PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING

NEURAL NETWORK VISION FOR ROBOT DRIVING

ARJUN CHANDRASEKARAN

DEEP LEARNING AND PERCEPTION (ECE 6504)


ALMOST DONE
YOU CAN SEE THE LIGHT!!
makeameme.org

PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING


Attribution: Christopher T Cooper

NEURAL NETWORK VISION FOR ROBOT DRIVING

OUTLINE

- Playing Atari with Deep Reinforcement Learning
  - Motivation
  - Intro to Reinforcement Learning (RL)
  - Deep Q-Network (DQN)
  - BroadMind
- Neural Network Vision for Robot Driving

DEEPMIND

PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING

OUTLINE

- Playing Atari with Deep Reinforcement Learning
    - Motivation
    - Intro to Reinforcement Learning (RL)
    - Deep Q-Network (DQN)
    - BroadMind
- Neural Network Vision for Robot Driving

MOTIVATION

> AUTOMATICALLY CONVERT UNSTRUCTURED INFORMATION INTO USEFUL, ACTIONABLE KNOWLEDGE.

Demis Hassabis

Source: Nikolai Yakovenko

MOTIVATION

CREATE AN AI SYSTEM THAT HAS THE ABILITY TO LEARN FOR ITSELF FROM EXPERIENCE.

Demis Hassabis

Source: Nikolai Yakovenko

MOTIVATION

CAN DO STUFF THAT MAYBE WE DON'T KNOW HOW TO PROGRAM.

Demis Hassabis

Source: Nikolai Yakovenko

MOTIVATION

In short,

CREATE ARTIFICIAL GENERAL INTELLIGENCE

WHY GAMES

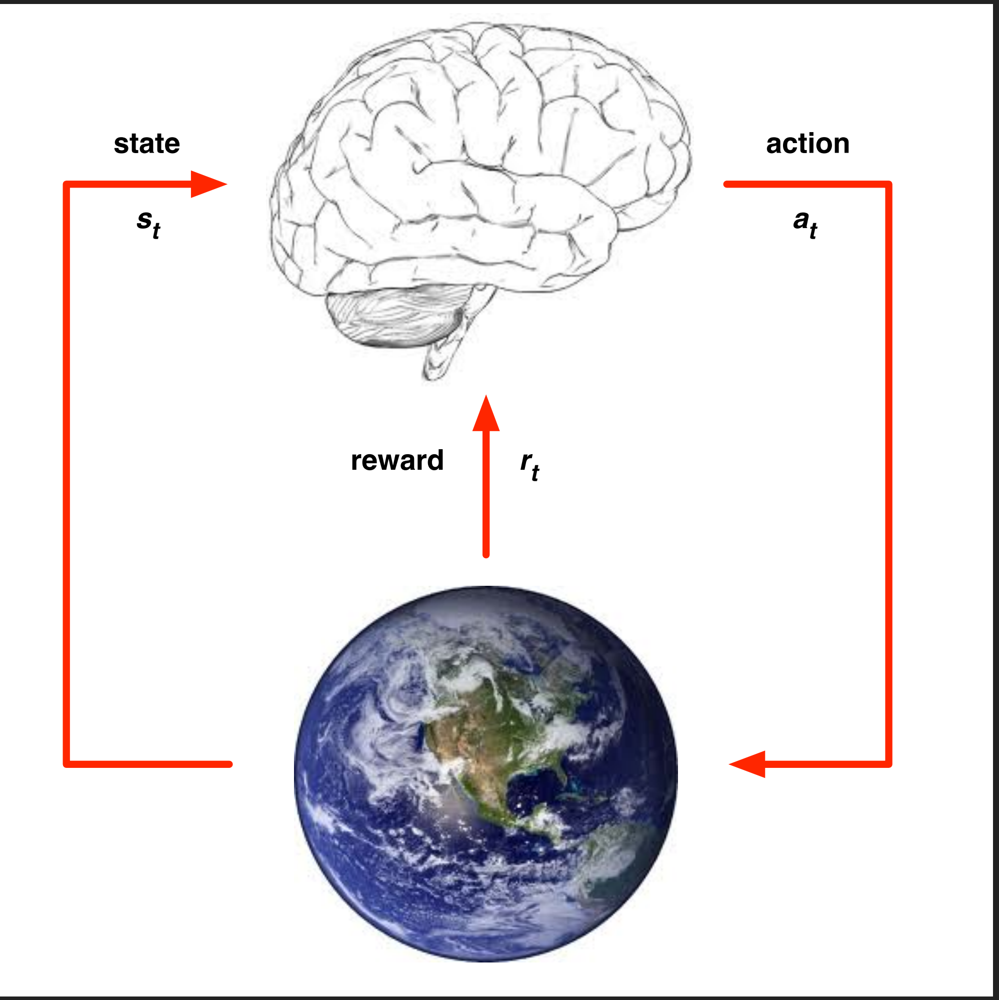
- Complexity.
- Diversity.
- Easy to create more data.
- Meaningful reward signal.
- Can train and learn to transfer knowledge between similar tasks.

Adapted from Nikolai Yakovenko

OUTLINE

- Playing Atari with Deep Reinforcement Learning
  - Motivation
  - Intro to Reinforcement Learning (RL)
  - Deep Q-Network (DQN)
  - BroadMind
- Neural Network Vision for Robot Driving

AGENT AND ENVIRONMENT



- At every time step t,
  - Agent executes action At
  - Receives observation Ot
  - Receives scalar reward Rt
- Environment
  - Receives action At
  - Emits observation Ot+1
  - Emits reward Rt+1

Source: David Silver

REINFORCEMENT LEARNING

- RL is a general-purpose framework for artificial intelligence
  - RL is for an agent with the capacity to act.
  - Each action influences the agent's future state.
  - Success is measured by a scalar reward signal
- RL in a nutshell:
  - Select actions to maximise future reward.

Source: David Silver

POLICY AND ACTION-VALUE FUNCTION

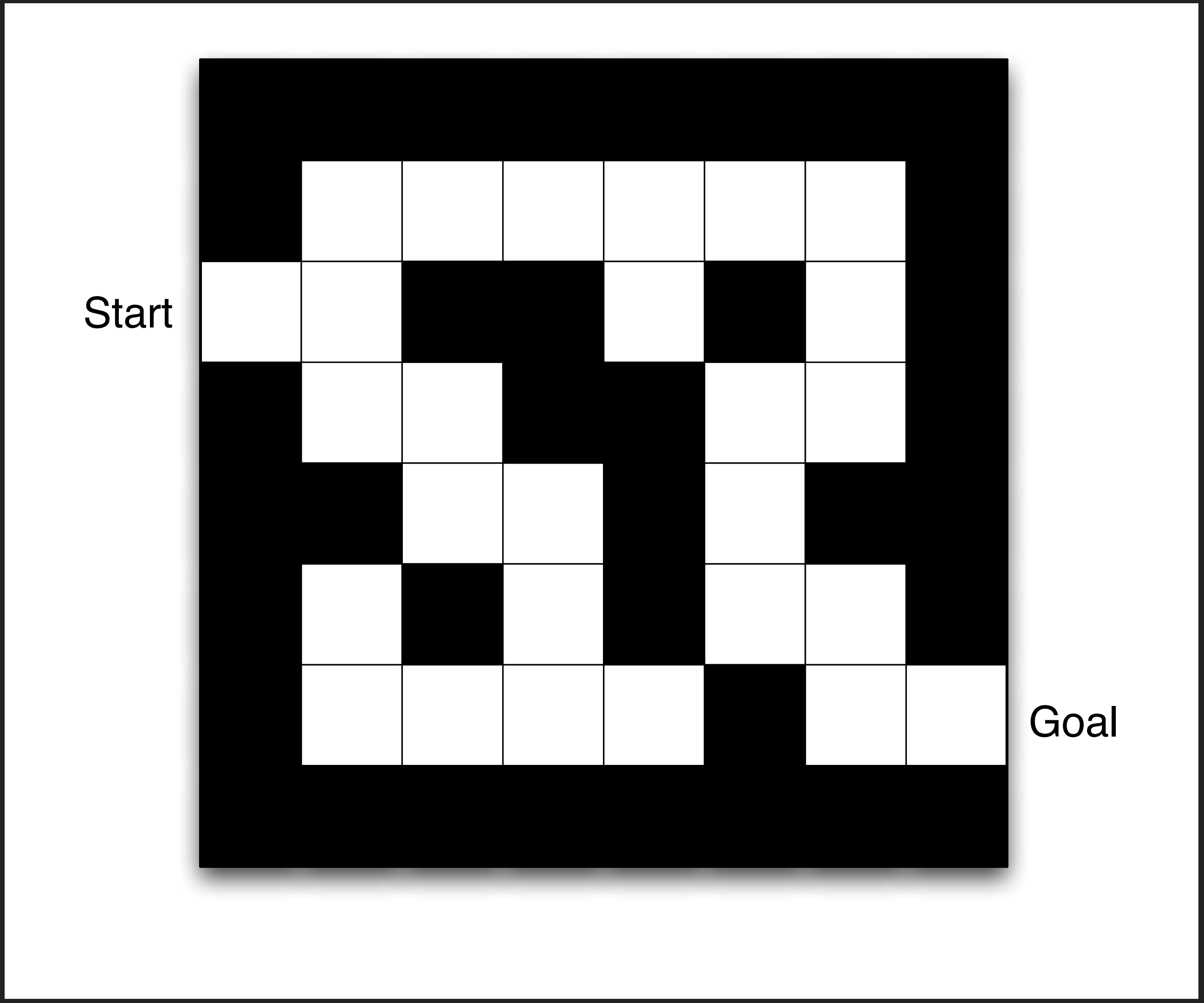
- Policy (Π) is a behavior function selecting actions given states: $a = \Pi(s)$
- Action-Value function $Q^{\Pi}(s, a)$ is the expected total reward from state s and action a under policy Π:
  - $Q^{\Pi}(s, a) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a]$
  - Indicates "how good is action a in state s"

Source: David Silver

Q FUNCTION / ACTION-VALUE FUNCTION

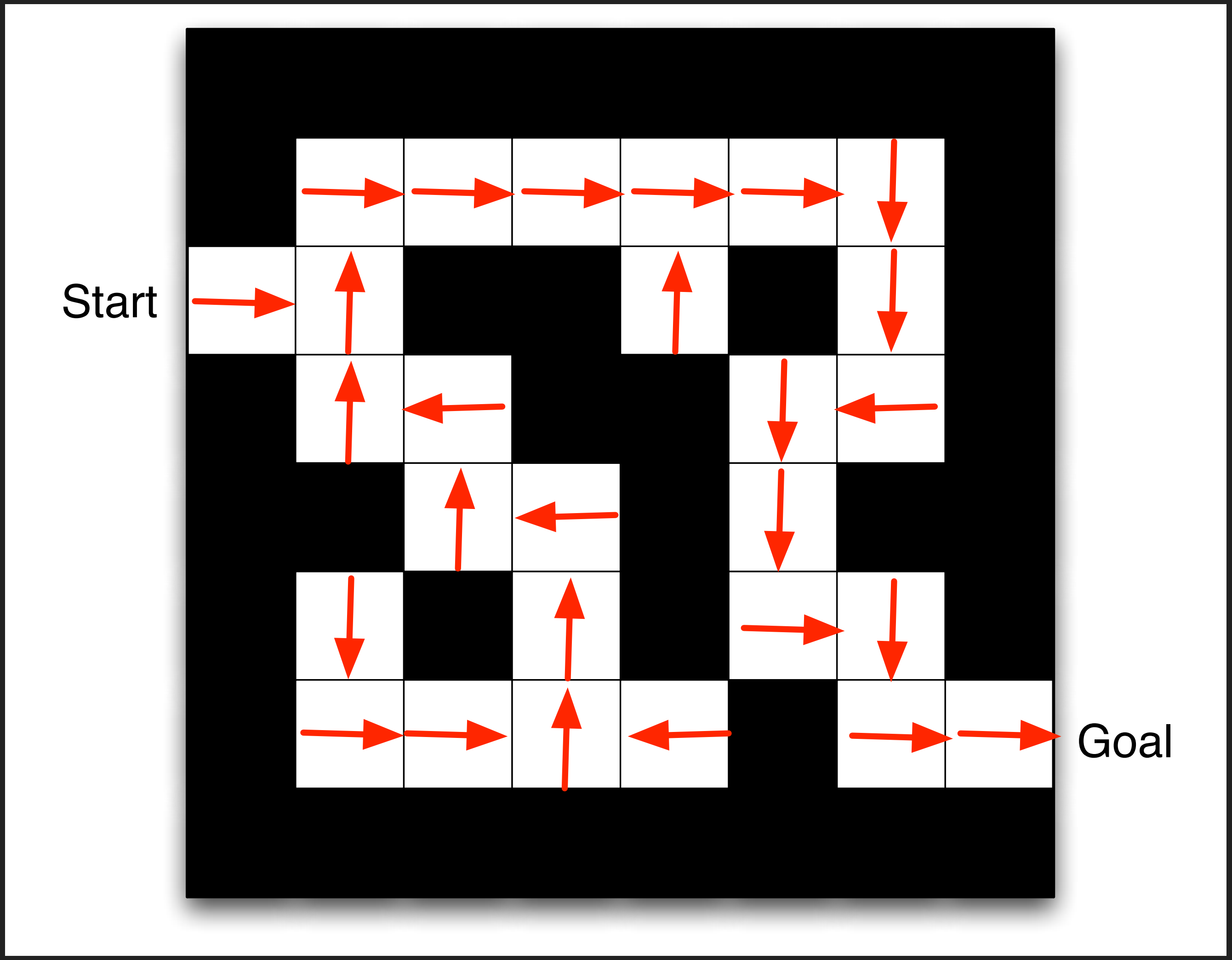
$$Q^{\Pi}(s, a) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a]$$

MAZE EXAMPLE



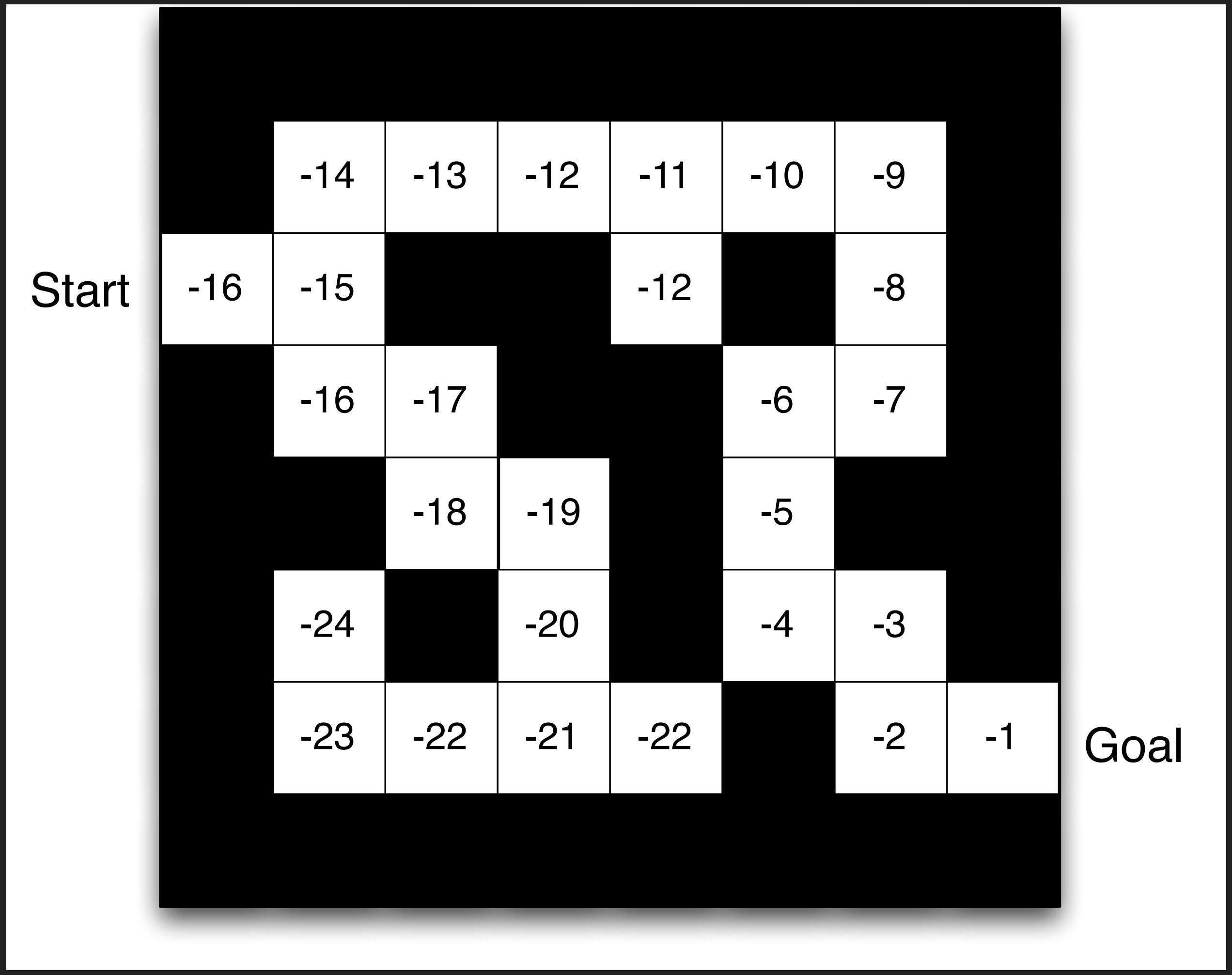
Source: David Silver

POLICY



Source: David Silver

VALUE FUNCTION: TO CHANGE PICTURE TO ACTION-VALUE FUNCTION



Source: David Silver

APPROACHES TO REINFORCEMENT LEARNING

- Policy-based RL
  - Search directly for the optimal policy $\prod^*$
  - This is the policy achieving maximum future reward
- Value-based RL
  - Estimate the optimal value function $Q^*(s,a)$
  - This is the maximum value achievable under any policy
- Model-based RL
  - Build a transition model of the environment
  - Plan (e.g. by lookahead) using model

Source: David Silver

APPROACHES TO REINFORCEMENT LEARNING

- Policy-based RL
  - Search directly for the optimal policy ∏*
  - This is the policy achieving maximum future reward
- Value-based RL
  - Estimate the optimal value function Q*(s,a)
  - This is the maximum value achievable under any policy
- Model-based RL
  - Build a transition model of the environment
  - Plan (e.g. by lookahead) using model

OUTLINE

- Playing Atari with Deep Reinforcement Learning
    - Motivation
    - Intro to Reinforcement Learning (RL)
    - Deep Q-Network (DQN)
    - BroadMind
- Neural Network Vision for Robot Driving

DEEP REINFORCEMENT LEARNING

- How to apply reinforcement learning to deep neural networks?
  - Use a deep network to represent value function/policy/model.
  - Optimize this value function/policy/model end-to-end.
  - Use SGD to learn the weights/parameters.

Source: David Silver

UNROLLING RECURSIVELY…

- Value function can be unrolled recursively

$Q^{\pi}(s, a) = E[r + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \mid s, a] = E_{s'}[r + \gamma Q^{\pi}(s', a') \mid s, a]$

- Optimal value function $Q^*(s,a)$ can be unrolled recursively

$Q^*(s, a) = E_{s'}[r + \gamma \max_{a'} Q^*(s', a') \mid s, a]$

- Value iteration algorithms solve the Bellman equation

$Q_{i+1}(s, a) = E_{s'}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a]$

Source: David Silver

DEEP Q-LEARNING

- Represent action-value function using a deep Q-network with weights w:

$Q(s, a, w) \sim Q^{\pi}(s, a)$

- Loss is the mean squared error defined in Q-values:

$L(w) = E[(r + \gamma \max_{a'} Q(s', a', w^{-}) - Q(s, a, w))^2]$

- Gradient

$\partial L(w)/\partial w = E[(r + \gamma \max_{a'} Q(s', a', w^{-}) - Q(s, a, w))^2] * \partial Q(s, a, w)/\partial w$

Source: David Silver

STABILITY ISSUES WITH DEEP RL

- Naive Q-learning oscillates or diverges with neural nets
  - Data is sequential
    - Successive samples are correlated, non-iid
  - Policy changes rapidly with slight changes to Q-values
    - Policy may oscillate
    - Distribution of data can swing from one extreme to another
- Scale of rewards and Q-values is unknown
  - Naive Q-learning gradients can be large unstable when backpropagated

Source: David Silver

DEEP Q-NETWORKS

- DQN provides a stable solution to deep value-based RL
  - Use experience replay
    - Break correlations in data, bring us back to iid setting
    - Learn from all past policies
- Freeze target Q-network
  - Avoid oscillations
  - Break correlations between Q-network and target
- Clip rewards or normalize network adaptively to sensible range
  - Robust gradients

Source: David Silver

TRICK 1 – EXPERIENCE REPLAY

- To remove correlations, build dataset from agent's own experience
  - Take action at according to €-greedy policy
  - Store transition (s_t, a_t, r_{t+1}, s_{t+1}) in replay memory D
  - Sample random mini-batch of transitions (s, a, r, s') from D
  - Minimize MSE between Q-network and Q-learning targets

Source: David Silver

TRICK 2 – FIXED TARGET Q-NETWORK

- To avoid oscillations, fix parameters used in Q-learning target
  - Compute Q-learning targets w.r.t. old, fixed parameters w^{-}

$$r + \gamma \max_{a'} Q(s', a', w^{-})$$

- Minimize MSE between Q-network and Q-learning targets

$$L(w) = E_{s,a,r,s' \sim D} \left[\left(r + \gamma \max_{a'} Q(s', a', w^{-}) - Q(s,a,w) \right)^2 \right]$$

- Periodically update fixed parameters $w^{-} \leftarrow w$

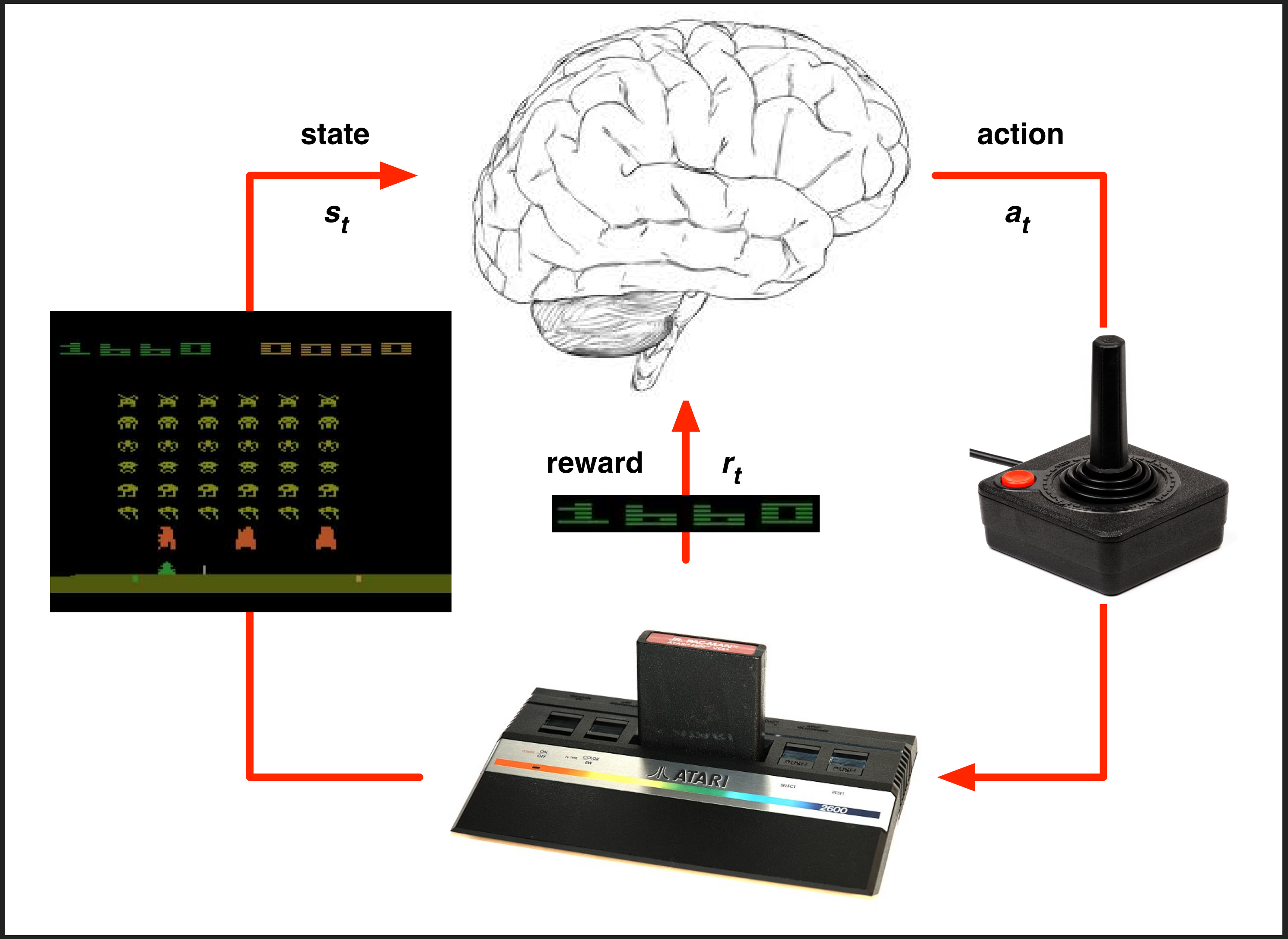
Source: David Silver

TRICK 3 – REWARD/VALUE RANGE

- Advantages
  - DQN clips the rewards to $[-1,+1]$
  - This prevents Q-values from becoming too large
  - Ensures gradients are well-conditioned
- Disadvantages
  - Can't tell difference between small and large rewards

Source: David Silver

BACK TO BROADMIND



Source: David Silver

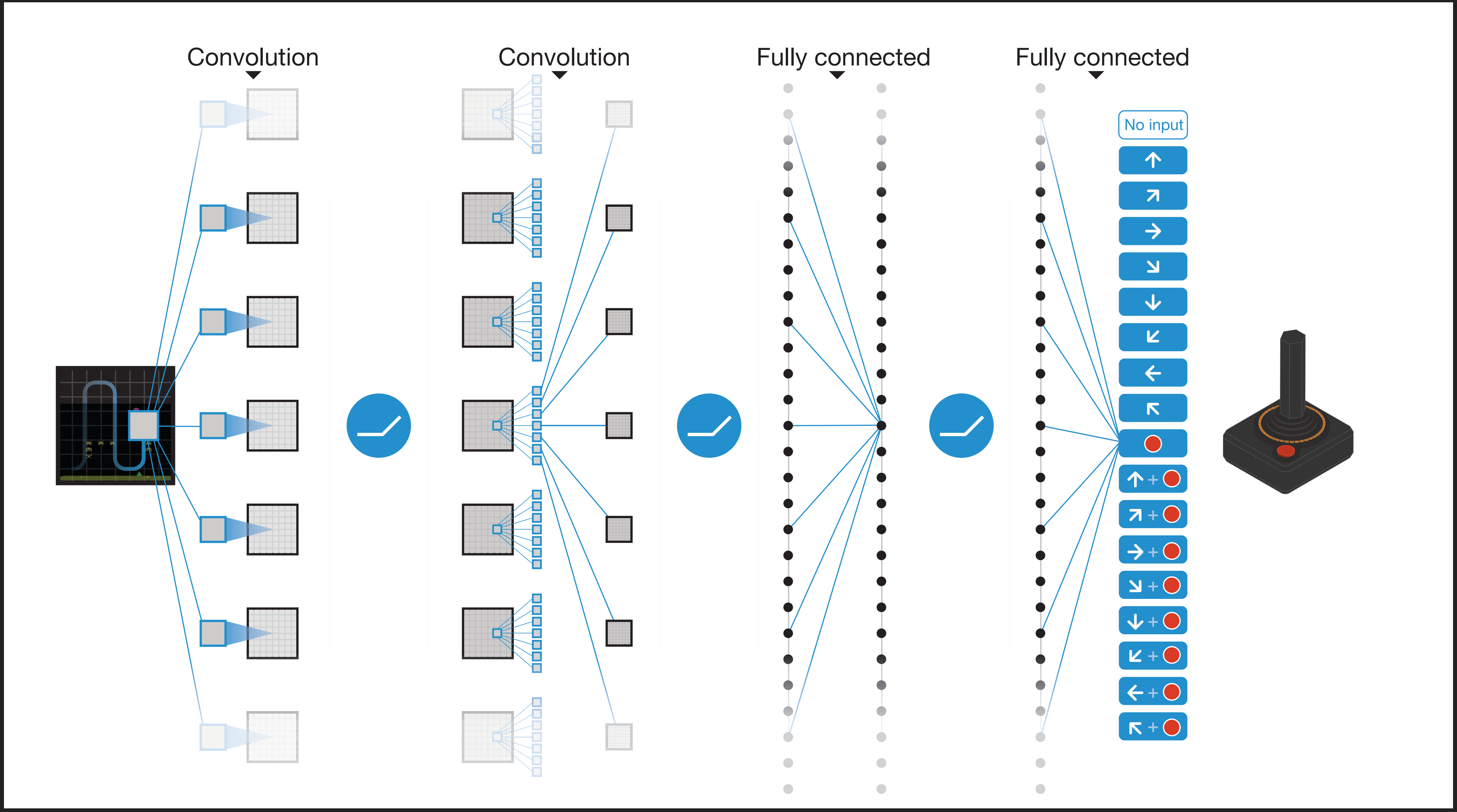
INTRODUCTION - ATARI AGENT (AKA BROADMIND)

- Aim to create a single neural network agent that is able to successfully learn to play as many of the games as possible.
- Agent plays 49 Atari 2600 arcade games.
- Learns strictly from experience - no pre-training.
- Inputs: game screen + score.
- No game-specific tuning.

INTRODUCTION – ATARI AGENT (AKA BROADMIND)

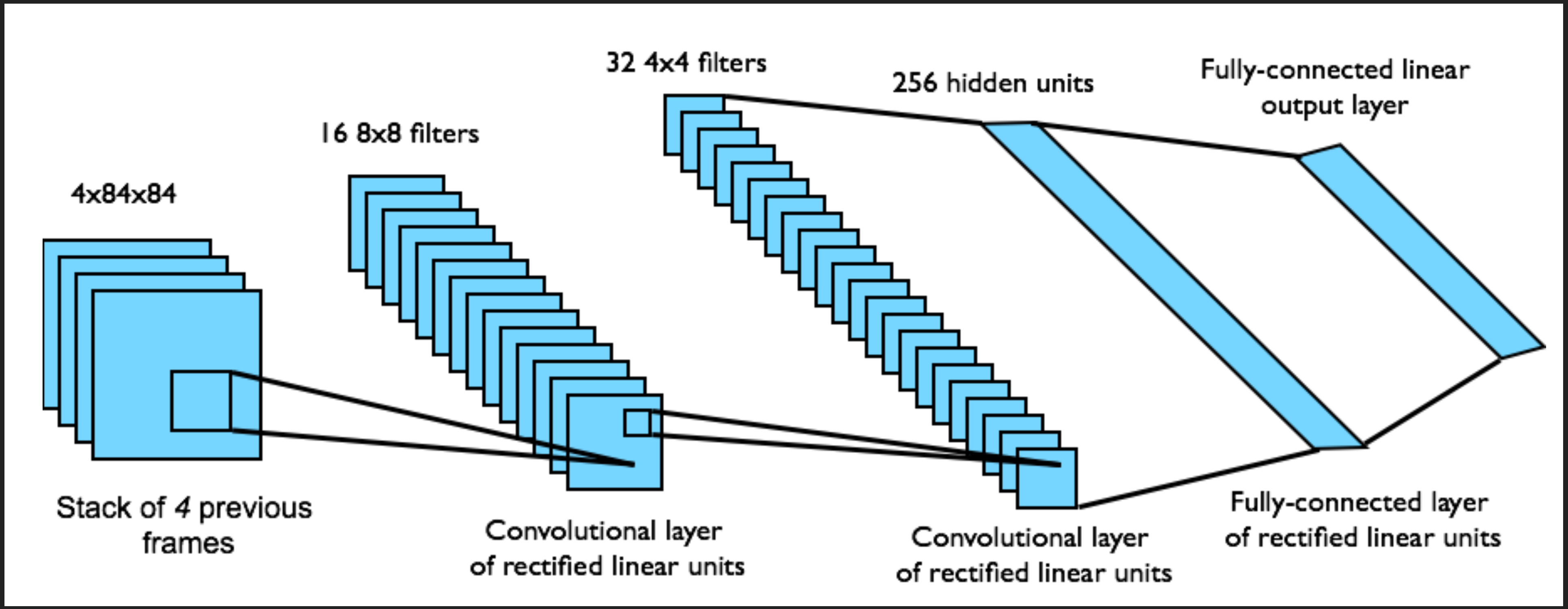
- State – screen transitions from a sequence of 4 frames.
  - Screen is 210*160 pixels with 128 color palette
- Actions – 18 corresponding to:
  - 9 directions of joystick (including no input).
  - 9 directions + button.
- Reward – Game score.

SCHEMATIC OF NETWORK



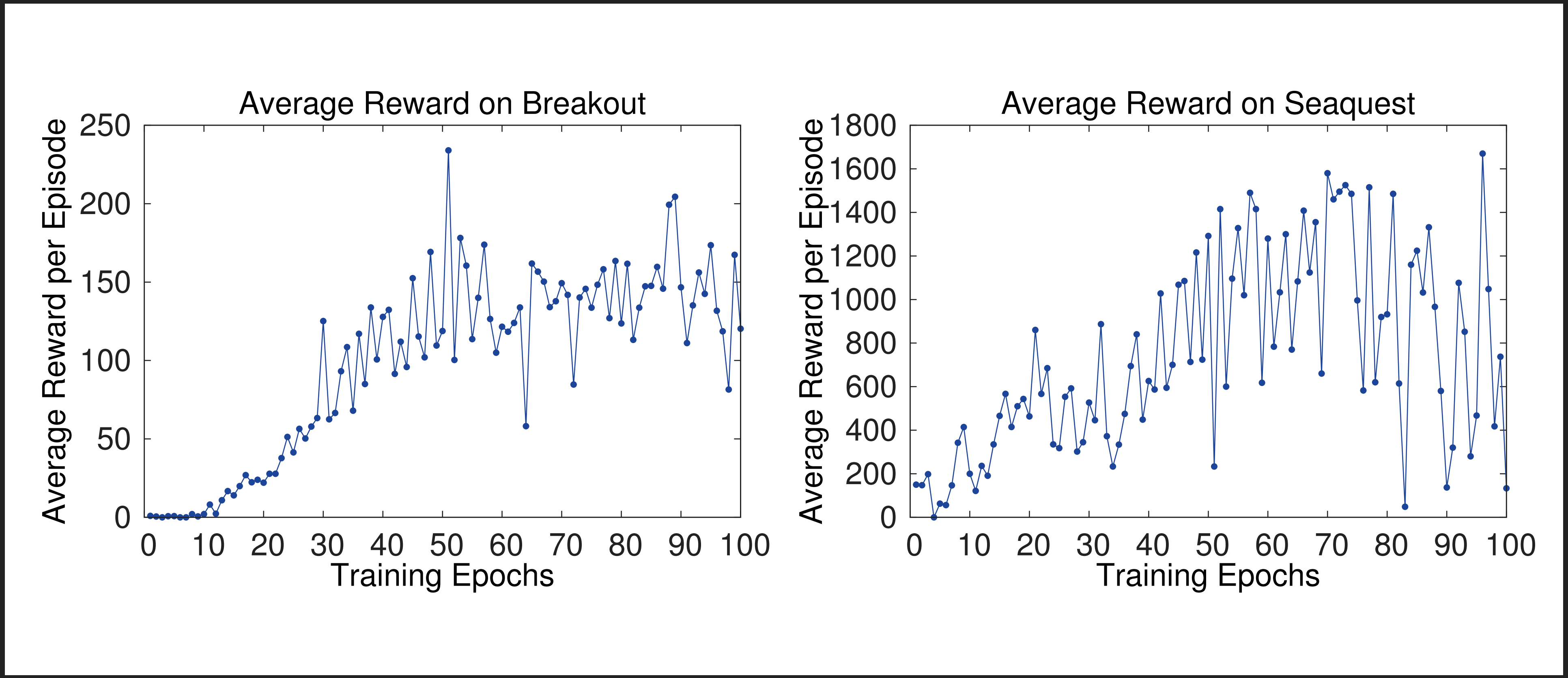
Mnih et. al.

NETWORK ARCHITECTURE



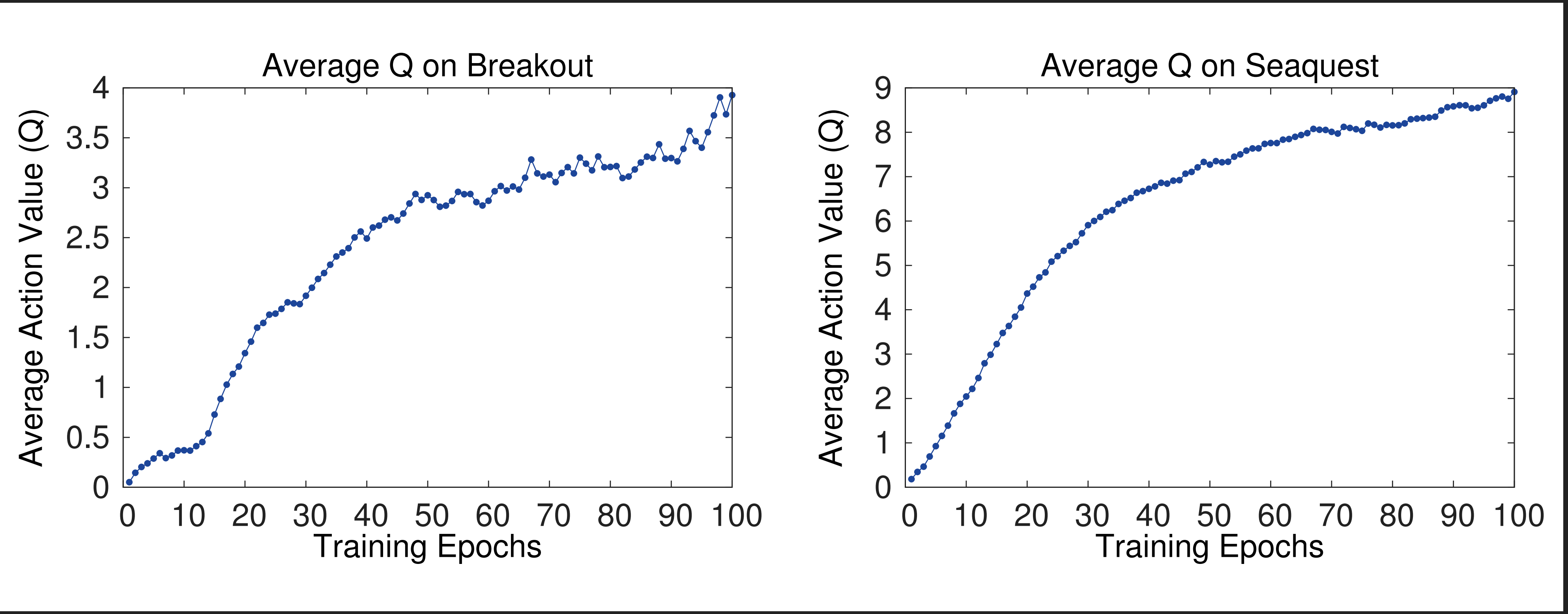
Mnih et. al.

EVALUATION



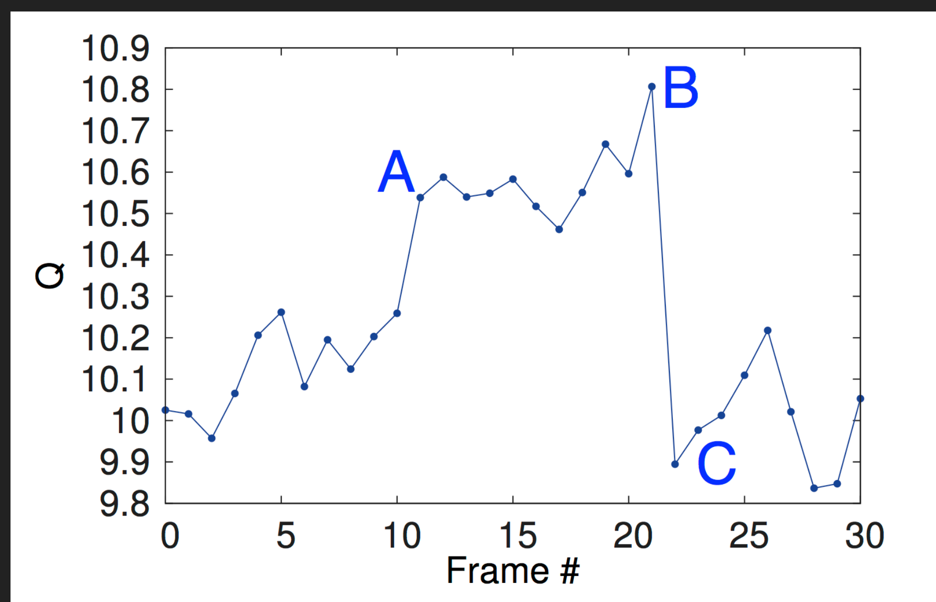
Mnih et. al.

EVALUATION



Mnih et. al.

EVALUATION



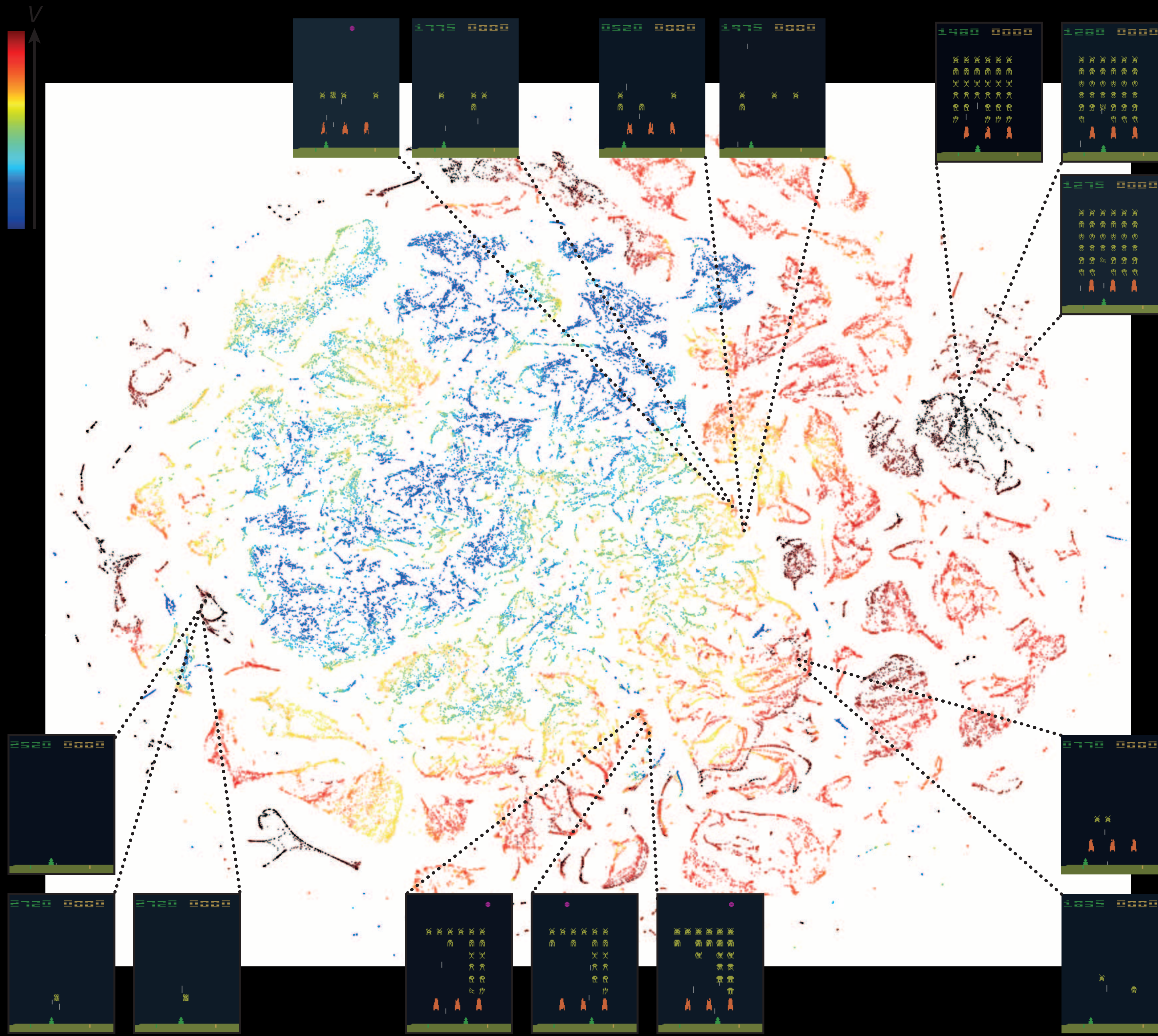
A

B

C

Mnih et. al.

VISUALIZATION OF GAME STATES IN LAST HIDDEN LAYER



Mnih et. al.

AVERAGE TOTAL REWARD

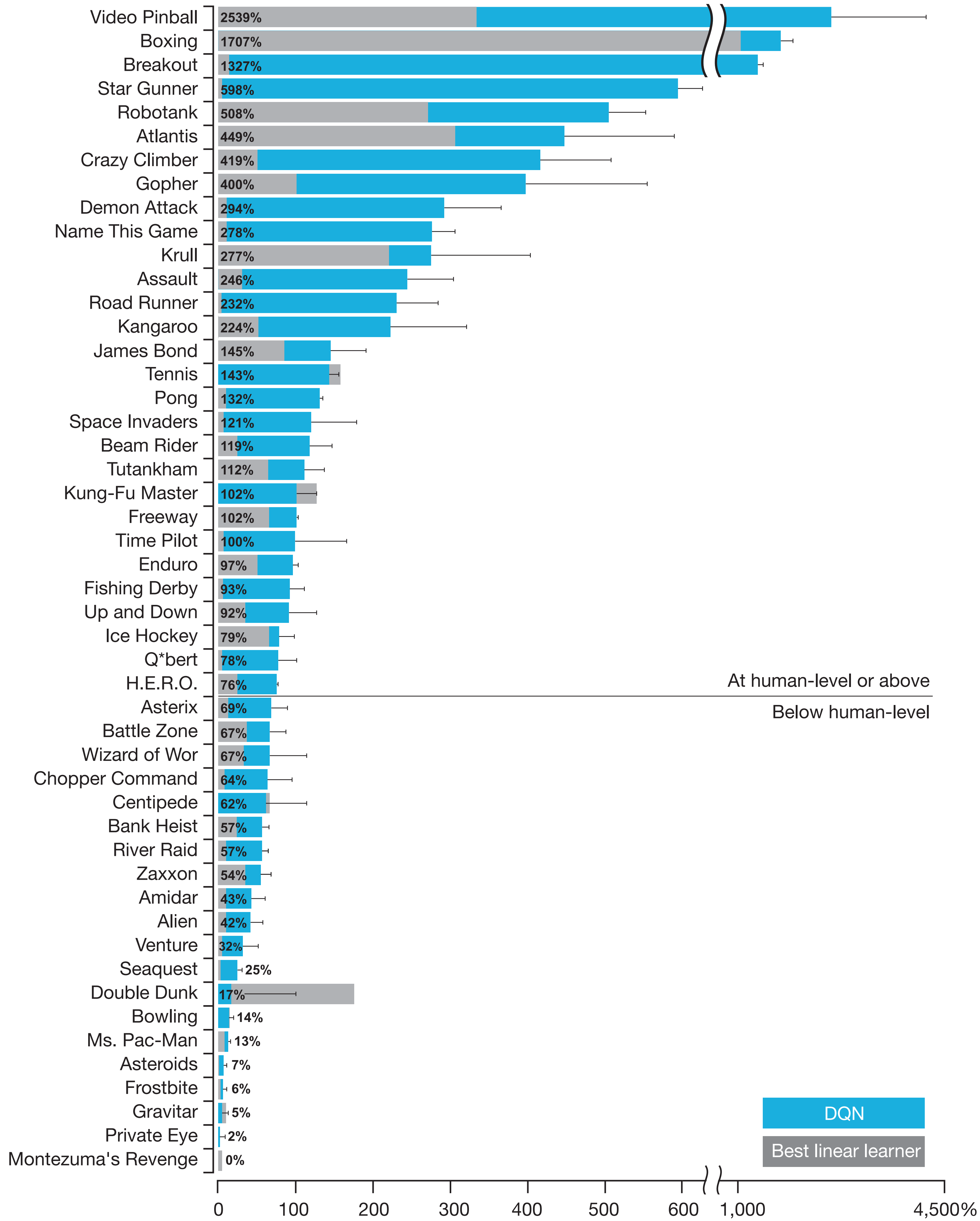
	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690

SINGLE BEST PERFORMING EPISODE

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Mnih et. al.

DQN PERFORMANCE

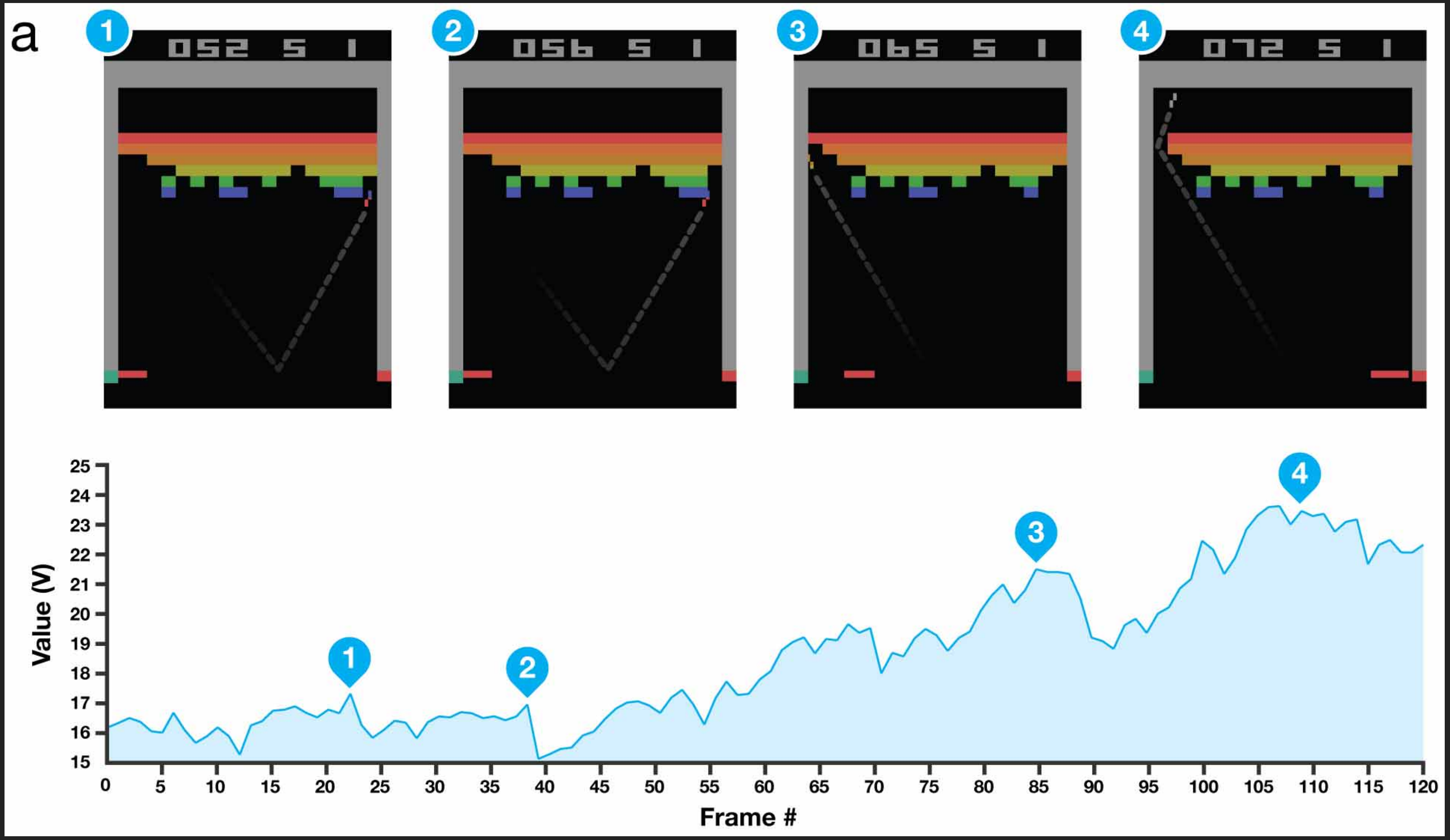


Mnih et. al.

BROADMIND LEARNS OPTIMAL STRATEGY

https://www.youtube.com/watch?v=rbsqaJwpu6A

VISUALIZATION OF VALUE FUNCTION



Mnih et. al.

STRENGTHS AND WEAKNESSES

- Good at
  - Quick-moving, complex, short-horizon games
  - Semi-independent trails within the game
  - Negative feedback on failure
  - Pinball
- Bad at:
  - long-horizon games that don’t converge
  - Any “walking around” game
  - Pac-Man

Source: Nikolai Yakovenko

FAILURE CASES

- Montezuma's revenge
  - Single reward at the end of the level. No intermediate rewards
  - Worldly knowledge helps humans play these games relatively easily.
  - https://www.youtube.com/watch?v=1rwPI3RG-lU

JUERGEN SCHMIDHUBER'S TEAM

Evolving Large-Scale Neural Networks for Vision-Based Reinforcement Learning

- Evolutionary Computation based deep NN for RL
- Learns to play a car-racing video game
- No pre-training or hand-coding of features
- Video

RELATED TOPICS/PAPERS

- Universal Value Function Approximators, DeepMind
  - http://jmlr.org/proceedings/papers/v37/schaul15.pdf
- Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning , UMich
  - http://papers.nips.cc/paper/5421-deep-learning-for-real-time-atari-game-play-using-offline-monte-carlo-tree-search-planning
- On Learning to Think: Algorithmic Information Theory for Novel Combinations of Reinforcement Learning Controllers and Recurrent Neural World Models, Juergen Schmidhuber
  - http://arxiv.org/abs/1511.09249

DEAN POMERLEAU

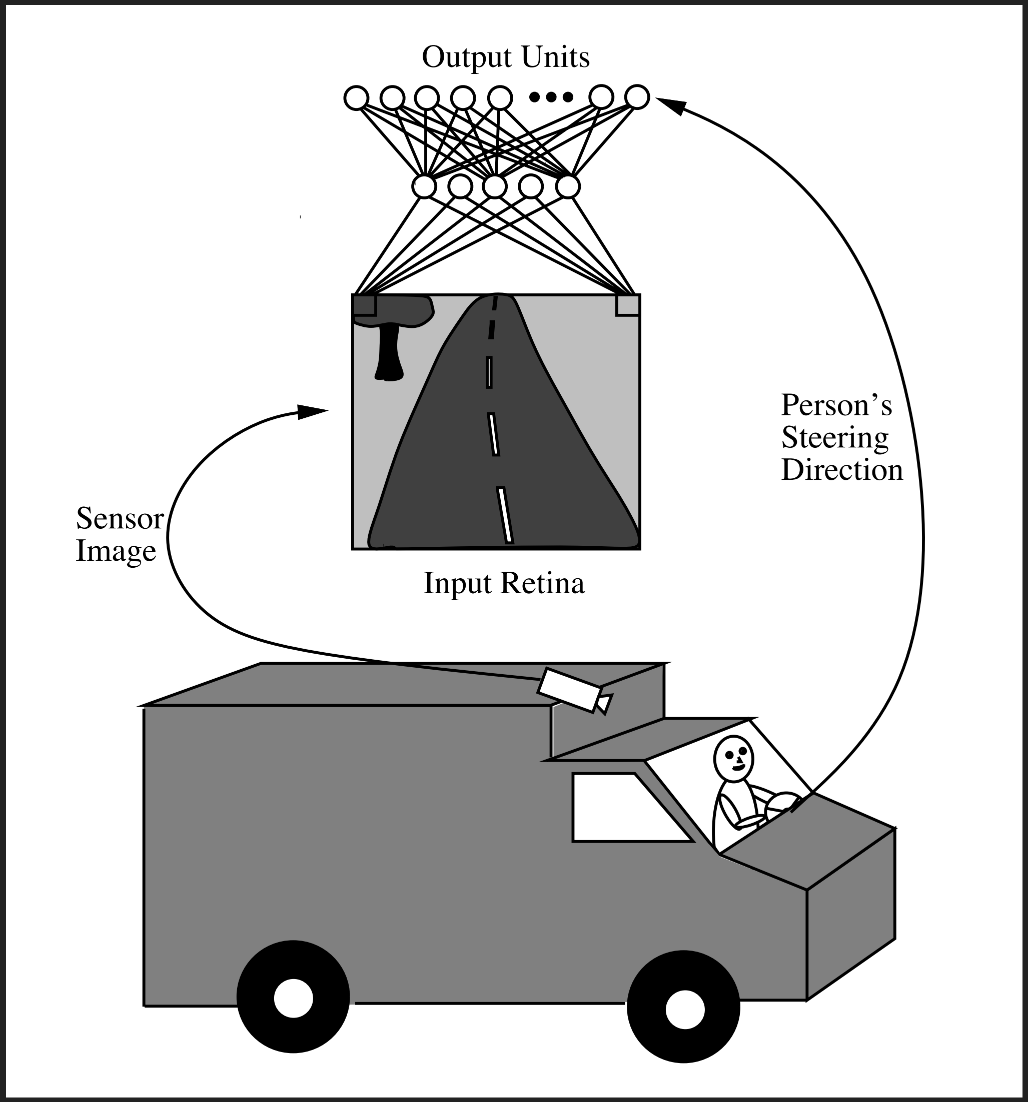
NEURAL NETWORK VISION FOR ROBOT DRIVING

ALVINN – AUTONOMOUS DRIVING SYSTEM

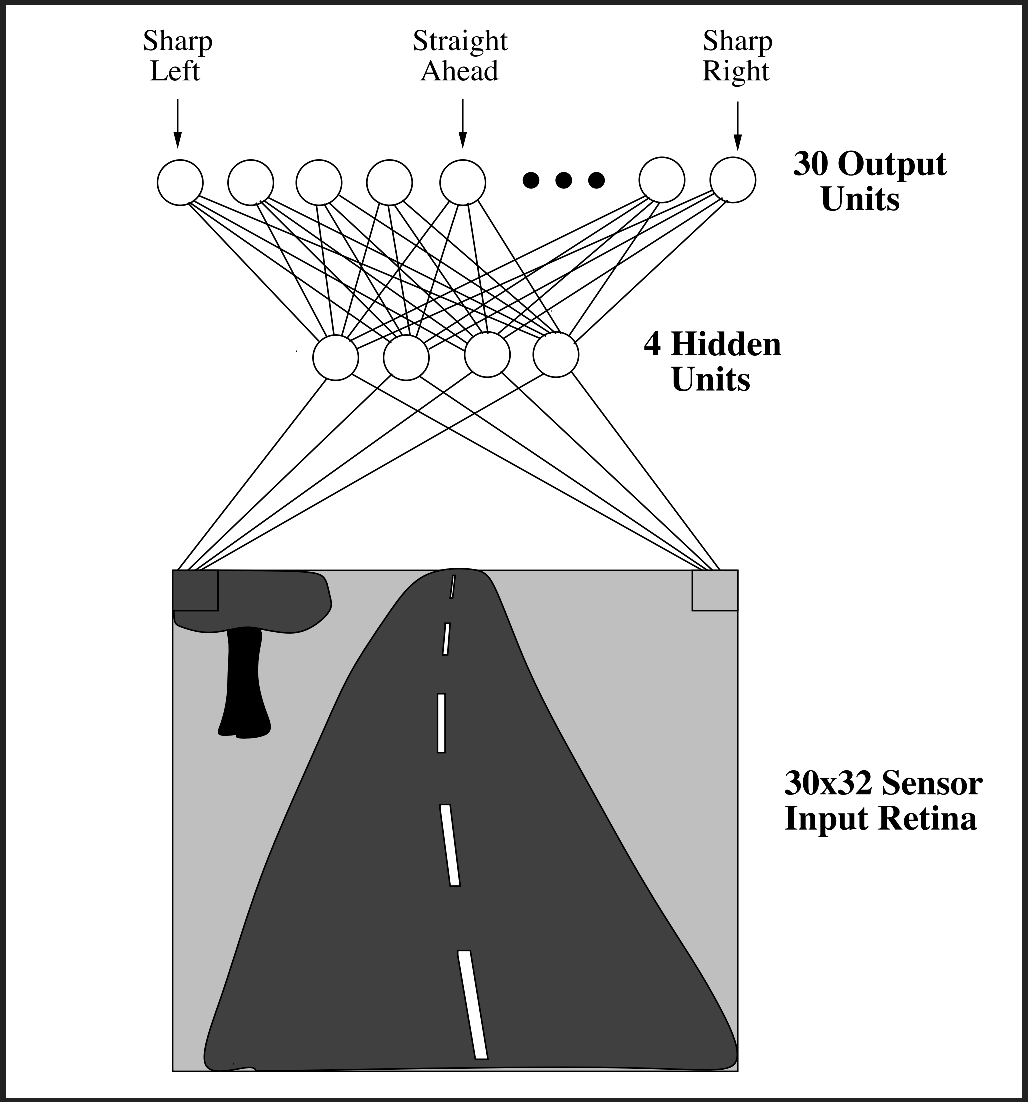
- ALVINN has successfully driven autonomously at speeds of up to 70 mph, and for distances of over 90 miles on a public highway north of Pittsburgh.

- Multiple NNs trained to handle: single lane dirt roads, single lane paved bike paths, two lane suburban neighborhood streets, and lined two lane highways.

SCHEMATIC OF LEARNING ON THE FLY



NN ARCHITECTURE FOR AUTONOMOUS DRIVING

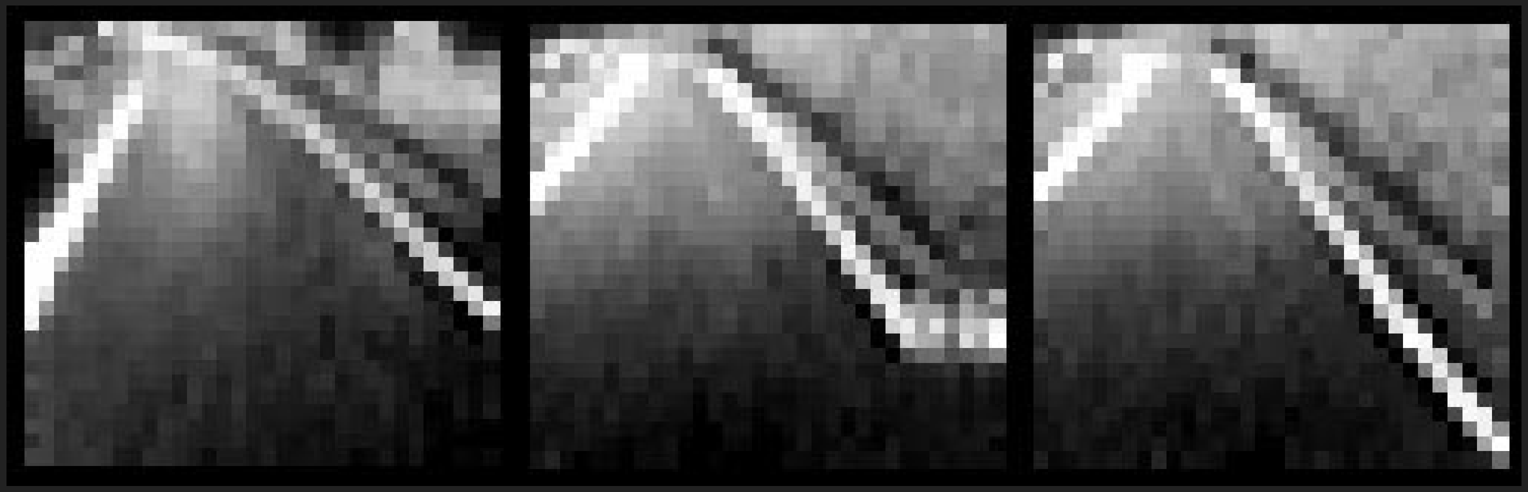


ARCHITECTURE

- 1-hidden layer NN.
- Input layer contains 960 neurons.
- 1 hidden layer containing 4 neurons.
- Output layer contains 30 neurons.

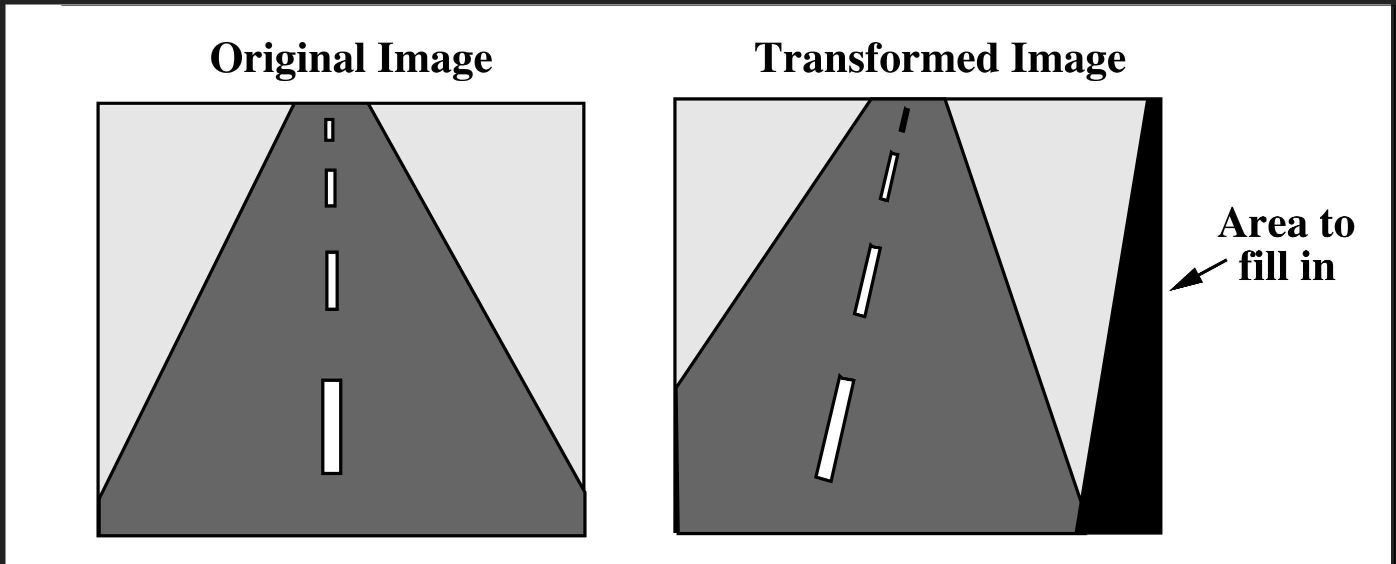
INPUT

- Input “retina” of size 30*32 can take down sampled input from video camera/scanning laser.
  - These days, LIDAR is commonly used to generate a 3D point cloud of the observed environment.
- Affine transforms of input image to augment training set.

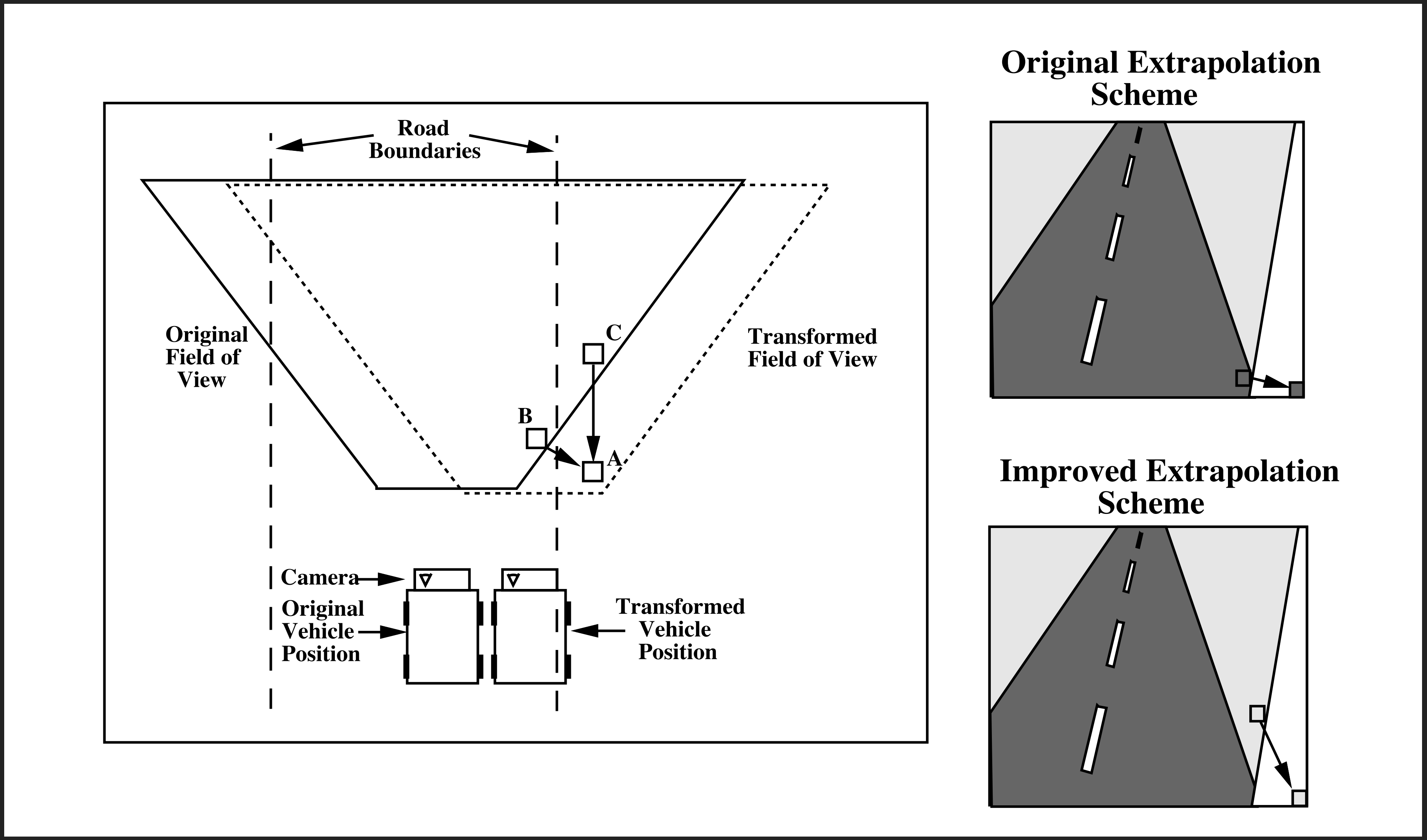


DATA AUGMENTATION

- Potential issues:
  - Misalignment errors never seen during training.
  - Lack of diversity in training set.
- Solution: Transform original image to augment training set.



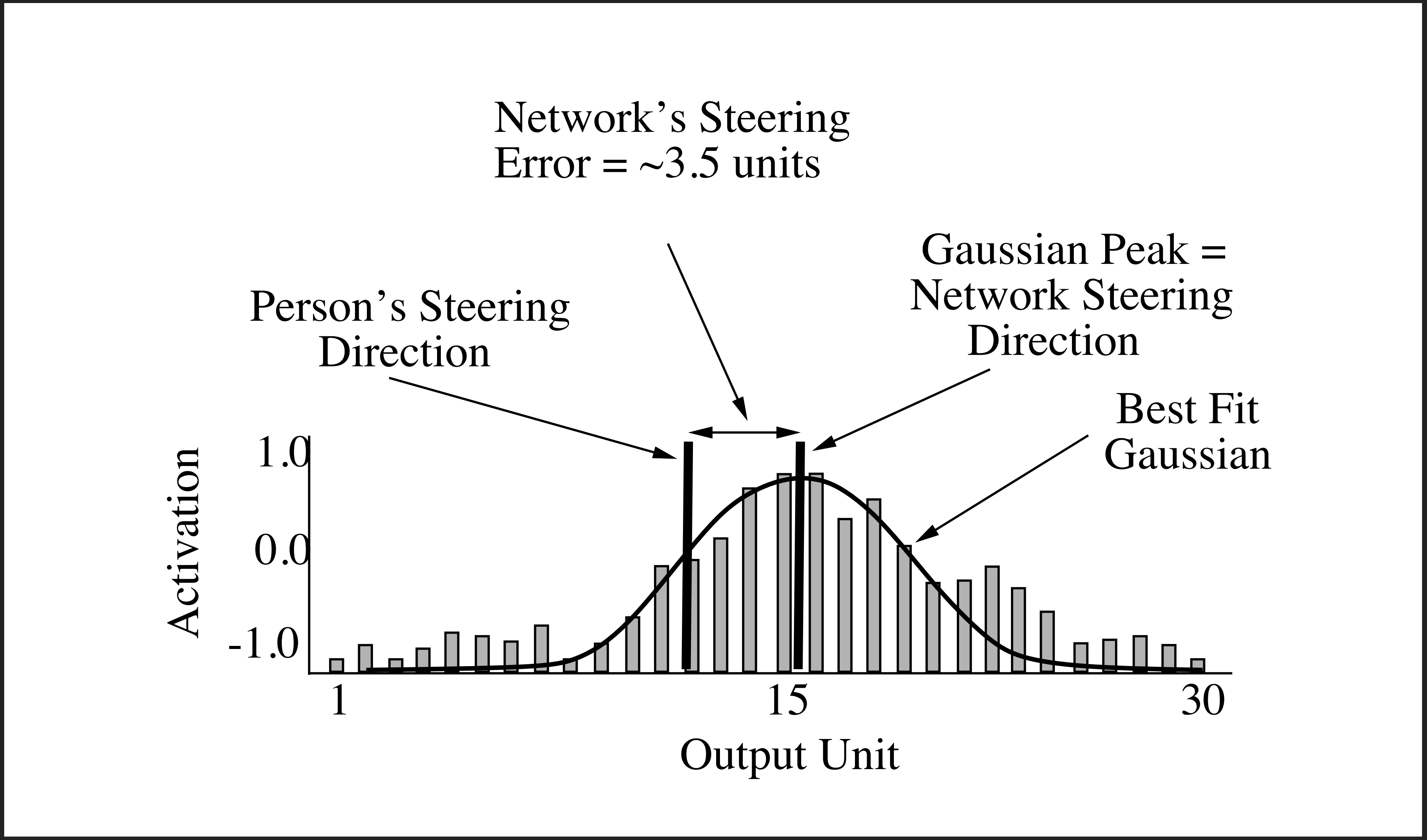
EXTRAPOLATION



OUTPUT

- Networks output the correct direction to steer, and a confidence score.
- Output from network with highest confidence is chosen.
- Direction to steer is the center of mass of “hill of activation”.

STEERING ERROR

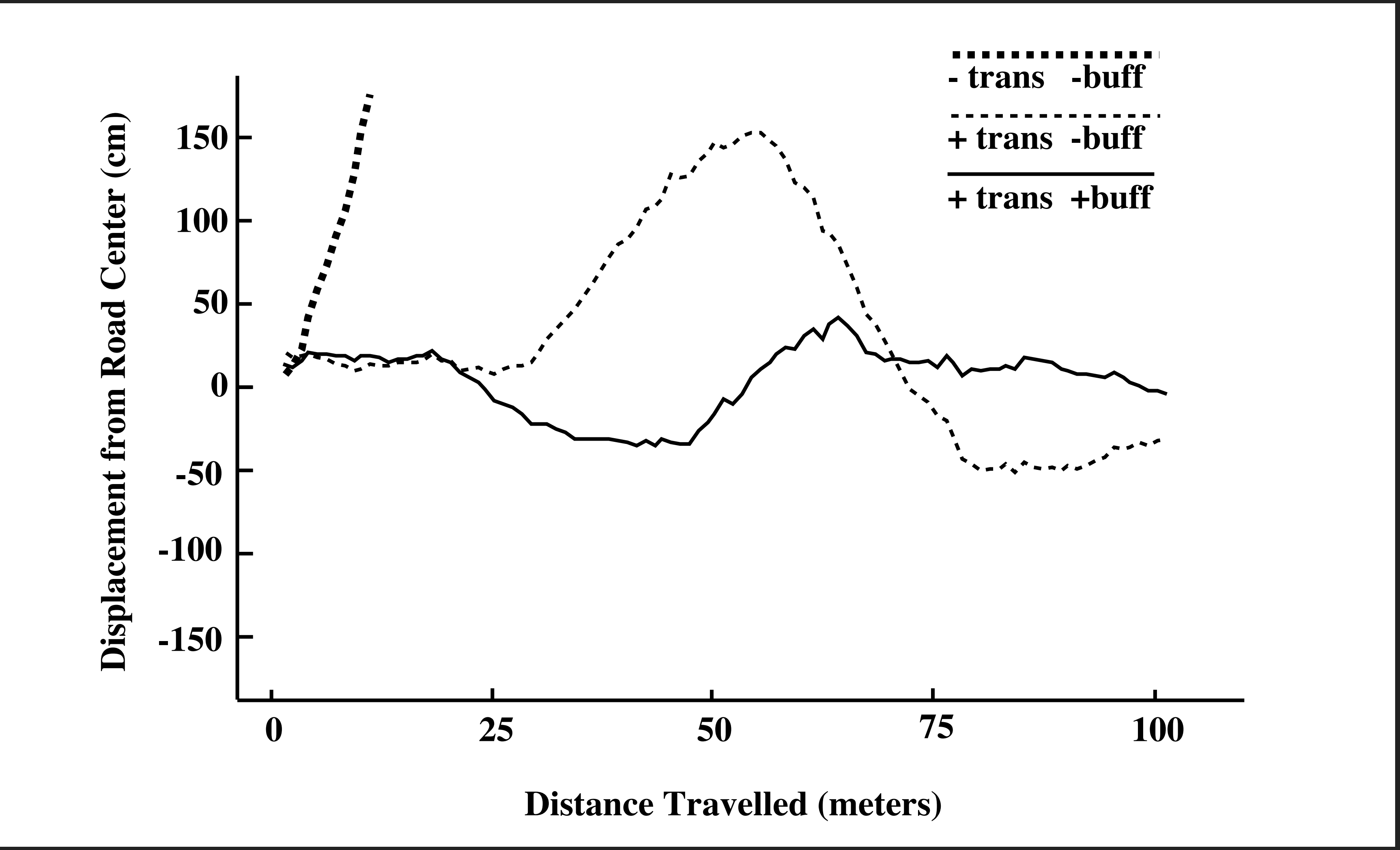


TRAINING

- Original sensor image is shifted and rotated to create 14 training exemplars.
- Buffer of 200 exemplar patterns used to train the network.
- Each exemplar is replaced with another with a constant probability to ensure diversity.
- 2.5sec per training cycle. Total training time = 4 min.

PERFORMANCE

Low value throughout is better.



ALVINN

- ALVINN (1995)
  - https://www.youtube.com/watch?v=ilP4aPDTBPE

TAKEAWAY

- Creating AGI is hard.
- Tangible first step.
- RNNAIs, Memory Networks + RL etc. promise an exciting future

THANK YOU!