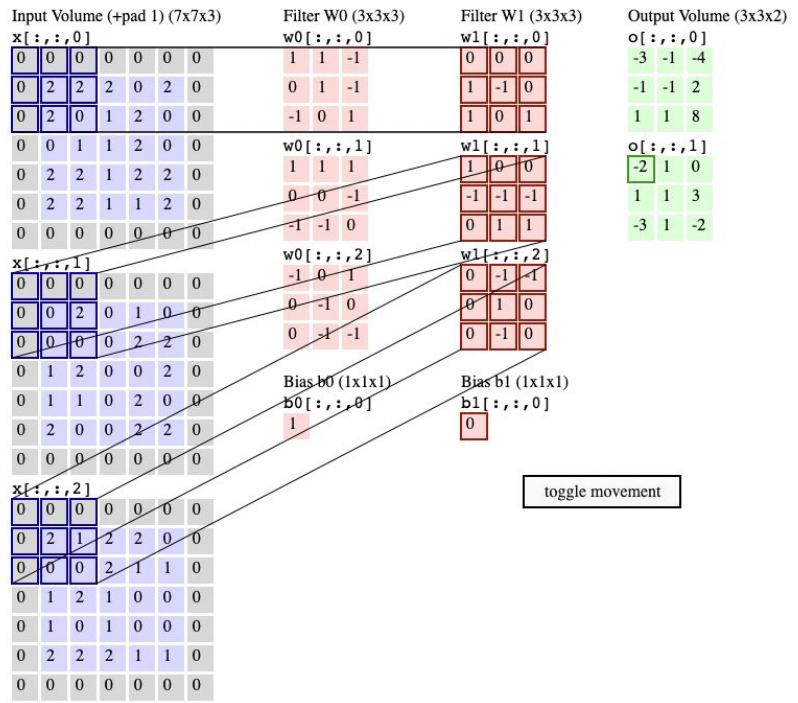
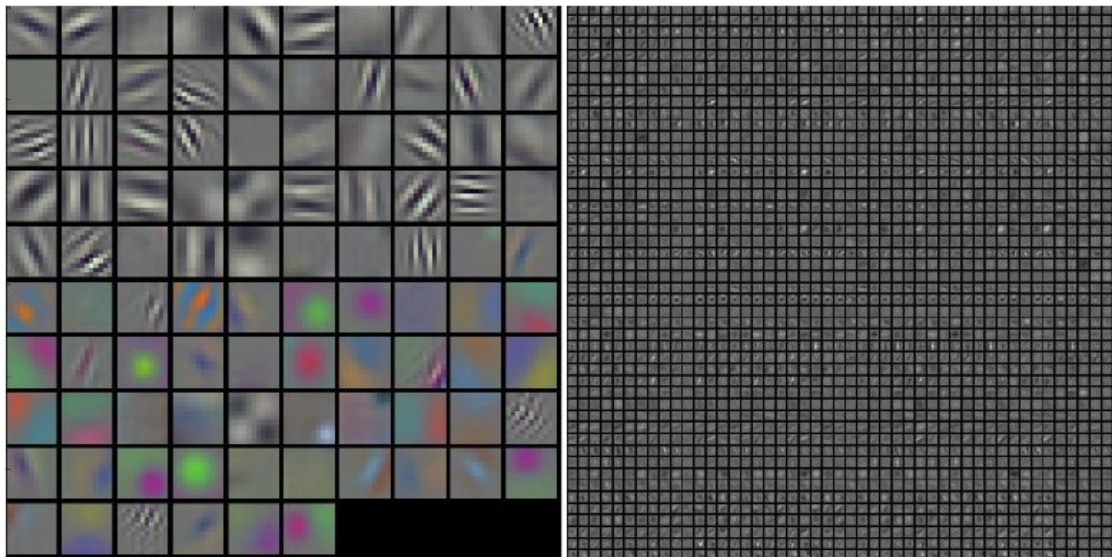


Figure 9.15: Deep Convolutional Network. Republished with permission of Proceedings of the IEEE, from Gradient-based learning applied to document recognition, LeCun, Bottou, Bengio, and Haffner, volume 86, 1998; permission conveyed through Copyright Clearance Center, Inc.

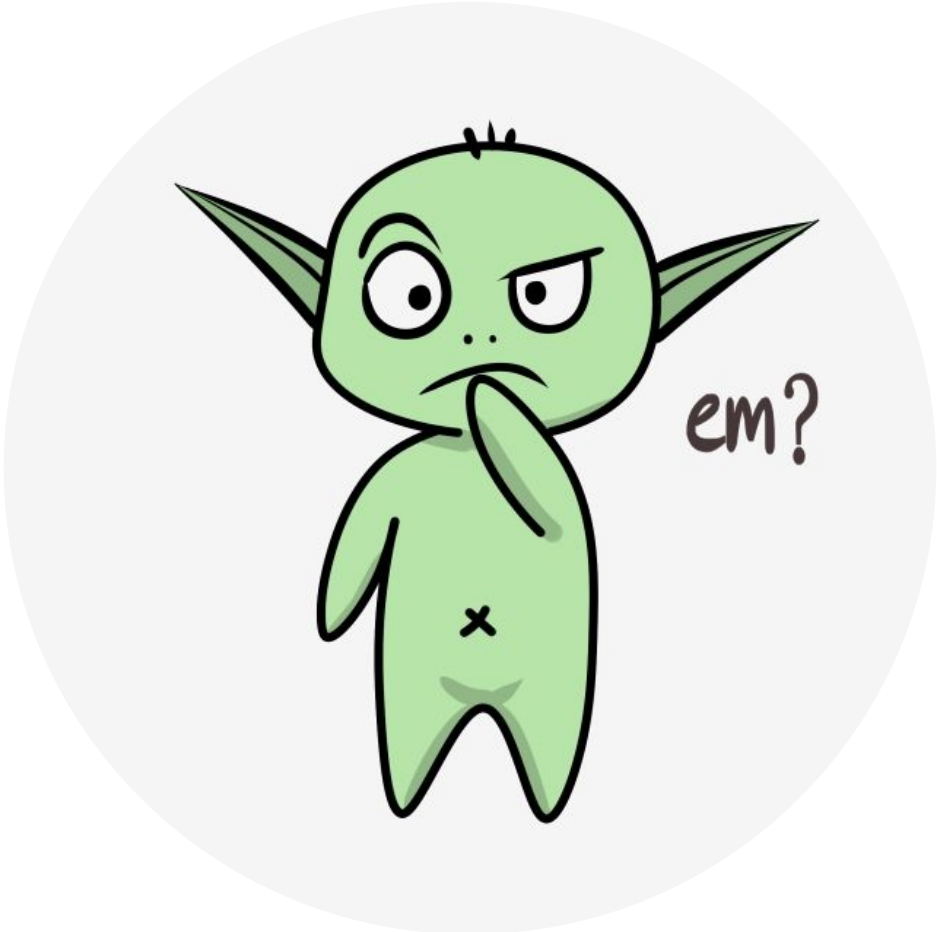
Deep Convolutional Network



Learned Representations



Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet. Notice that the first-layer weights are very nice and smooth, indicating nicely converged network. The color/grayscale features are clustered because the AlexNet contains two separate streams of processing, and an apparent consequence of this architecture is that one stream develops high-frequency grayscale features and the other low-frequency color features. The 2nd CONV layer weights are not as interpretable, but it is apparent that they are still smooth, well-formed, and absent of noisy patterns.



Journal of Artificial Intelligence Research 47 (2013) 253–279

Submitted 02/13; published 06/13

The Arcade Learning Environment: An Evaluation Platform for General Agents

Marc G. Bellemare

University of Alberta, Edmonton, Alberta, Canada

MG17@CS.UALBERTA.CA

Yavar Naddaf

*Empirical Results Inc., Vancouver,
British Columbia, Canada*

YAVAR@EMPIRICALRESULTS.CA

Joel Veness

Michael Bowling

University of Alberta, Edmonton, Alberta, Canada

VENESS@CS.UALBERTA.CA

BOWLING@CS.UALBERTA.CA

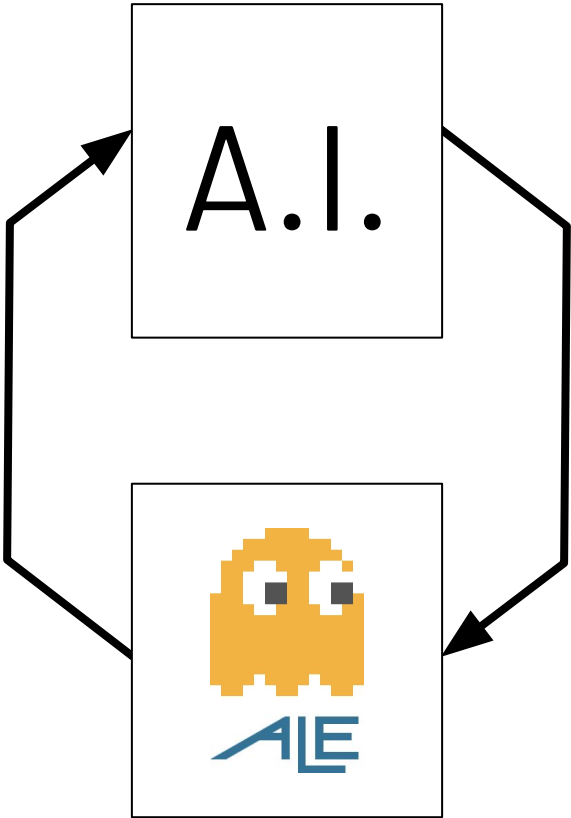
Abstract

In this article we introduce the Arcade Learning Environment (ALE): both a challenge problem and a platform and methodology for evaluating the development of general, domain-independent AI technology. ALE provides an interface to hundreds of Atari 2600 game environments, each one different, interesting, and designed to be a challenge for human players. ALE presents significant research challenges for reinforcement learning, model learning, model-based planning, imitation learning, transfer learning, and intrinsic motivation. Most importantly, it provides a rigorous testbed for evaluating and comparing approaches to these problems. We illustrate the promise of ALE by developing and benchmarking domain-independent agents designed using well-established AI techniques for both reinforcement learning and planning. In doing so, we also propose an evaluation

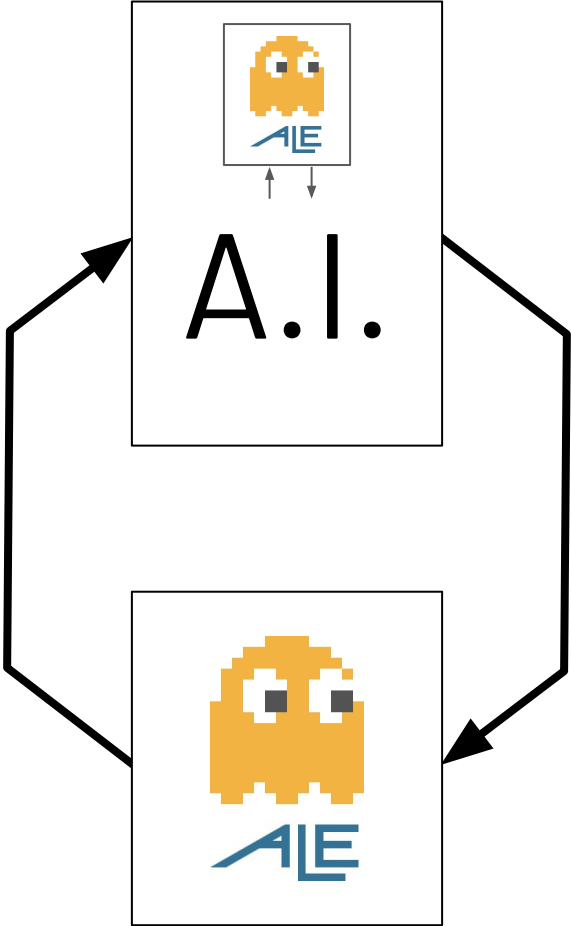
Arcade Learning Environment

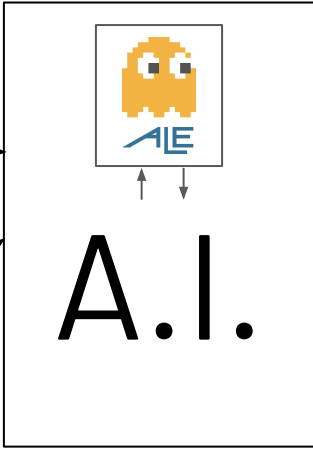
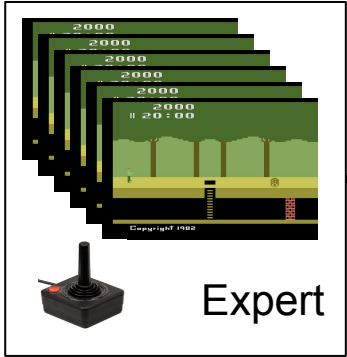
Over 50 Domains in 8 Minutes 23 Seconds

Reinforcement Learning



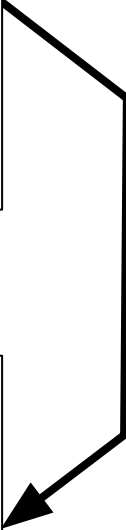
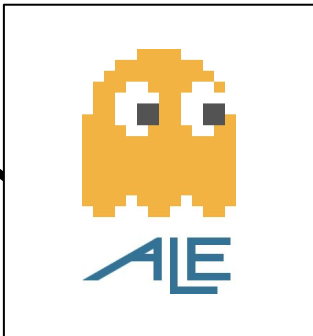
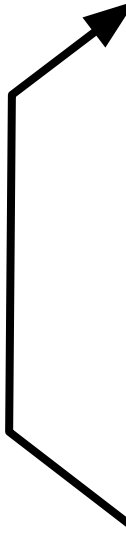
Model Learning



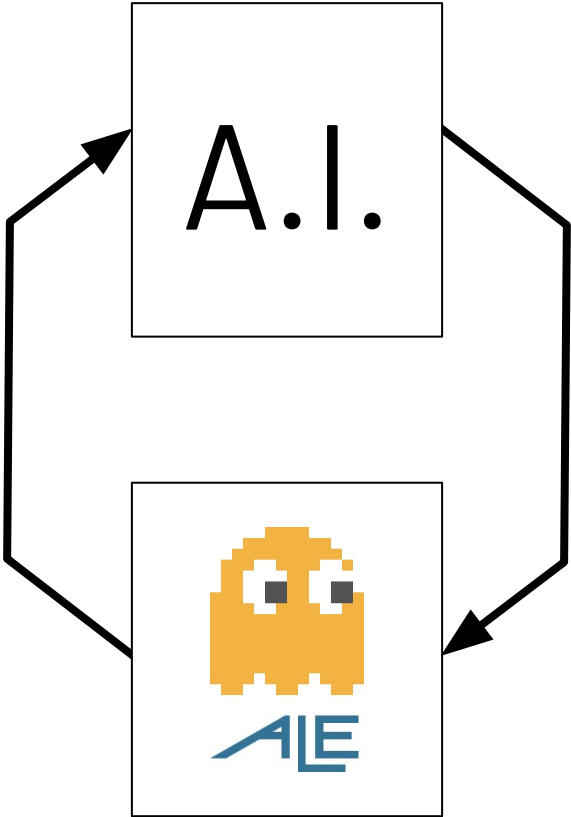


A.I.

Imitation/Apprenticeship Learning



Exploration



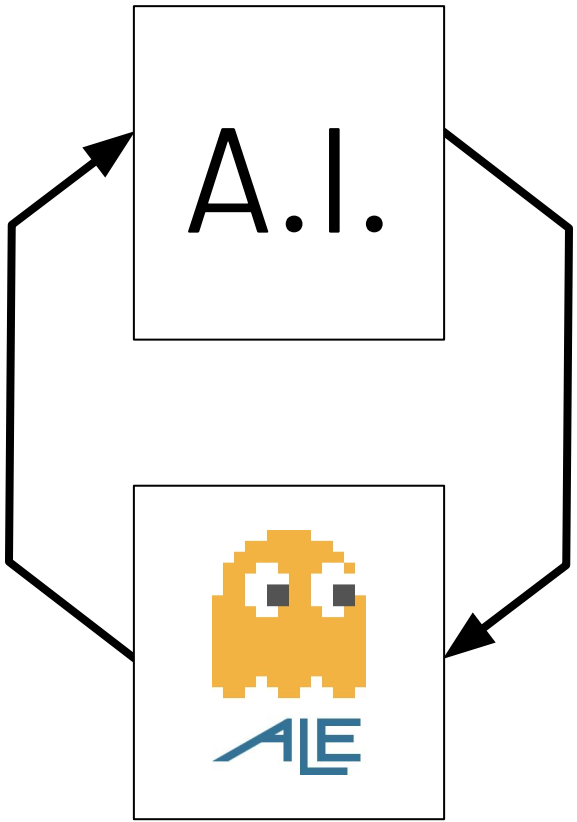
Transfer Learning

Pitfall!

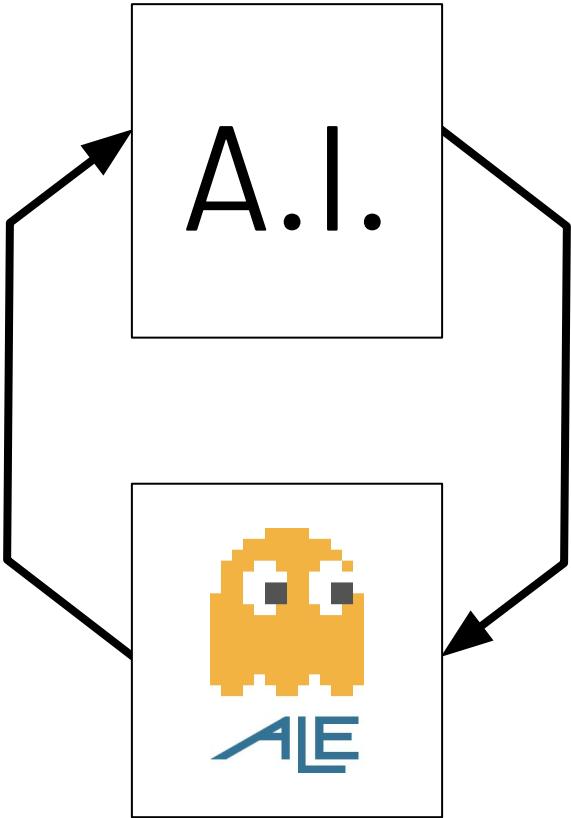


⋮

Pitfall II



Intrinsic Motivation





Deep Q-Network (and Deep RL)

[Mnih et al., 2013, 2015]

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

Daan Wierstra Martin Riedmiller

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Abstract

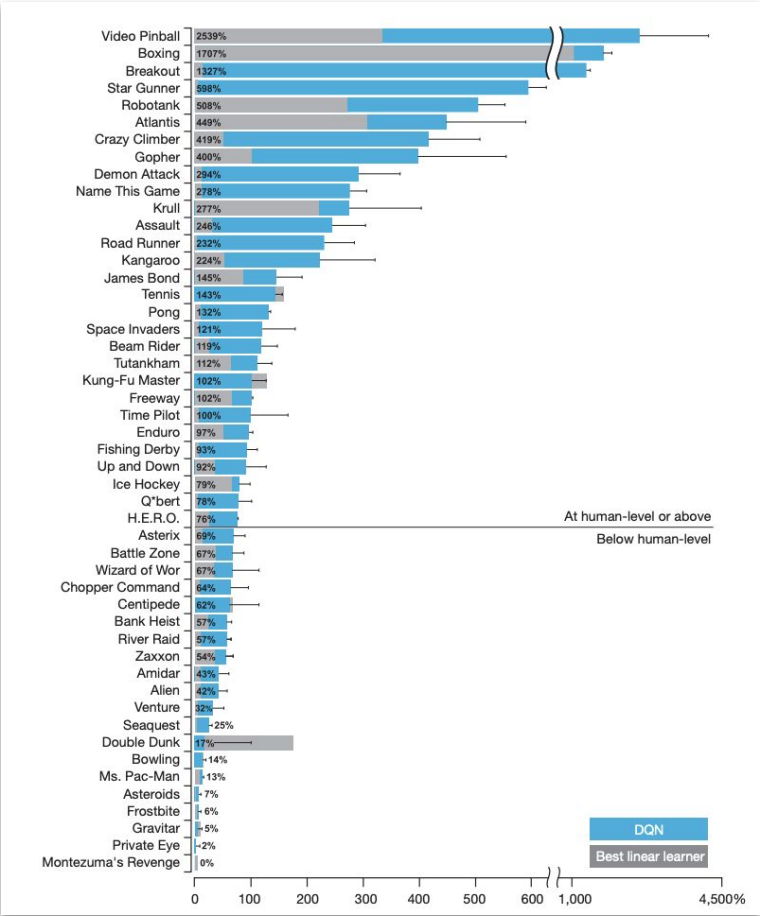
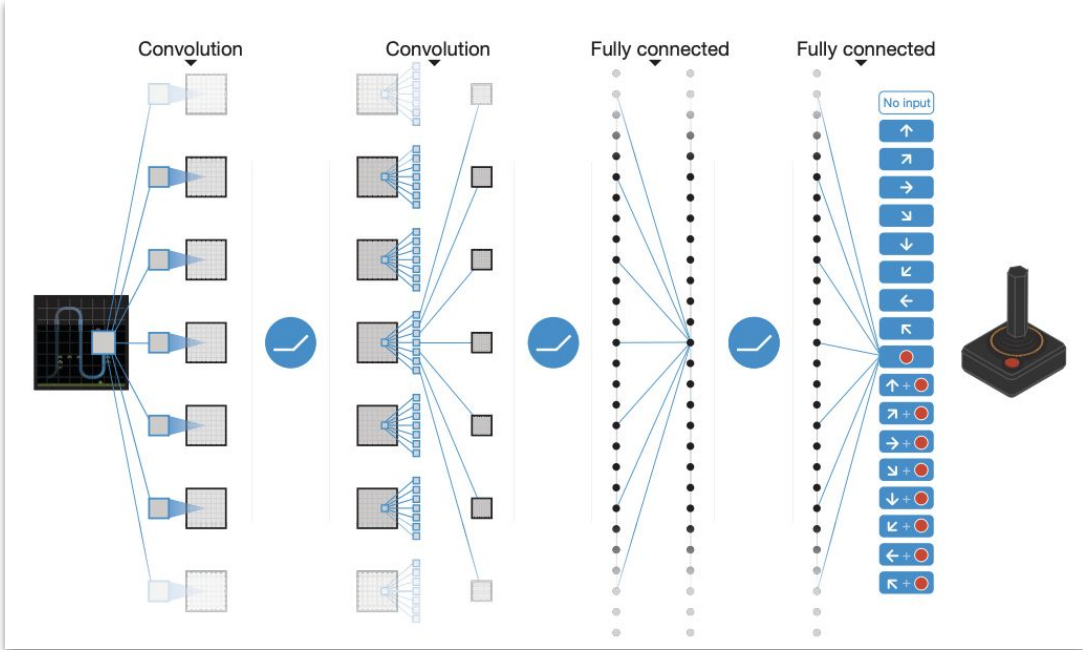
Dec 2013

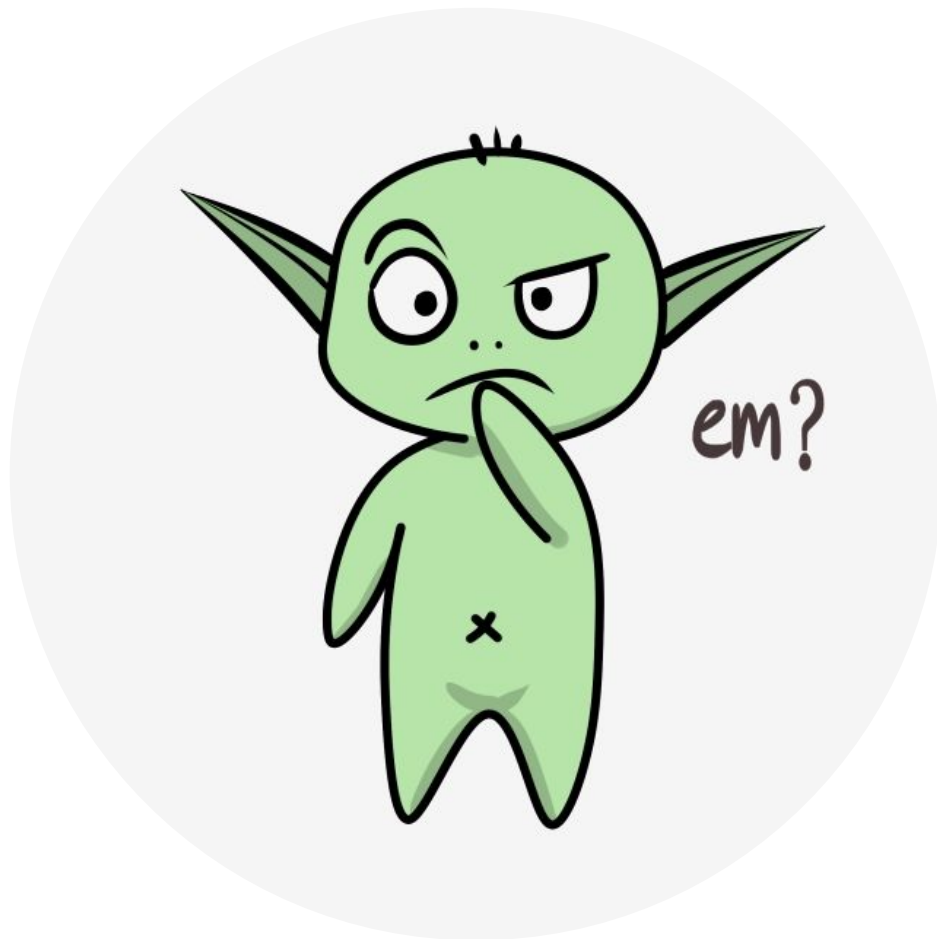




Deep Q-Network (and Deep RL)

[Mnih et al., 2013, 2015]





What if we were to design features instead?

What if we were to design features instead?

State of the Art Control of Atari Games Using Shallow Reinforcement Learning

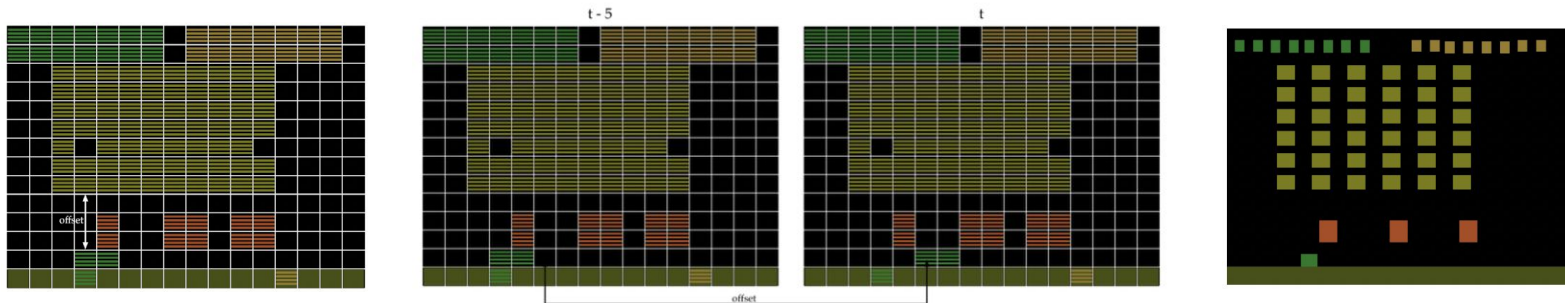
Yitao Liang[†], Marlos C. Machado[‡], Erik Talvitie[†], and Michael Bowling[‡]

[†]Franklin & Marshall College
Lancaster, PA, USA

[‡]University of Alberta
Edmonton, AB, Canada

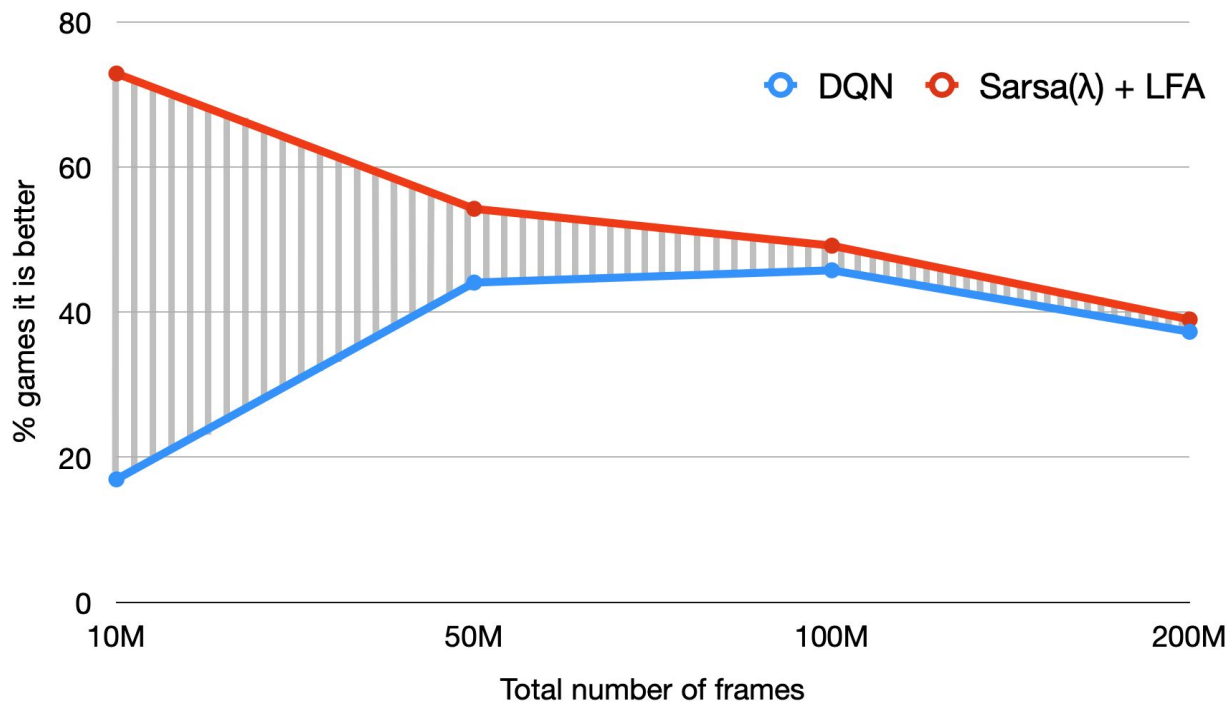
{yliang, erik.talvitie}@fandm.edu

{machado, mbowling}@ualberta.ca



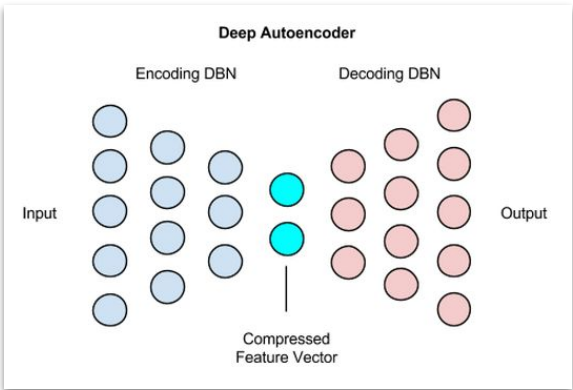
There are many trade-offs, we need to understand them

[Liang et al., 2016; Machado et al. 2018]





There are many many inductive biases one can use



<https://wiki.pathmind.com/deep-autoencoder>

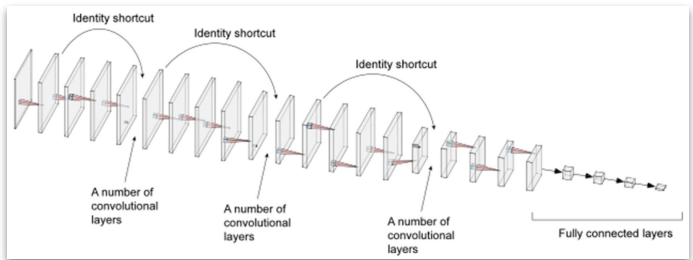
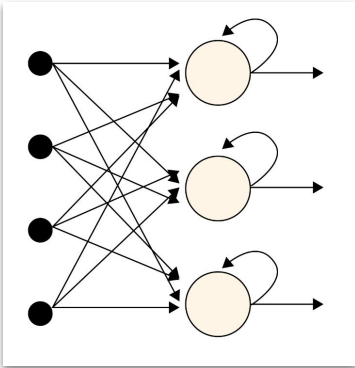


Image by Yu, Miao, and Wang (2022)



<https://www.scaler.com/topics/deep-learning/rnn/>

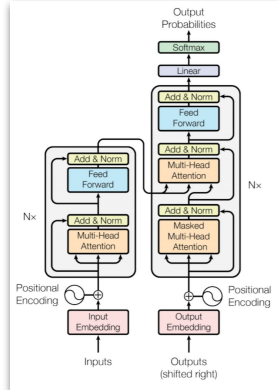


Image by Vaswani et al. (2017)