# Introduction to Deep Learning

## Advanced Topics

Prof. Songhwai Oh

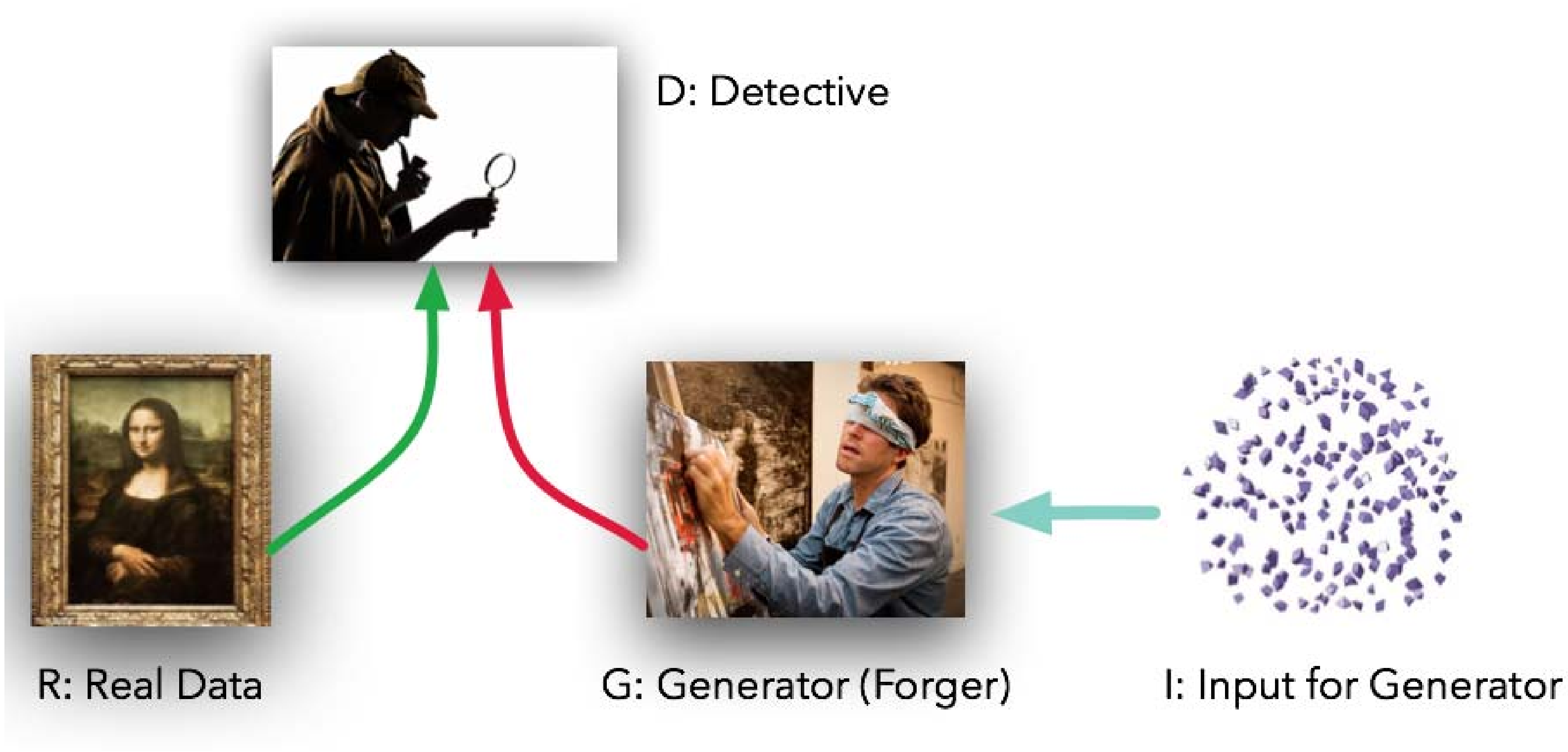
ECE, SNU

# Generative Adversarial Networks (GAN)

# Generative Adversarial Networks

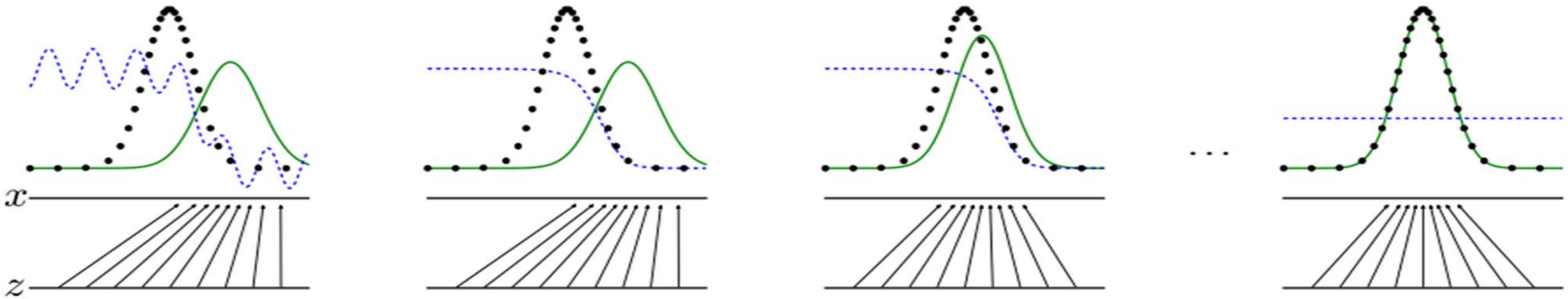
## How can we generate more realistic images?

Our Answer: **Generative Adversarial Network (GAN)**

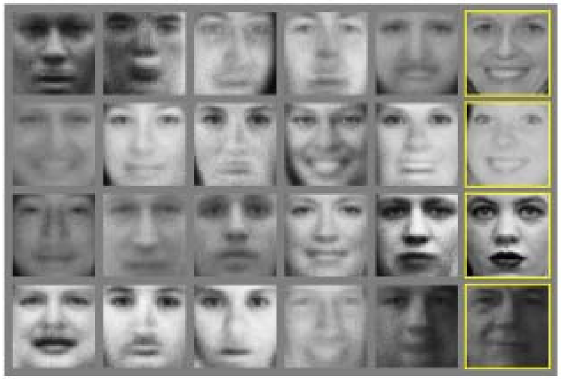



(Source: Dev Nag, Medium)

# Generative Adversarial Networks (GAN)

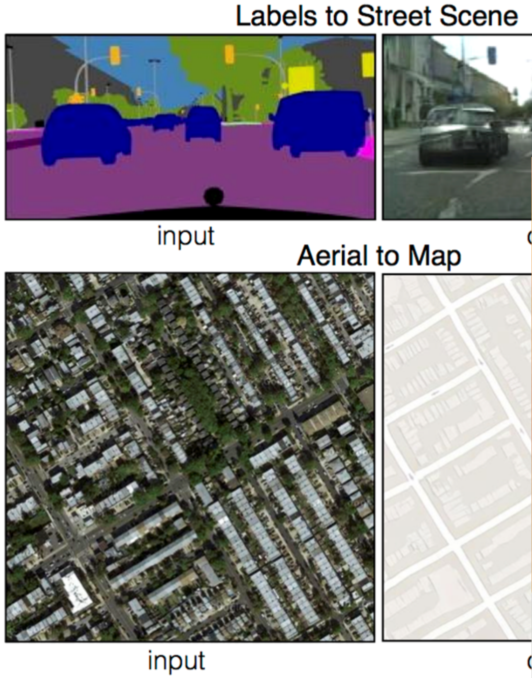



- **Discriminator** (blue dashed line)
  - Discriminates between samples from the **data generating distribution** (black dotted line) from those of the **Generator** (green solid line)
- **Generator** (green solid line)
  - The objective of the **Generator** is to learn the **data generating distribution** so that it can deceive the **Discriminator**
- Generative adversarial nets are trained by making a competition between the **Generator** and the **Discriminator**.
- The theoretical results from [4] shows that the **Generator** converge to a good estimator of the data generating distribution, and the **Discriminator** fails to discriminate the real and fake data.



[4] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "**Generative adversarial nets**." In *Advances in neural information processing systems*, pp. 2672-2680. 2014.

# State-of-the-Art GANs




[1]

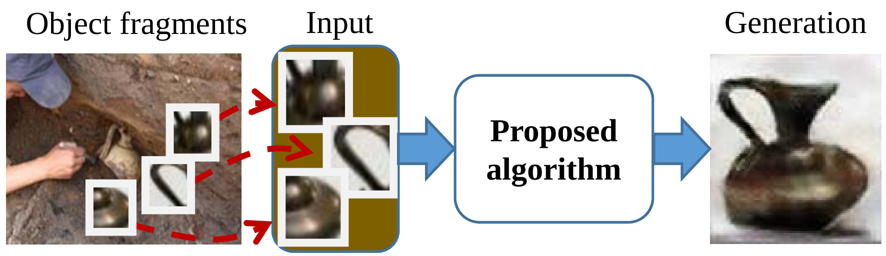
[2]

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros , “**Image-to-Image Translation with Conditional Adversarial Nets**,” CVPR 2017.
[2] Karras, T., Aila, T., Laine, S., & Lehtinen, J. **Progressive growing of GANs for improved quality, stability, and variation.** ICLR 2018.

# DEEP INSPIRATION

Donghoon Lee, Sangdoo Yun, Sungjoon Choi, Hwiyeon Yoo, Ming-Hsuan Yang, and Songhwai Oh, "**Unsupervised Holistic Image Generation from Key Local Patches**," in Proc. of the European Conference on Computer Vision (ECCV), Sep. 2018.

## Motivation



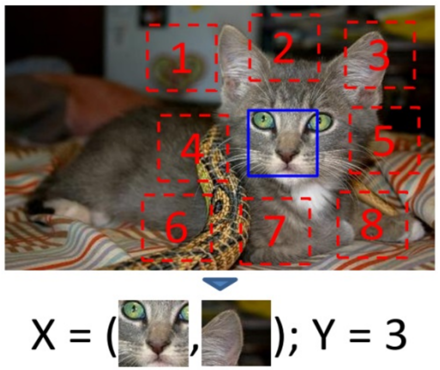

Can we generate an image conditioned on the appearances of local patches?

## Challenges

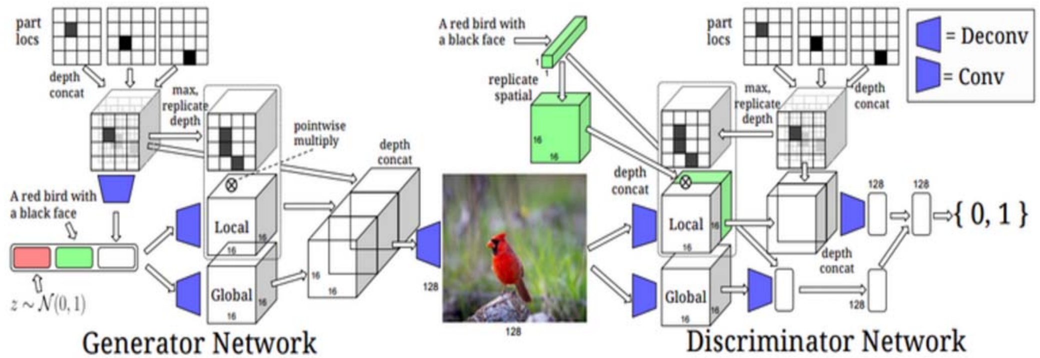
- ✓ A spatial relationship between input patches need to be inferred.
- ✓ Generated image should look like a real image.
- ✓ Generated image should contain input patches without significant modifications.

## Related Work

[1]




[2]

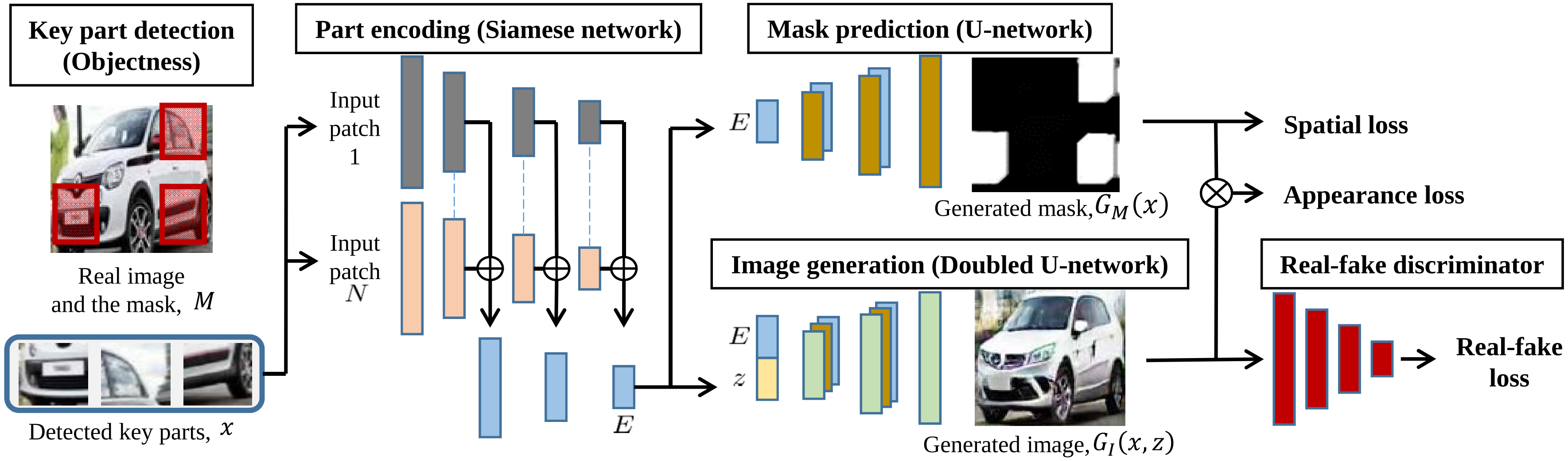



- \+ It predicts a few relative locations of two patches
- \- It cannot generate an image

- \+ It draws an image conditioned on an input text and part locations
- \- Part locations must be labelled in the training set

[1] Carl Doersch, Abhinav Gupta, and Alexei A. Efros, "Unsupervised visual representation learning by context prediction", ICCV 2015
[2] Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee, "Learning What and Where to Draw", NIPS 2016

## Our approach

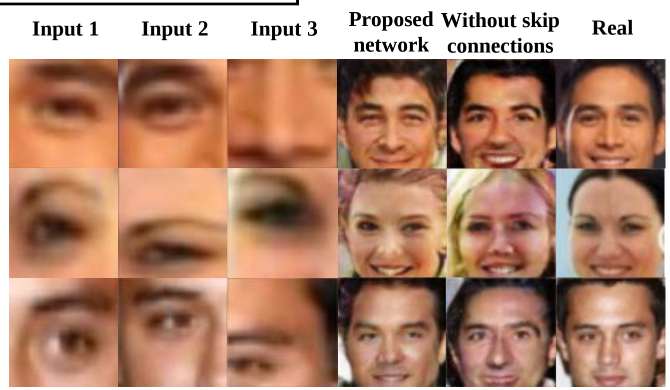



## Key local patches

They are informative local regions to generate an entire image

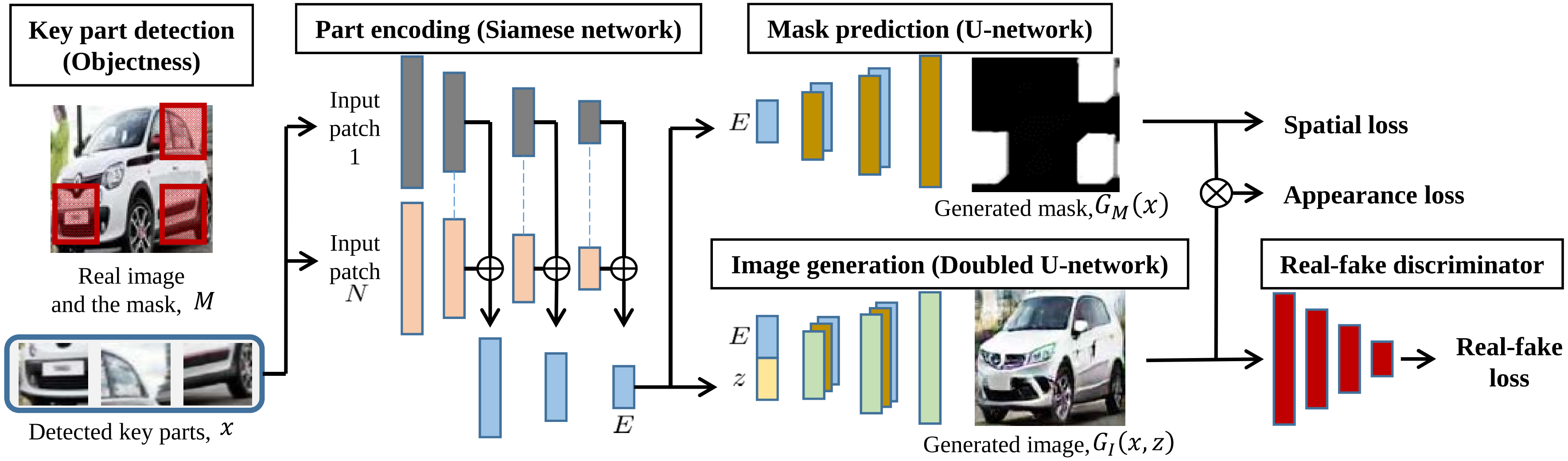
The EdgeBox is used to detect key patches (no labelling costs)

## Skip connections




Skip connections of U-net helps to improve the generation quality

## Our approach




## Loss function

$$\min_{G_M, G_I} \max_{D} L_R(G_I, D) + \lambda_1 L_S(G_M) + \lambda_2 L_A(G_M, G_I)$$

**Real-fake loss**

$$L_R(G_I, D) = E_{y \sim p_{data}(y)}[\log D(y)] + E_{x,y,y' \sim p_{data}(x,y,y'),\, M \sim p_{data}(M)} \Big[ \log\big(1 - D(G_I(x,z))\big) + \log\big(1 - D(M \otimes G_I(x,z) + (1-M) \otimes y)\big) + \log\Big(1 - D\big((1-M) \otimes G_I(x,z) + M \otimes y\big)\Big) + \log\Big(1 - D\big((1-M) \otimes y' + M \otimes y\big)\Big) \Big]$$

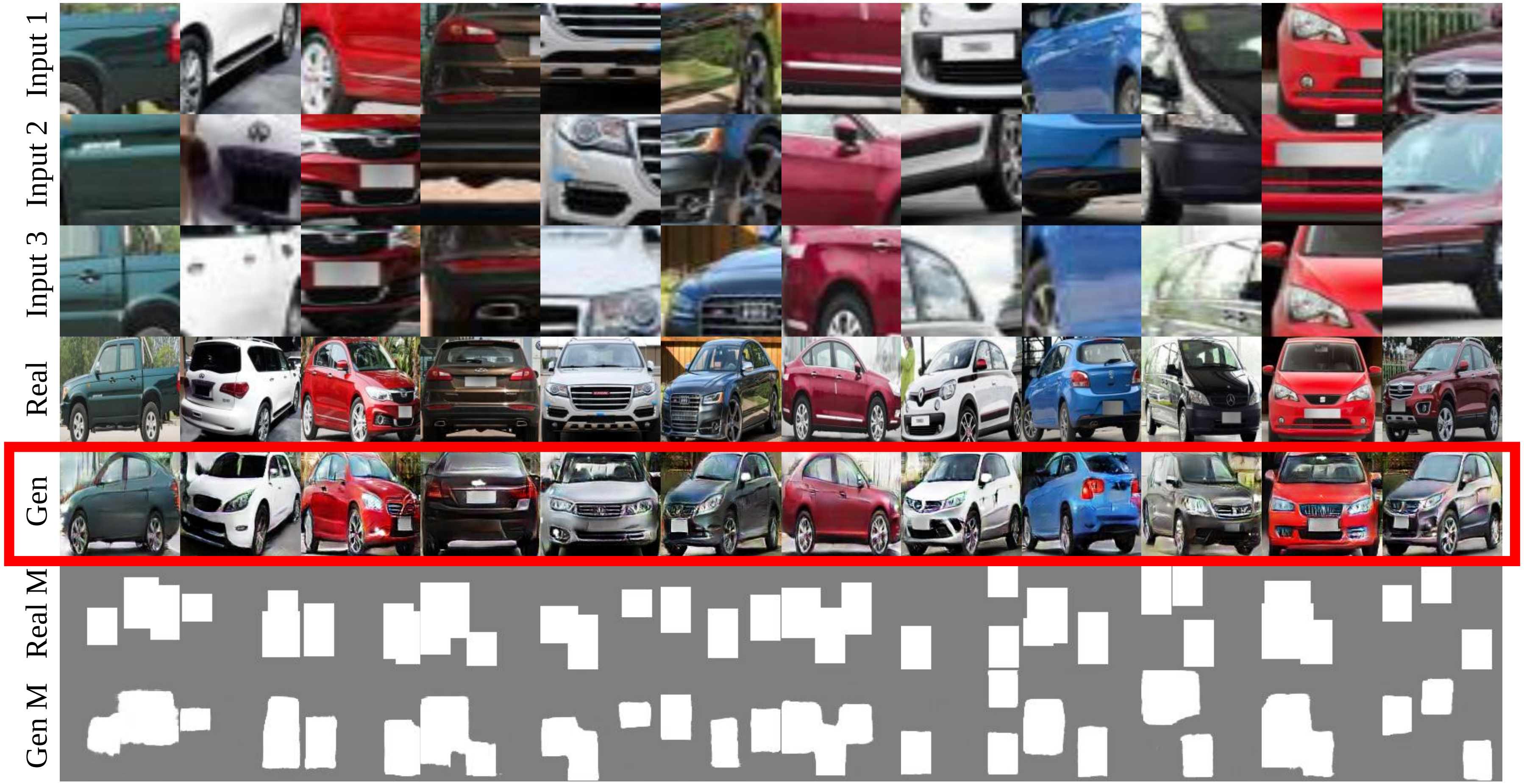
**Spatial loss**

$$L_S(G_M) = E_{x \sim p_{data}(x),\, M \sim p_{data}(M)} \left[ \left\| G_M(x) - M \right\|_1 \right]$$

**Appearance loss**

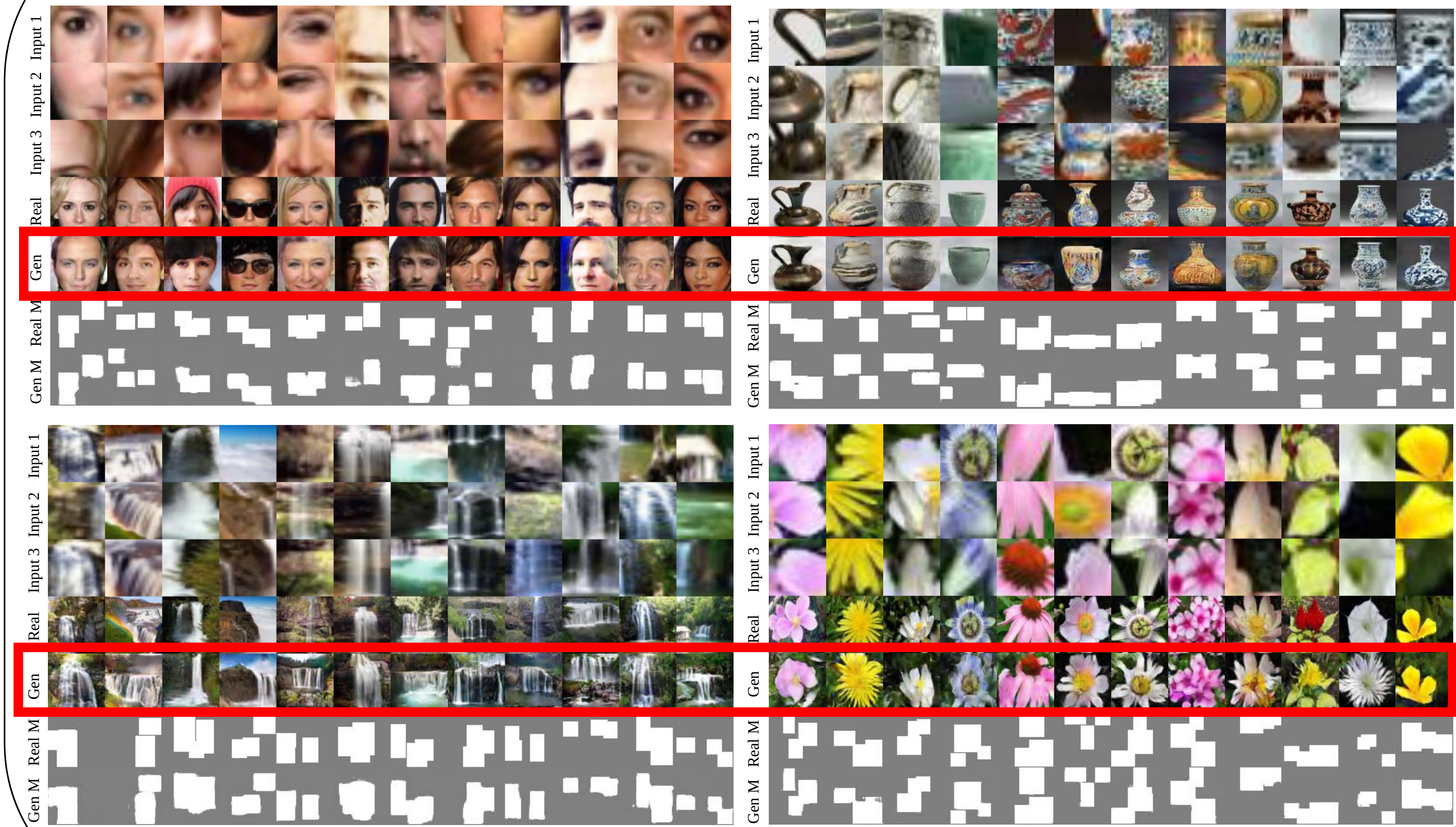
$$L_A(G_M, G_I) = E_{x,y \sim p_{data}(x,y),\, z \sim p_z(z),\, M \sim p_{data}(M)} \left[ \left\| G_I(x,z) \otimes G_M(x) - y \otimes M \right\|_1 \right]$$

# Image Generation Results (CompCars dataset, 128x128 pixels)



# Image Generation Results (CompCars dataset, 128x128 pixels)

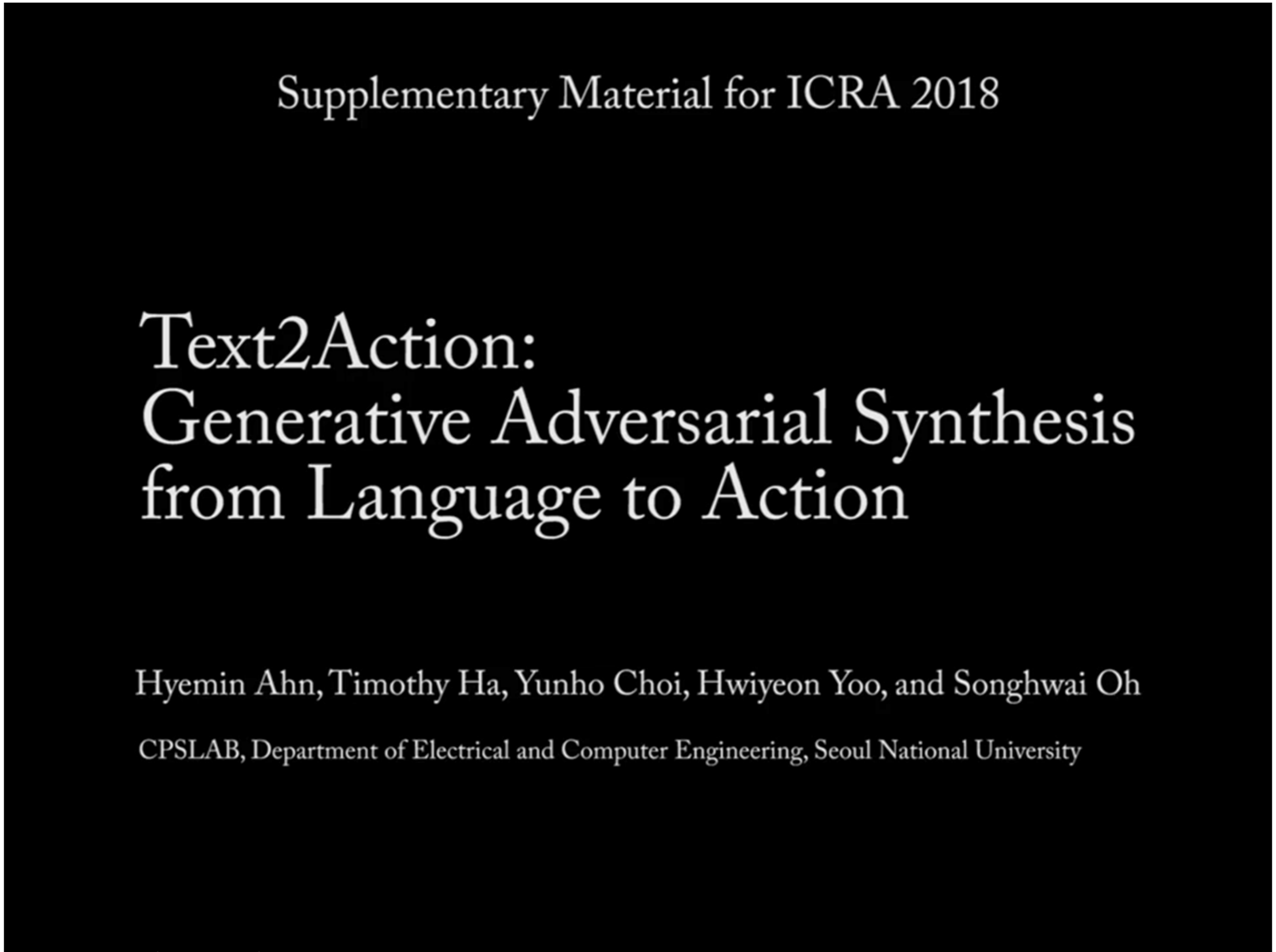
# More Results (CelebA, SNU pottery, SNU waterfall, Flowers102 datasets)



# TEXT2ACTION: MAKE A ROBOT ACT LIKE A HUMAN JUST BY TELLING

Hyemin Ahn, Timothy Ha, Yunho Choi, Hwiyeon Yoo, and Songhwai Oh, "**Text2Action: Generative Adversarial Synthesis from Language to Action**," in Proc. of the IEEE International Conference on Robotics and Automation (ICRA), May 2018.

# Text2Action: Experimental Results



# Text2Action Network

**Text2Action: Generative Adversarial Synthesis from Language to Action**

- A neural network generating the human behavior which is described by a given sentence.

⬇

**If we want to create this network, what specific tasks should we do?**

1. How to handle an input sentence (natural language)
   - What is a "Sentence"?
     - Sequence of characters / words
       - From the input sentence, how can we encode features related to the action?
2. How to generate the action from the processed language feature
   - What is an "Action"?
     - Sequence of poses in time.
       - In order to generate the each pose, how can we transfer the feature vector which is encoded from the input sentence?

**Word2Vec**
**RNN**

**Sequence to Sequence**

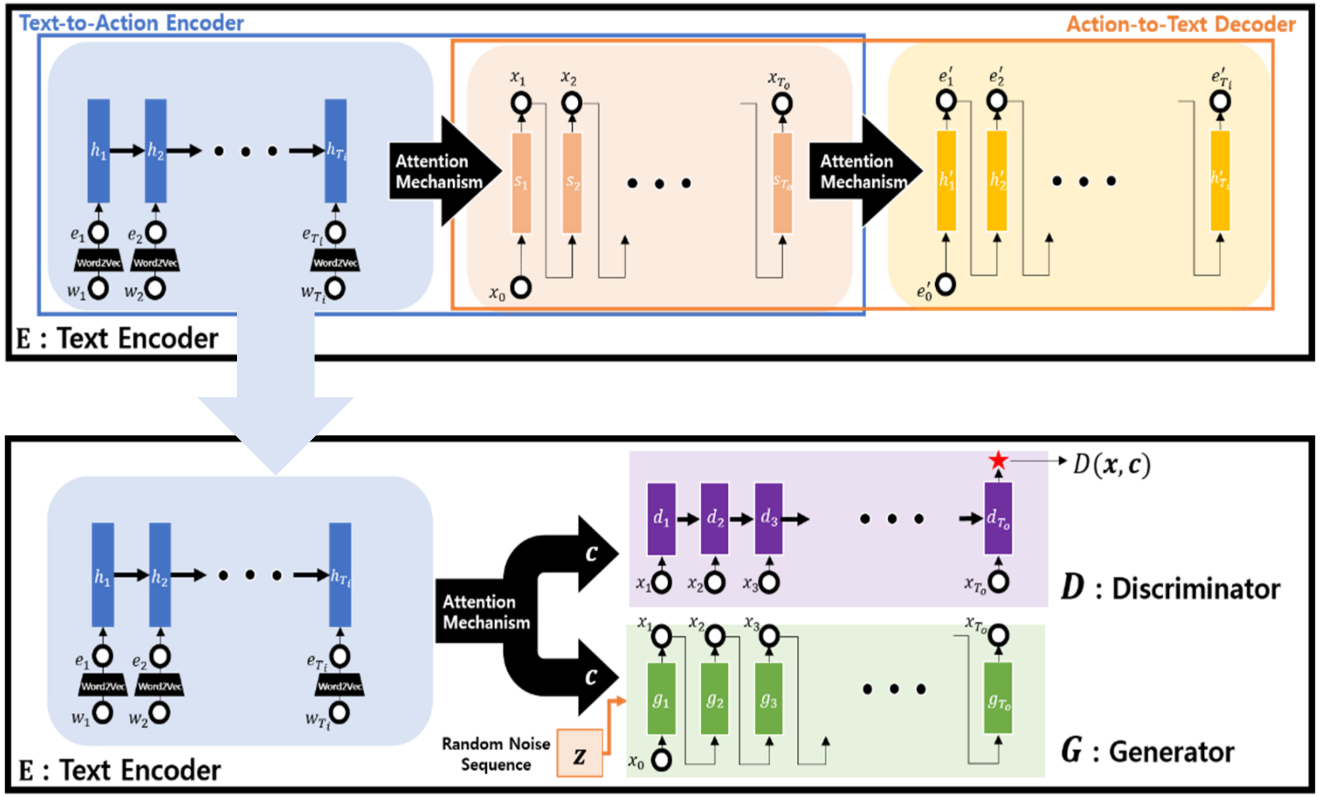
# Text2Action Network

**Text Encoder**

- 256 dimension for the LSTM hidden state vector
- Adam Optimizer with the learning rate $5 \times 10^{-5}$ is used for training

**Generator**

- 256 dimension for the LSTM hidden state vector




- Adam Optimizer with the learning rate $2 \times 10^{-6}$ is used for training

**Discriminator**

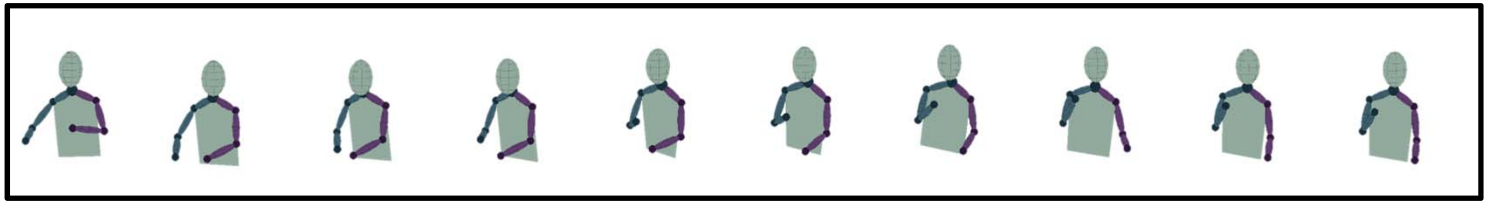
- 256 dimension for the LSTM hidden state vector
- Adam Optimizer with the learning rate $2 \times 10^{-6}$ is used for training

# Text2Action: Details

## Example

Input sentence : A chef is cooking a meal in the kitchen

Output action:



**Maximal sentence length** : 18 words except spaces

**Output pose dimension** : 24 dimension (8 joints, 3 dimension for each joint)

**Output action length** : 10 seconds

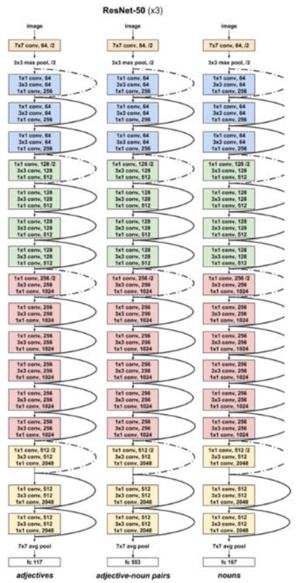
**Frame rate** : 10 frames per second

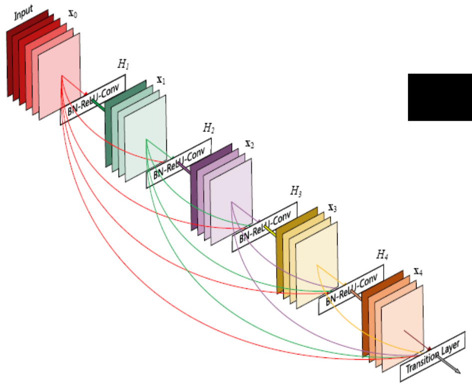
# NESTEDNET: NESTED SPARSE NETWORKS

Eunwoo Kim, Chanho Ahn, and Songhwai Oh, "**NestedNet: Learning Nested Sparse Structures in Deep Neural Networks**," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2018. **(Spotlight Presentation)**

# Related Work

## Going deeper and denser …

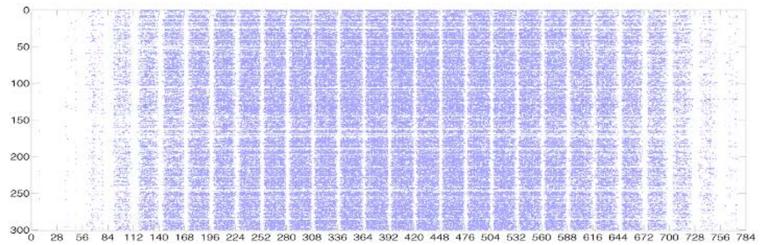


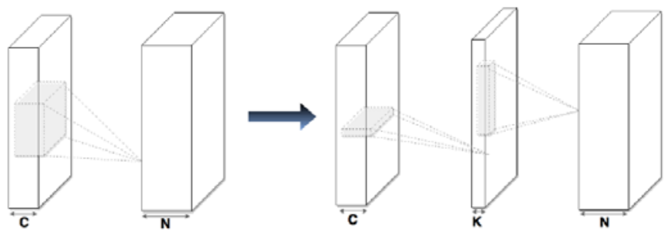

DenseNet
(CVPR'17 Best Paper)

ResNet (CVPR'16 Best Paper)

## Going to a compact network




sparse deep network (NIPS'15)



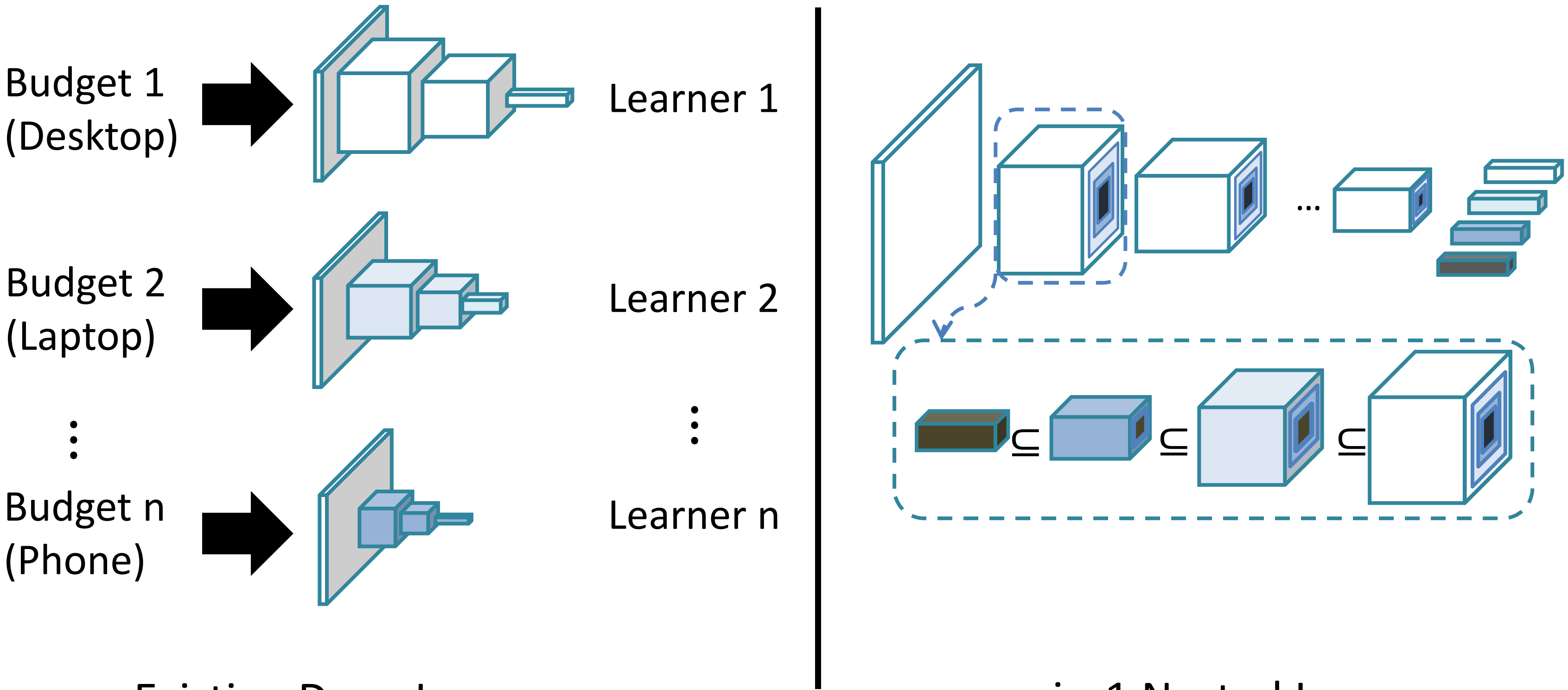

Low-rank deep network (ICLR'17)

Over-parameterized & highly redundancy
Difficult to achieve real-time inference

Approximating weight tensors
Save memory storage & faster inferencing

# Network-in-Network

## Different hardware computing capabilities



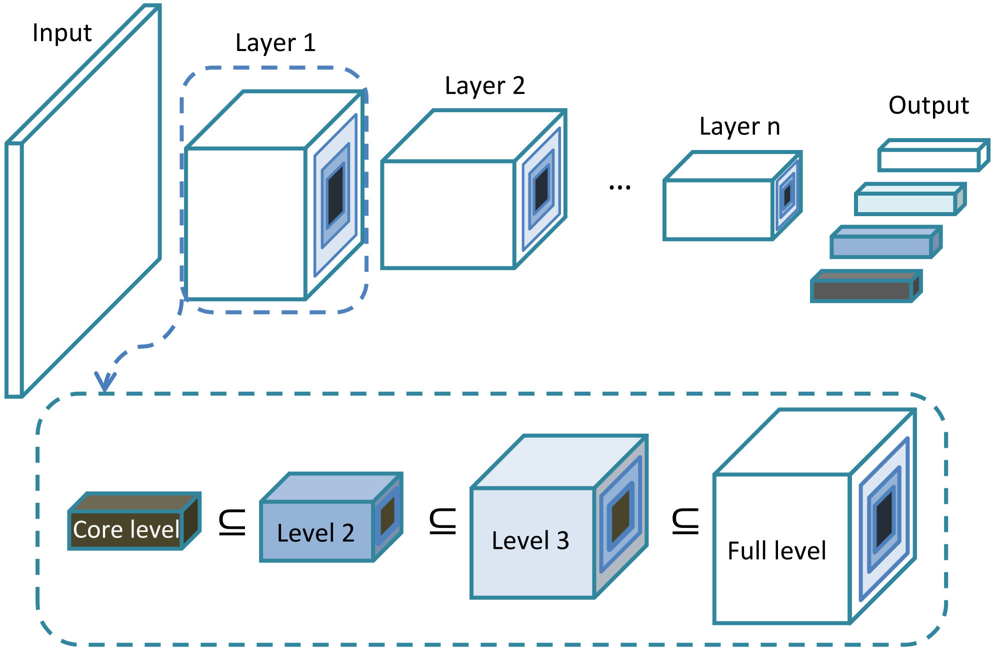

Existing Deep Learner

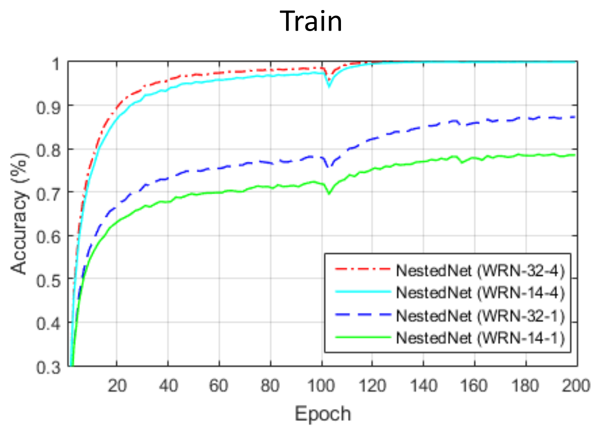
n-in-1 Nested Learner

**Why it is possible?** **Unnecessary redundancy** of existing deep networks.
We can fully utilize deep networks with a nested learner for multiple tasks.

# Training NestedNet

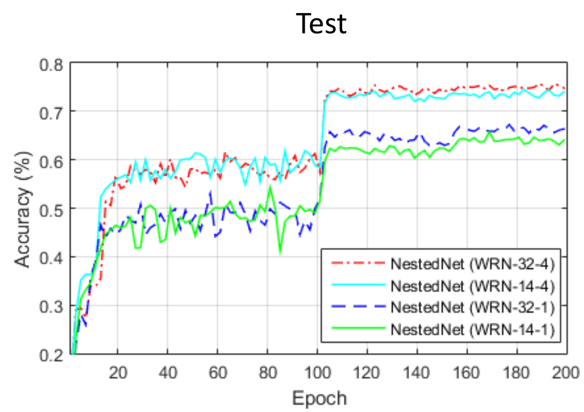
## **N-in-1 nested sparse network** (with anytime prediction)



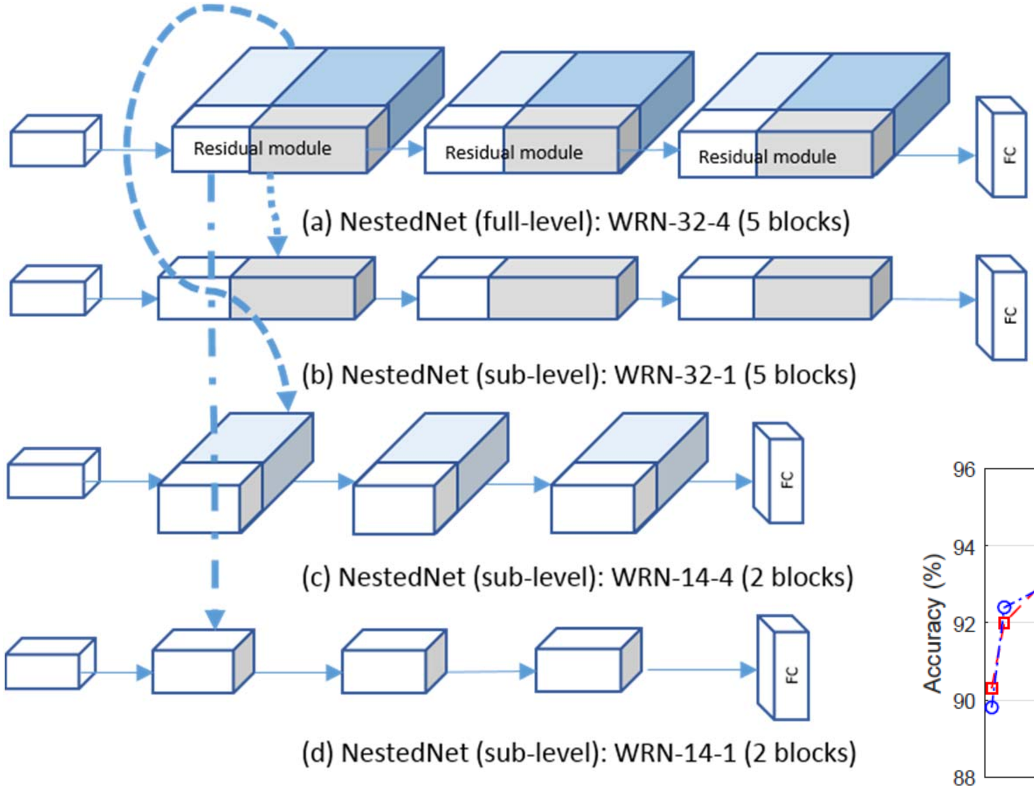

$$\min_{W} \frac{1}{l_n}\sum_{j=1}^{l_n} \mathcal{L}\left(Y, f\left(X, \mathcal{P}_{\Omega_{M^j}}(W)\right)\right) + \lambda \cdot \mathcal{R}\left(\mathcal{P}_{\Omega_{M^{l_n}}}(W)\right)$$

$$s.t.\ M^j \subseteq M^k, j \leq k, \forall j,k \in [1, \ldots, l_n]$$

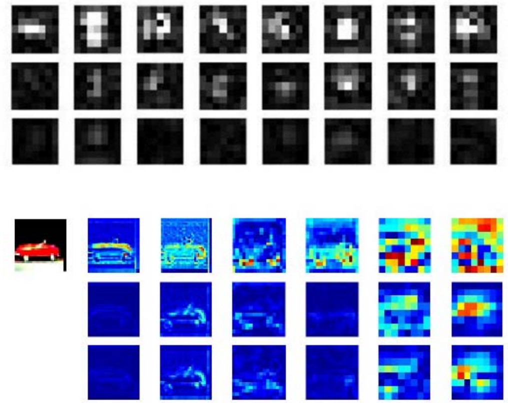


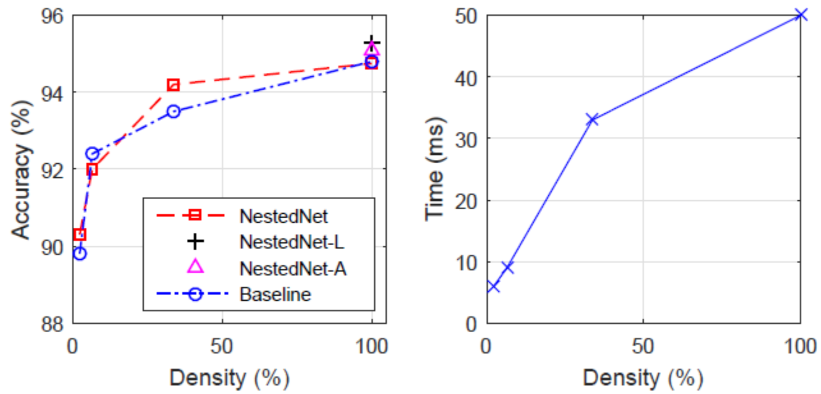
# Experiments: Knowledge Distillation

## Knowledge Distillation




Learned filters and activations

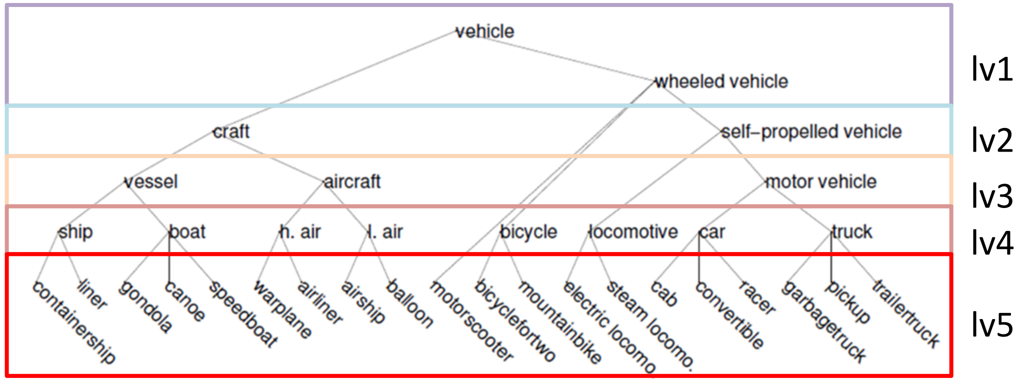


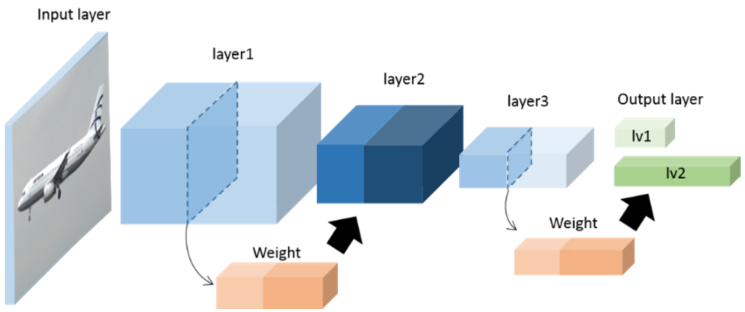



Results on CIFAR-10

# Experiments: Hierarchical Classification

## Hierarchical representation




Nested structure for class hierarchy

| Network | Architecture | $N_C$ | $N_P$ | Accuracy |
|---|---|---|---|---|
| Baseline | WRN-14-4 | 20 | 2.7M | 82.4% |
| | WRN-14-8 | 100 | 10.8M | 75.8% |
| NestedNet | WRN-14-4 | 20 | 2.7M | 83.9% |
| | WRN-14-8 | 100 | 10.8M* | 77.3% |
| NestedNet-A | WRN-14-8 | 100 | 10.9M* | 78.3% |
| NestedNet-L | WRN-14-8 | 100 | 10.9M* | 78.0% |
| SplitNet [15] | WRN-14-8 | 100 | 7.4M | 74.9% |
| Baseline | WRN-32-2 | 20 | 1.8M | 82.1% |
| | WRN-32-4 | 100 | 7.4M | 75.7% |
| NestedNet | WRN-32-2 | 20 | 1.8M | 83.7% |
| | WRN-32-4 | 100 | 7.4M* | 76.6% |
| NestedNet-A | WRN-32-4 | 100 | 7.4M* | 78.0% |
| NestedNet-L | WRN-32-4 | 100 | 7.4M* | 77.7% |

Classification accuracy on CIFAR-100

| Network | $N_C$ | $N_P$ | Accuracy |
|---|---|---|---|
| Baseline | 4 | 0.7M | 92.8% |
| | 11 | 2.8M | 89.2% |
| | 100 | 11.1M | 79.8% |
| NestedNet | 4 | 0.7M | 94.0% |
| | 11 | 2.8M | 90.2% |
| | 100 | 11.1M | 79.9% |
| NestedNet-A | 100 | 11.1M | 80.2% |
| NestedNet-L | 100 | 11.1M | 80.3% |

Classification accuracy on ImageNet

# DEEP REINFORCEMENT LEARNING

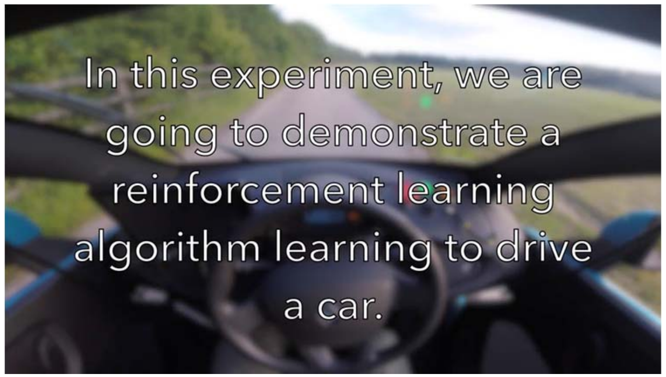
# Reinforcement Learning (RL)

- **Markov decision processes (MDPs)**
  - Complete model is known
- **Reinforcement learning (RL)**
  - Use observed rewards to learn an optimal policy for the environment
  - Complete model is not known
- **Methods**
  - Q-learning
  - Actor-critic
  - Policy gradient

Video Games, 2013


AlphaGo, 2016

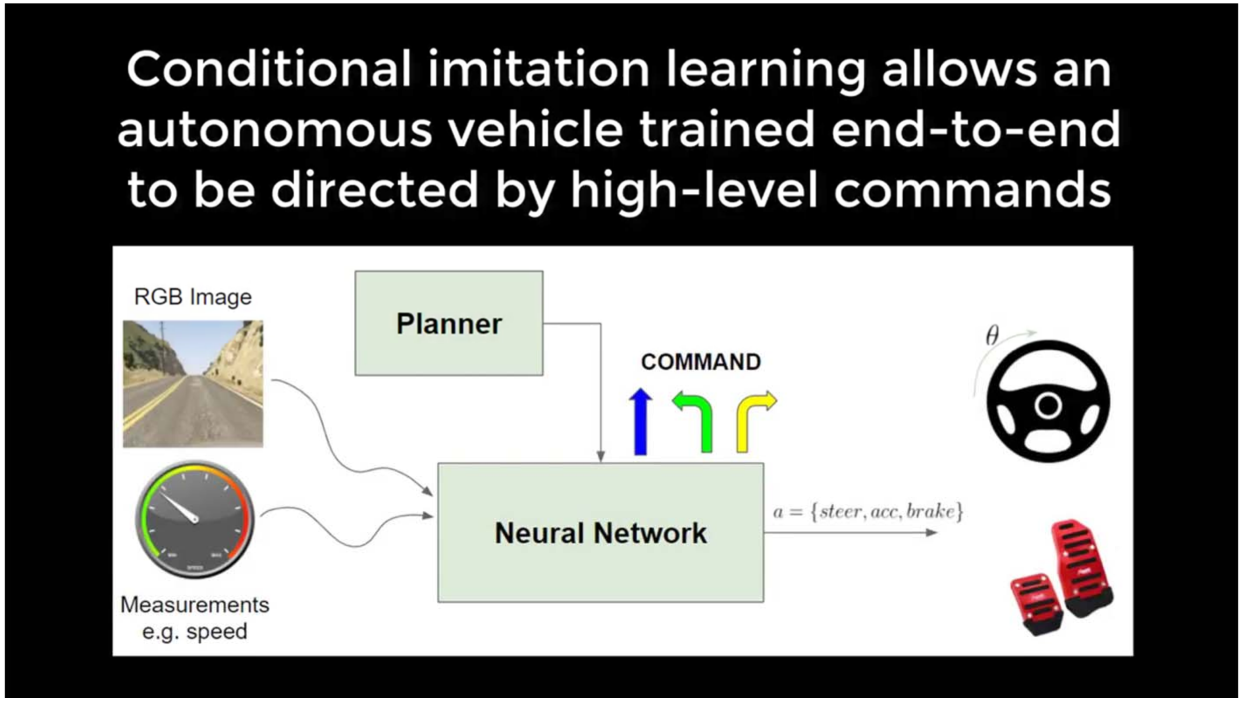

Autonomous Driving, 2018



# Inverse Reinforcement Learning (IRL)

- **Inverse reinforcement learning (IRL)**
  - For many complex problems, designing the reward function is extremely challenging.
  - Learn the reward function from expert's demonstrations
  - **Imitation learning** or **learning from demonstration**

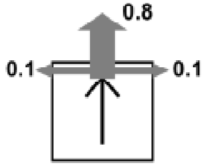
Conditional Imitation learning for driving, 2018 (Intel)



# Markov Decision Processes (MDP)

**Definition**

- A set of states $s \in S$
  - Each cell in the grid map
- A set of actions $a \in A$
  - Up, Down, Right, Left, Stay
- Transition probability
  - $P(s'|s,a)$



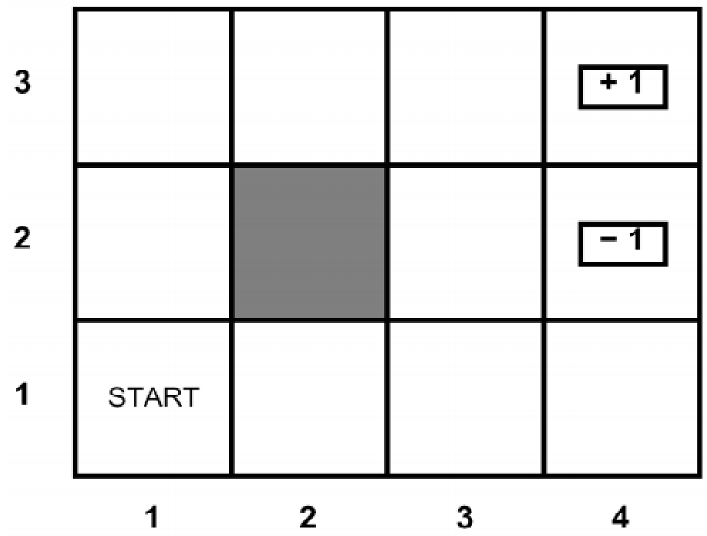

- Reward function $r(s,a)$
- Initial state distribution d
  - $P(s_0 = START) = 1$
- Terminal states (Optional)

**Markov Decision Process**

$S$: states
$A$: actions
$T$: transition model
$r$: reward function

**Goal**

$\pi$: policy ($S \rightarrow A$)



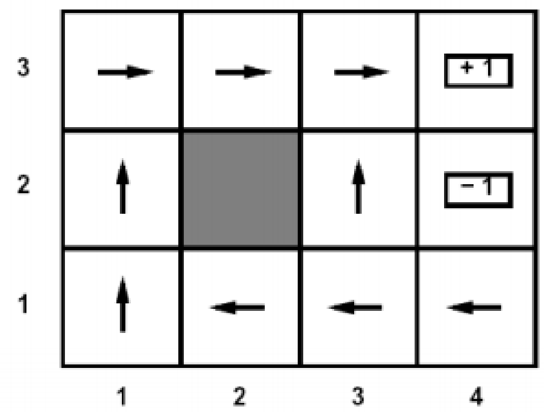

Source: AIMA

# Markov Decision Processes (MDP)

- **Goal of MDPs**: find an optimal policy $\pi^*: S \rightarrow A$
  - A policy $\pi$ gives an action $a$ for each state $s$
  - An optimal policy maximizes the expected sum of rewards
  - Markov decision problem

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

Optimal Policy




- Bellman Equation (Optimal condition of MDPs)
  - $Q(s,a) = r(s,a) + \gamma \sum_{s'} V(s') P(s'|s,a)$ ← **Q-function**
  - $V(s) = \max_{a} Q(s,a)$ ← **Value function**
  - $\pi(s) = \operatorname{argmax}_{a} Q(s,a)$ ← **Policy**

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "**Playing Atari with deep reinforcement learning**." NIPS Deep Learning Workshop 2013

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "**Human-level control through deep reinforcement learning**." *Nature* 518, no. 7540 (2015): 529.

# DEEP Q-NETWORK (DQN)

# Q-Learning with DNNs

Bellman equation (optimality condition):

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \Big| s,a\right]$$

-> value iteration

Q-network, $Q(s,a;\theta)$, approximates $Q^*(s,a)$ and is learned by minimizing:

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right]$$

$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a\right]$$

$\rho(s,a)$: probability distribution over $s$ and $a$.

Target value is **not** fixed (cf. supervised learning)

Stochastic gradient descent to learn $\theta$:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

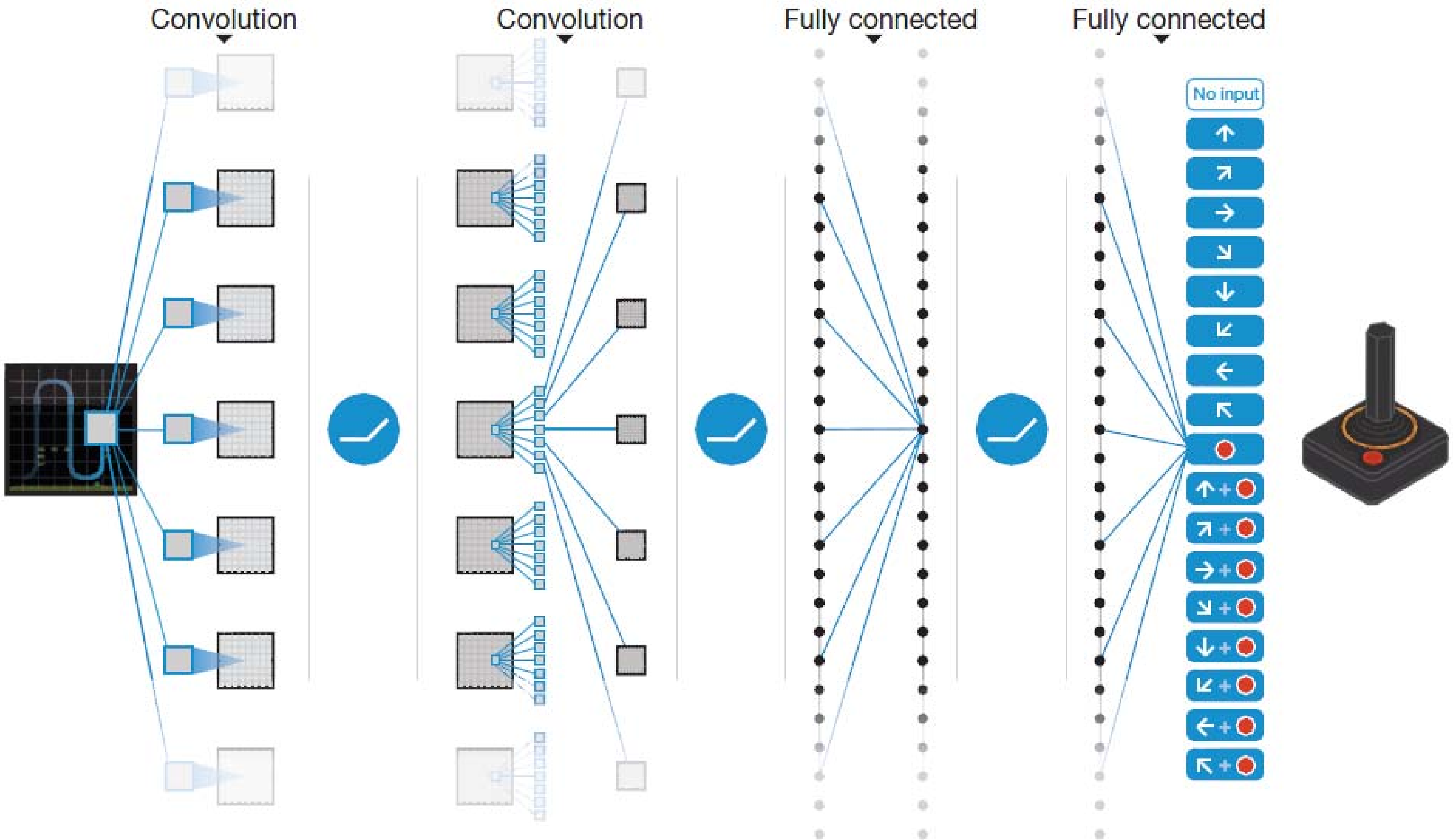
**Issues**: correlated data and non-stationary distribution -> **make Q-learning unstable**
**Solution**: experience replay

# Deep Q-Learning with Experience Replay

**Algorithm 1** Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
**for** episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$ ← Fixed length of sequence (last 4 frames)
  **for** $t = 1, T$ **do**
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ } $\epsilon$-greedy
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$ — minibatch = 32
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  **end for**
**end for**

# DQN




Input: 84 x 84 x 4 image, 3 convolutional layers, 2 fully connected layers

# Target Network (Final Version)

**Algorithm 1: deep Q-learning with experience replay.**
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
**For** episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  **For** $t = 1$,T **do**
    With probability $\varepsilon$ select a random action $a_t$
    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$ ← Target network
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$ ← Target network update
  **End For**
**End For**

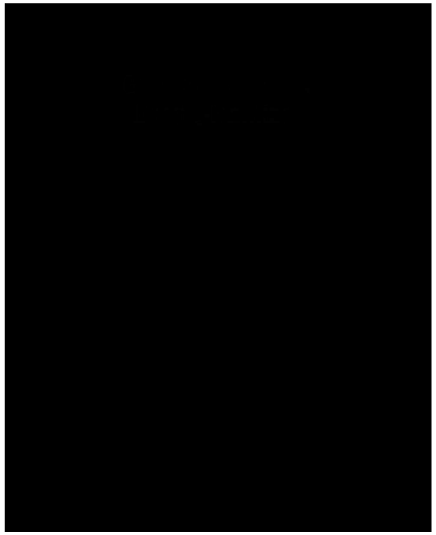
- Issues with Q-learning with function approximation: update that increases $Q(s_t, a_t)$ can also increase $Q(s_{t+1}, a_{t+1})$ for all $a_{t+1}$ and increase target $y_i$, leading to oscillation or divergence.
- A separate network to generate the target values, $y_i$.
- A target network makes an algorithm becomes more stable.

*clipping the error term between [-1,1]

# Training

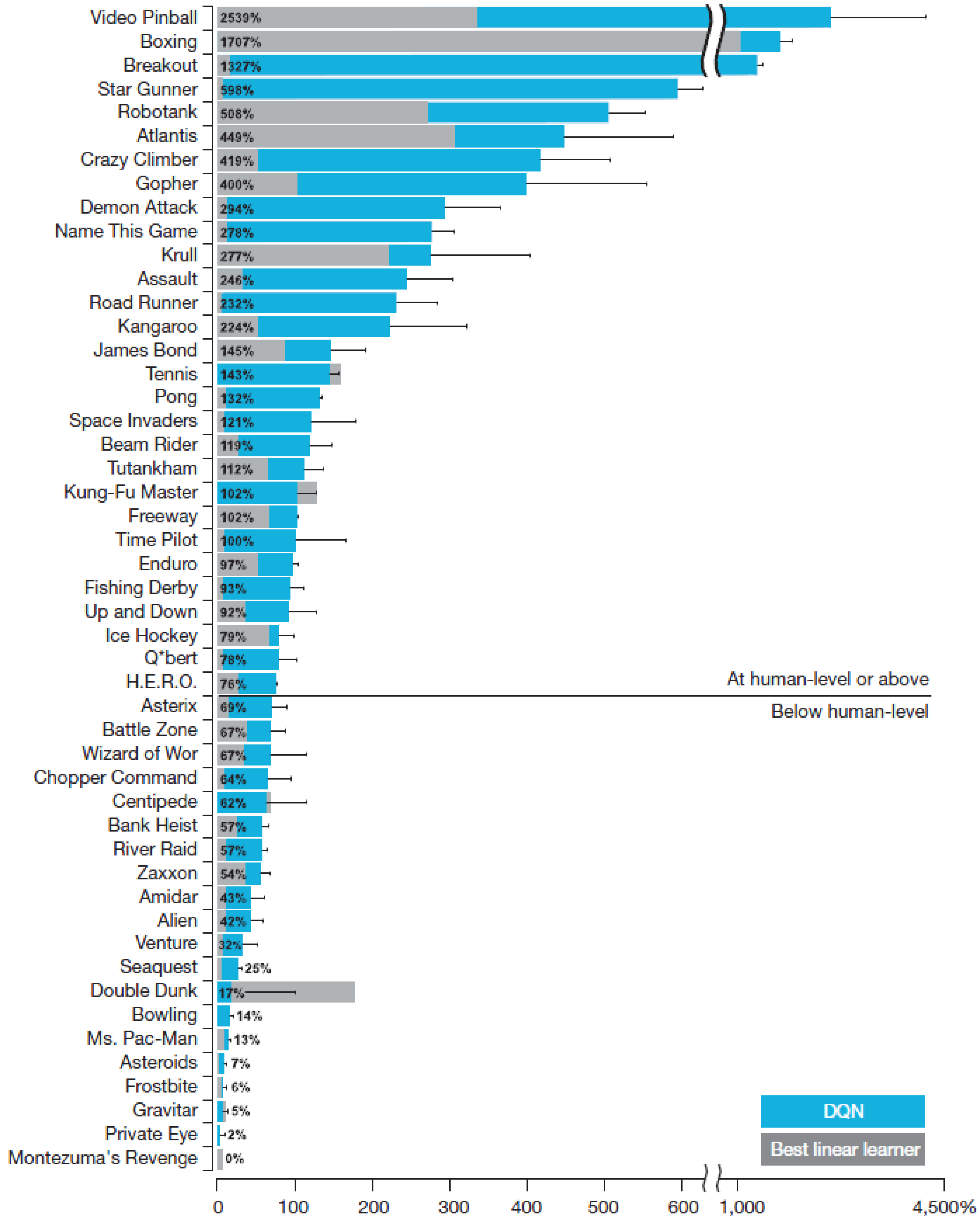
- Different networks for different games but the same architecture and hyperparameters
- RMSProp: divides the learning rate by a running average of recent gradient magnitudes
- Trained for 50 million frames (38 days of gaming)

# Results



Breakout

Space Invaders



# Key ingredients in DQN

- Experience replay
- Target network
- Error clipping

- and "a lot of engineering"

H. van Hasselt, A. Guez, and D. Silver, "**Deep reinforcement learning with double q-learning**," in Proc. of the AAAI Conference on Artificial Intelligence (AAAI), Feb, 2016.

# DOUBLE Q-LEARNING

# Double Q-Learning

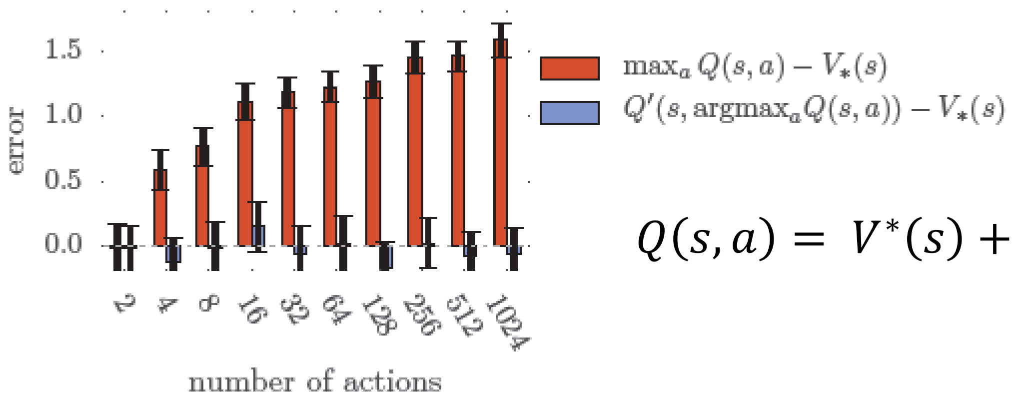
- DQN is likely to select overestimated values -> overoptimistic value estimates
- Decouple the selection from the evaluation

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t)$$

selection

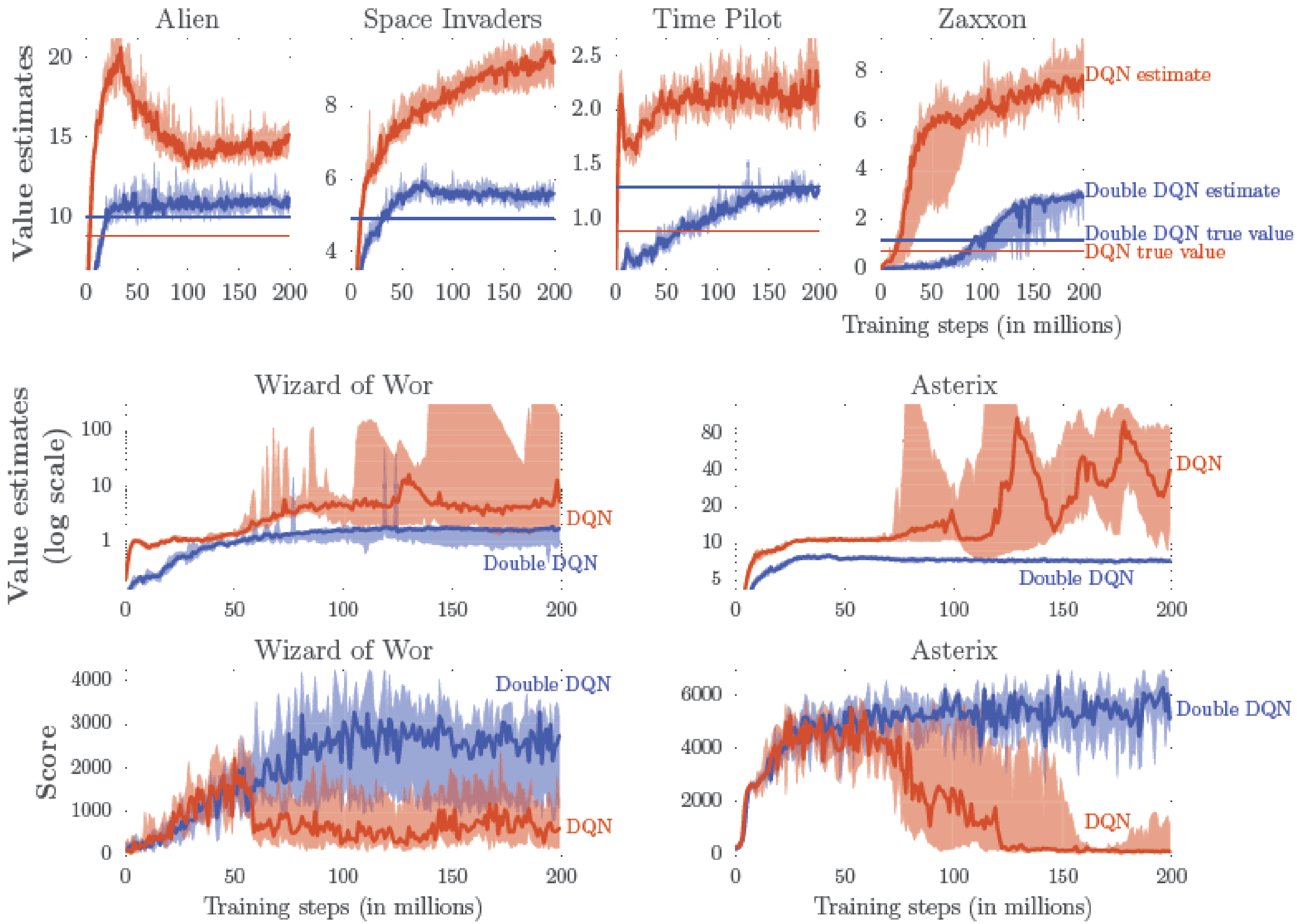
evaluation (uses the target network)

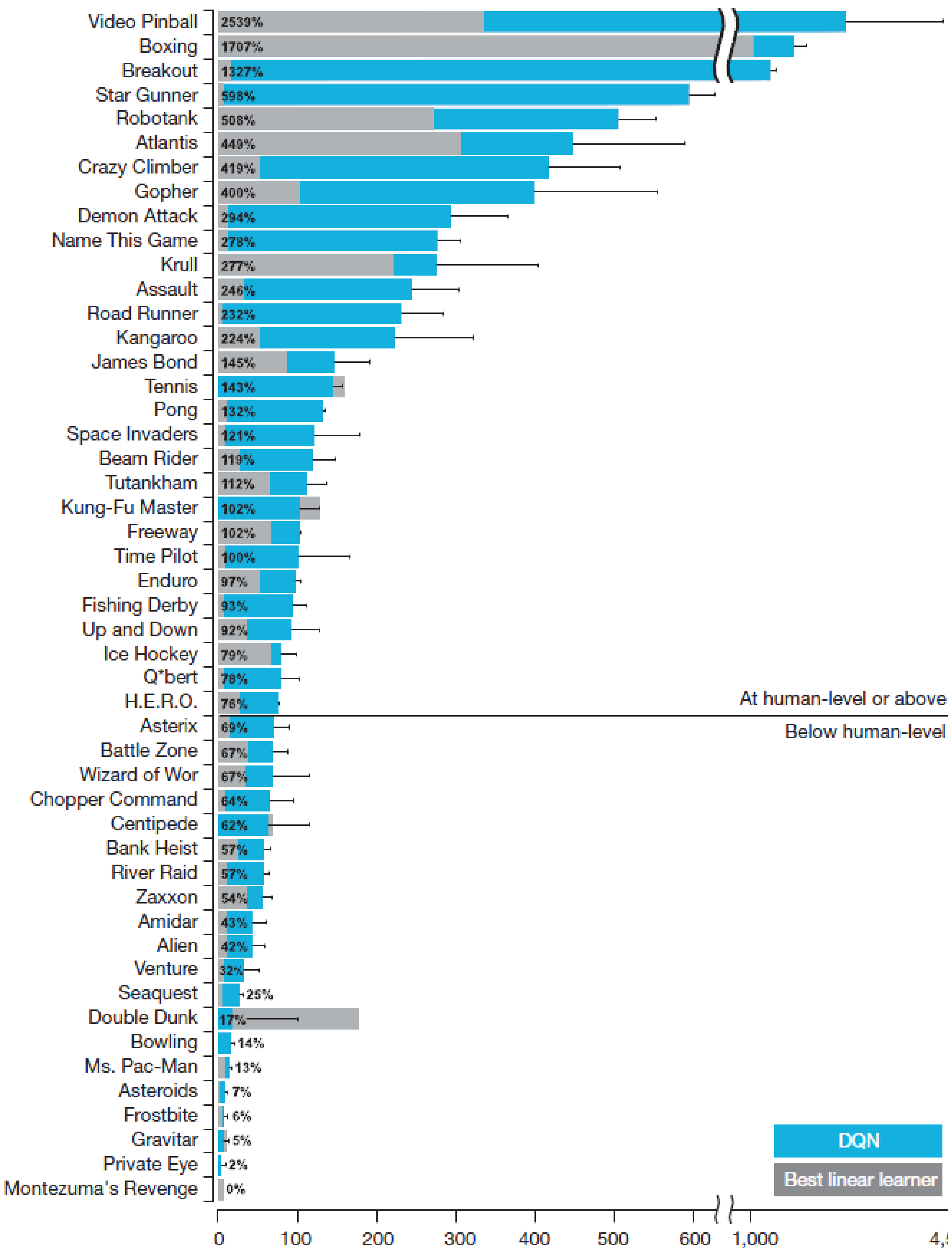
cf. DQN: $Y_t^{\text{Q}} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t)$

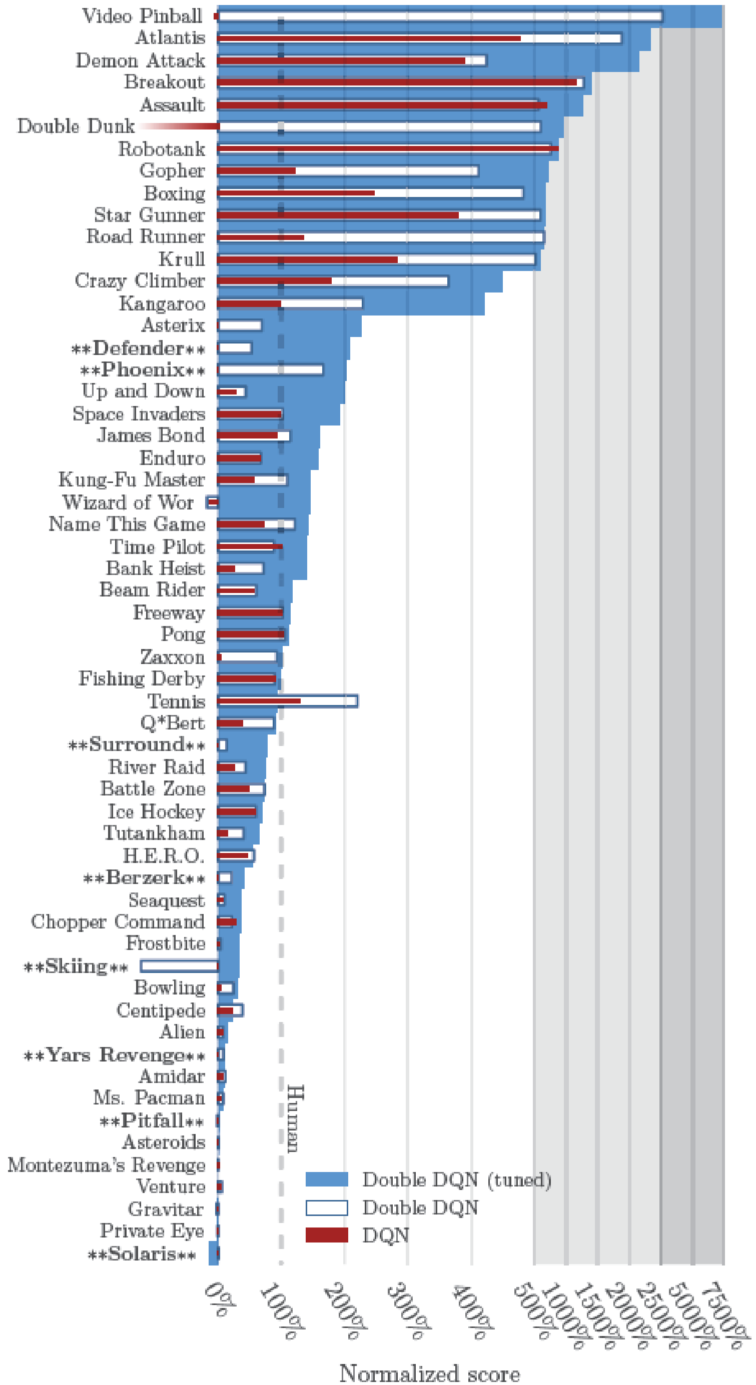



$$Q(s,a) = V^*(s) + \epsilon_a, \quad \epsilon_a \sim_{iid} \mathcal{N}(0,1)$$

# Comparison


Alien
Space Invaders
Time Pilot
Zaxxon
Value estimates
DQN estimate
Double DQN estimate
Double DQN true value
DQN true value
Training steps (in millions)
Wizard of Wor
Asterix
Value estimates (log scale)
DQN
Double DQN
Score
Training steps (in millions)


Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000


Video Pinball
Atlantis
Demon Attack
Breakout
Assault
Double Dunk
Robotank
Gopher
Boxing
Star Gunner
Road Runner
Krull
Crazy Climber
Kangaroo
Asterix
**Defender**
**Phoenix**
Up and Down
Space Invaders
James Bond
Enduro
Kung-Fu Master
Wizard of Wor
Name This Game
Time Pilot
Bank Heist
Beam Rider
Freeway
Pong
Zaxxon
Fishing Derby
Tennis
Q*Bert
**Surround**
River Raid
Battle Zone
Ice Hockey
Tutankham
H.E.R.O.
**Berzerk**
Seaquest
Chopper Command
Frostbite
**Skiing**
Bowling
Centipede
Alien
**Yars Revenge**
Amidar
Ms. Pacman
**Pitfall**
Asteroids
Montezuma's Revenge
Venture
Gravitar
Private Eye
**Solaris**
Human
Double DQN (tuned)
Double DQN
DQN
0%
100%
200%
300%
400%
500%
1000%
1500%
2000%
2500%
5000%
7500%
Normalized score

T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "**Prioritized Experience Replay**," ICLR, 2016

# PRIORITIZED EXPERIENCE REPLAY

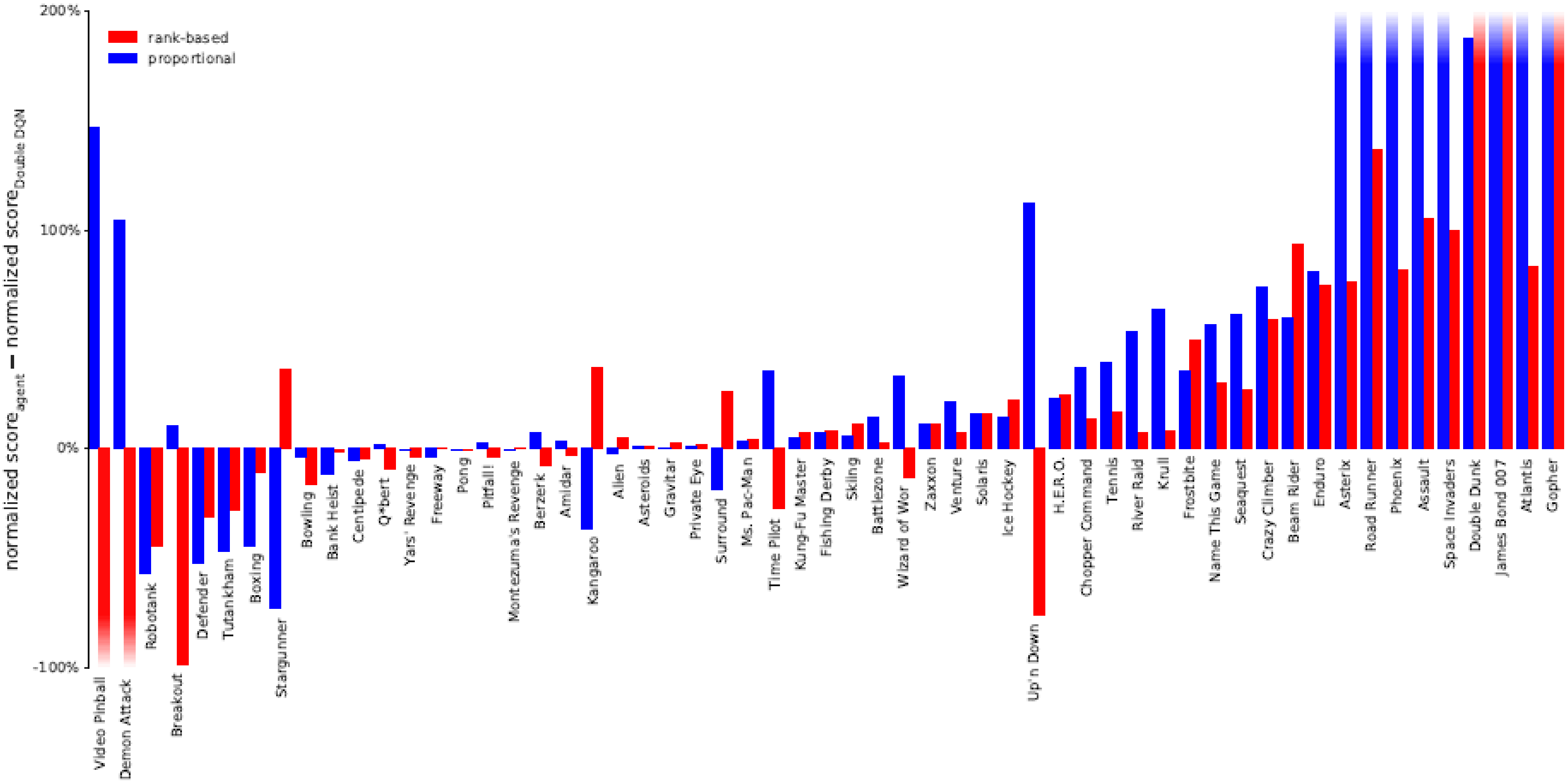
# Prioritized Experience Replay

**Algorithm 1** Double DQN with proportional prioritization

**Input:** minibatch $k$, step-size $\eta$, replay period $K$ and size $N$, exponents $\alpha$ and $\beta$, budget $T$.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe $S_0$ and choose $A_0 \sim \pi_\theta(S_0)$
**for** $t = 1$ **to** $T$ **do**
    Observe $S_t, R_t, \gamma_t$
    Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
    **if** $t \equiv 0 \mod K$ **then**
        **for** $j = 1$ **to** $k$ **do**
            Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
            Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
            Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
            Update transition priority $p_j \leftarrow |\delta_j|$
            Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
        **end for**
        Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
        From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
    **end if**
    Choose action $A_t \sim \pi_\theta(S_t)$
**end for**

Uniform if $\alpha = 0; \beta \uparrow 1$

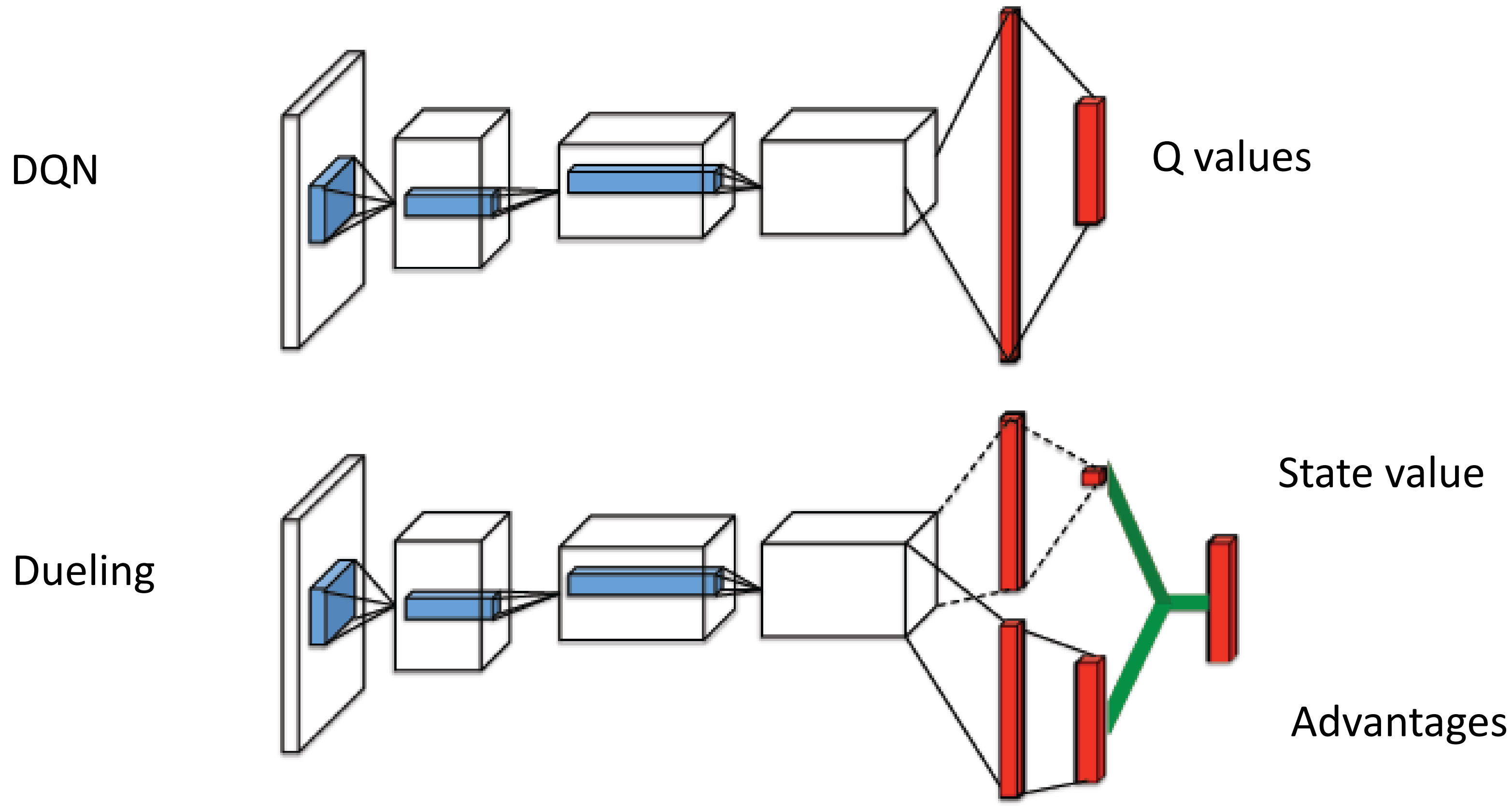
# Comparison with Double DQN



Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. "**Dueling network architectures for deep reinforcement learning**." in Proc. of the International Conference on Machine Learning (ICML), Jun, 2016.

# DUELING DQN

# Dueling Architecture




Advantage: $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$

**Dueling**: Updates state values more often -> Better estimates

# Forward Mapping for Q

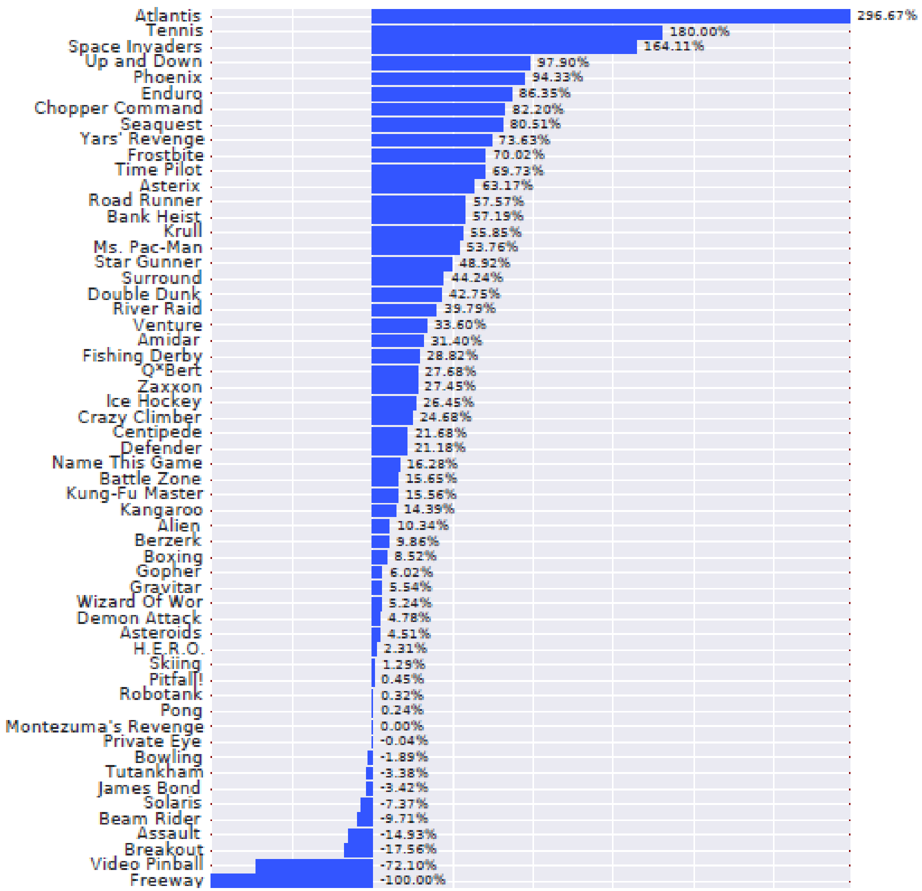
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a' \in |\mathcal{A}|} A(s,a';\theta,\alpha)\right)$$

- Makes a greedy action have zero advantage
- Addresses the identifiability issue

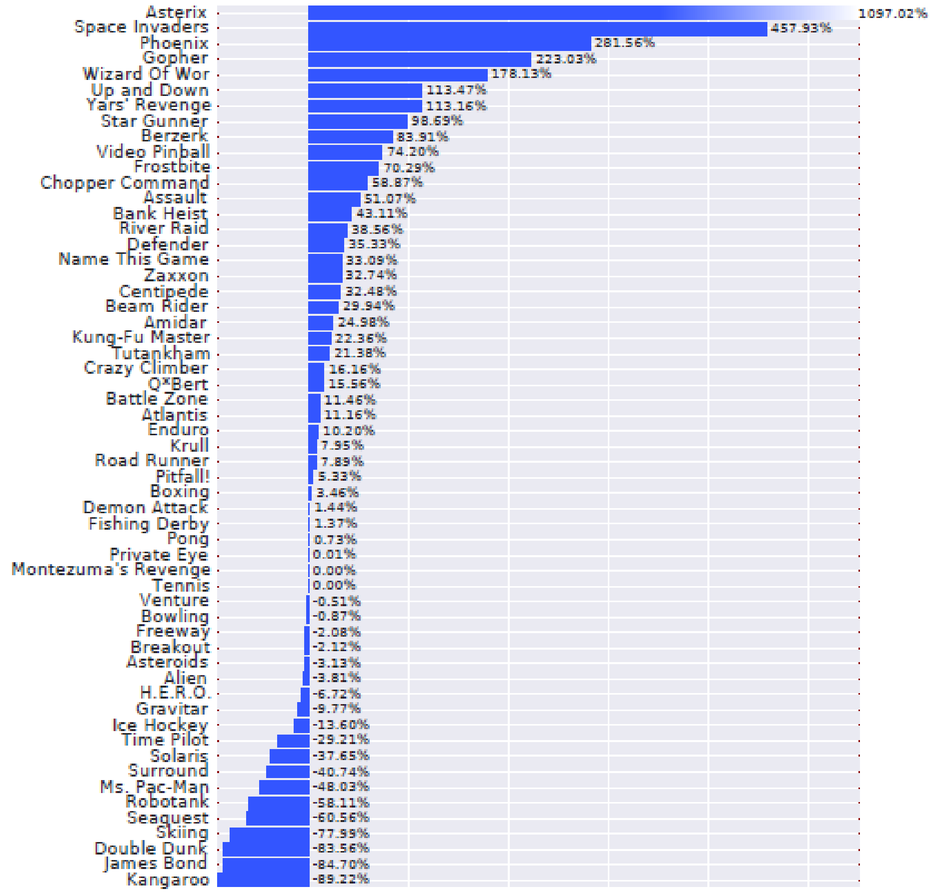
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\alpha)\right)$$

- Increases the stability of the algorithm
- Also addresses the identifiability issue

# Results




Dueling vs. Double DQN



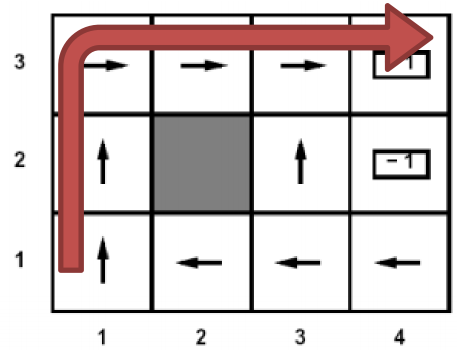

Dueling vs. Prioritized Double DQN

# SPARSE REINFORCEMENT LEARNING

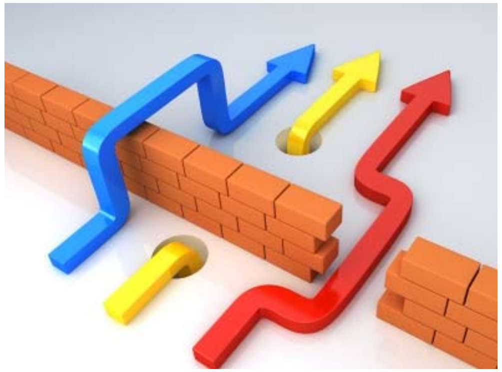

- Kyungjae Lee, Sungjoon Choi, Songhwai Oh, "**Maximum Causal Tsallis Entropy Imitation Learning**", in Proc. of Neural Information Processing Systems (NIPS), Dec. 2018.
- Kyungjae Lee, Sungjoon Choi, and Songhwai Oh, "**Sparse Markov Decision Processes with Causal Sparse Tsallis Entropy Regularization for Reinforcement Learning,**" IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1466-1473, Jul. 2018.

# Optimal Policy

- Optimal policy of an MDP: $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$
  - Deterministic function
  - An agent selects the exact same action at the same state
  - It can cause drawbacks in presence of multiple optimal actions




- Knowing multiple optimal action choices can be useful for real world problems
  - Go and chess
  - Driving a car and avoiding obstacles
  - Robustness against:
    – Unexpected or dynamic events
    – Modeling and estimation errors



# Stochastic Policies

– A stochastic policy can provide multiple action choices

– **Softmax** distribution: widely used stochastic policy function

– Probability is exponentially proportional to the state action value Q(s,a)

$$\pi(a|s) = \frac{\exp Q(s,a)}{\sum_{a'} \exp Q(s,a')}$$

– Softmax policy function:



Softmax Policy

# Soft MDPs

– The softmax distribution is the optimal solution of a soft MDP problem:

$$\max_{\pi} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] + \alpha H(\pi)$$

$$\text{subject to} \quad \forall\, s, a \quad \sum_{a} \pi(a|s) = 1, \qquad \pi(a|s) \geq 0$$

- Causal entropy regularization : $H(\pi) = \mathbb{E}[-\sum_{t=0}^{\infty} \gamma^t \log(\pi(a_t|s_t))]$
- It finds an optimal policy distribution $\pi(a|s)$ (not a deterministic function)
- $H(\pi)$ gives extra rewards to a multi-modal distribution

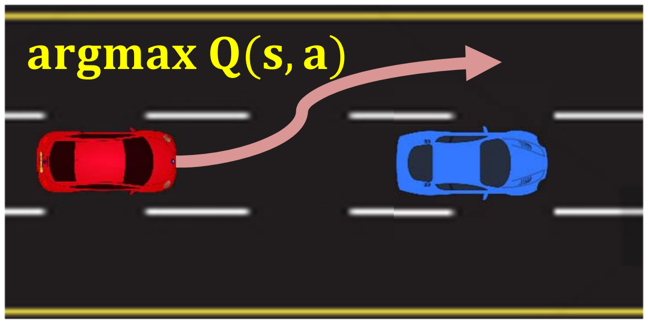


Softmax Policy

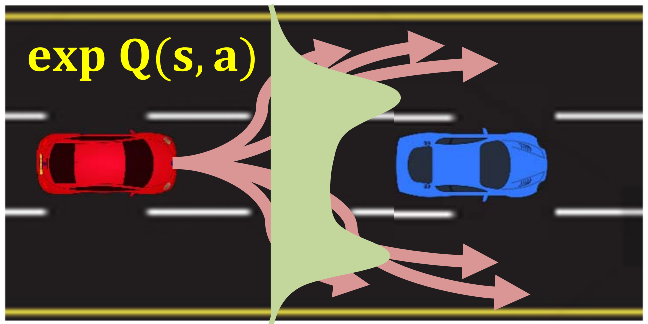
# Soft Bellman Equation

– Optimal condition of soft MDPs
  - $Q(s,a) = R(s,a) + \gamma \sum_{s'} V(s')P(s'|s,a)$
  - $V(s) = \alpha \log\left(\sum_{a'} \exp\left(\frac{Q(s,a')}{\alpha}\right)\right)$
  - $\pi(a|s) = \frac{\exp\left(\frac{Q(s,a)}{\alpha}\right)}{\sum_{a'} \exp\left(\frac{Q(s,a')}{\alpha}\right)}$

- Softmax policy can represent multiple optimal actions



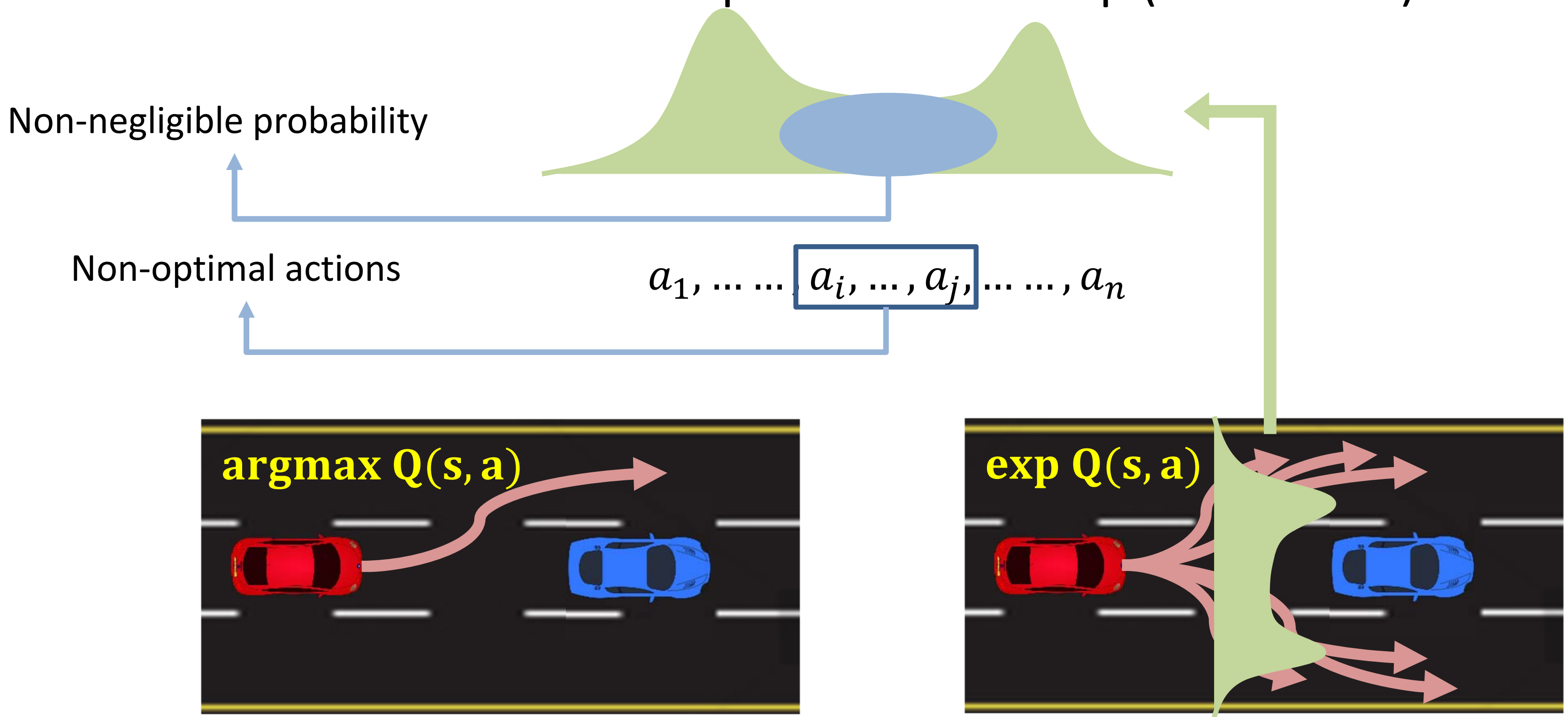
Original MDP (Deterministic)



Soft MDP (Stochastic)

# Drawback of Softmax Distribution

- Softmax policy assigns non-negligible probability mass to non-optimal actions even if state-action values of these actions are dismissible
- This behavior causes a performance drop (Theorem 5)

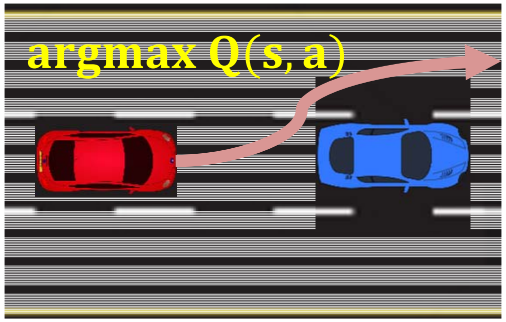



Original MDP (Deterministic)

Soft MDP (Stochastic)

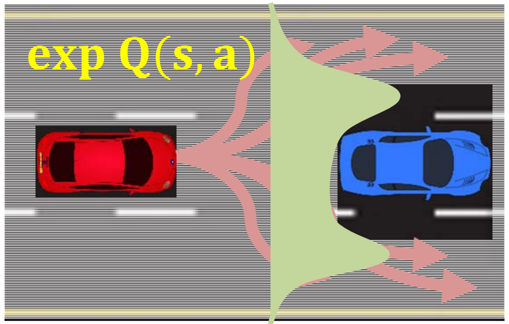
# Sparse Policy Distribution

- Drawback of softmax distribution: performance drop due to the non-optimal actions
- Sparse MDPs
    - Optimal policy is a sparse and multi-modal distribution



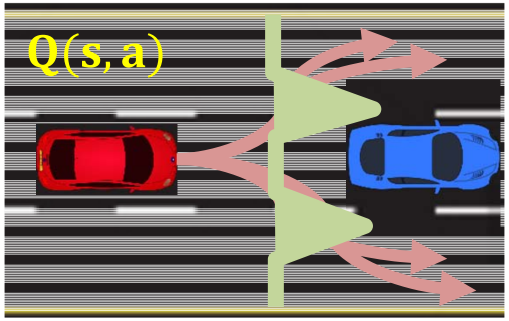

Original MDP (Deterministic)

$$\pi(s) = \text{argmax}\, Q(s, a)$$




Soft MDP (Stochastic)

$$\pi(a|s) = \frac{\exp Q(s, a)}{Z}$$




Sparse MDP (Stochastic)

$$\pi(a|s) = \max(Q(s, a) - \tau, 0)$$

# Sparse Markov Decision Processes (MDP)

CPSLAB
http://cpslab.snu.ac.kr

**Sparse Markov Decision Processes with Causal Sparse Tsallis Entropy Regularization for Reinforcement Learning**

Kyungjae Lee, Sungjoon Choi, and Songhwai Oh
CPSLAB, ECE
Seoul National University

# Sparse MDP Problem

- Sparse MDP Problem:

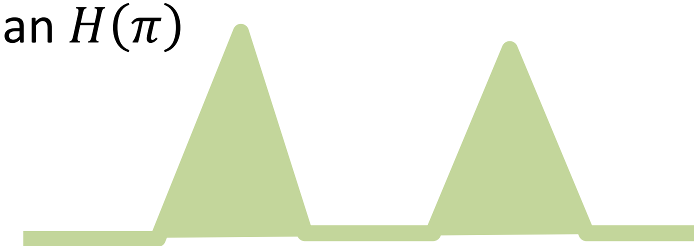
$$\max_{\pi} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] + \alpha W(\pi)$$

$$\text{subject to} \quad \forall\, s, a \quad \sum_{a} \pi(a|s) = 1, \qquad \pi(a|s) \geq 0$$

  - Causal Sparse Tsallis Entropy Regularization:

$$W(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \frac{\gamma^t}{2}(1 - \pi(a_t|s_t))\right]$$

  - $W(\pi)$ gives extra rewards to multi-modal policy distribution, but weaker than $H(\pi)$



Sparsemax Policy

# Tsallis Entropy

– Tsallis entropy:
  - $S_{q,k}(p) = \frac{k}{q-1}\left(1 - \sum p_i^q\right)$
  - Generalization of the standard Boltzmann–Gibbs entropy
  - Since 2000, Tsallis entropy is widely used in the field of physics, information theory, and social science
  - Tsallis entropy (nonadditive) has been used to describe complex phenomena that cannot be explained by the Boltzmann–Gibbs entropy (additive)

– Tsallis entropy has been successfully applied to explain:
  - The fluctuation of the magnetic field in the solar wind
  - The velocity distributions in dissipative dusty plasma
  - Thermostatistics of overdamped motion of interacting particles
  - Heavy tail distributions are derived from a maximum Tsallis entropy problem

Constantino Tsallis, "**Possible Generalization of Boltzmann-Gibbs statistics**," *Journal of statistical physics* 52.1 (1988): 479-487.

# Sparse Tsallis Entropy Regularization

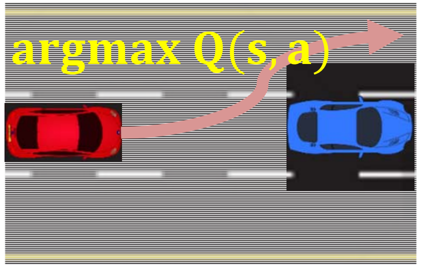
- Tsallis entropy: $S_{q,k}(p) = \frac{k}{q-1}\left(1 - \sum p_i^q\right)$
  - $q$ is called entropic-index
  - $k$ is a positive real constant
- Special cases:
  - Boltzmann–Gibbs entropy: $S_{1,1}(p) = \lim_{q \to 1} \frac{\sum\left(1-p_i^{q-1}\right)p_i}{q-1} = \sum -p_i \log p_i$
  - Sparse Tsallis entropy: $S_{2,\frac{1}{2}}(p) = \frac{1}{2}\sum p_i(1-p_i)$
- Causal Sparse Tsallis Entropy Regularization:

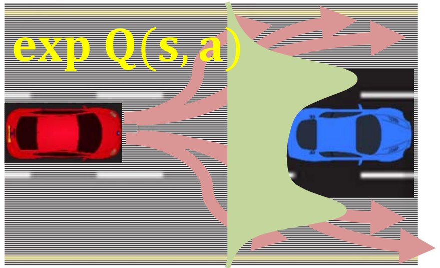
$$W(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \frac{\gamma^t}{2}(1 - \pi(a_t|s_t))\right]$$

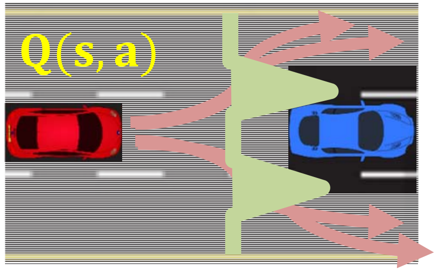
  - $W(\pi)$ is an extension of sparse Tsallis entropy to the causally conditioned random variables

**Theorem 2.** $\mathbb{E}\left[\sum_{t=0}^{\infty} \frac{\gamma^t}{2}(1 - \pi(a_t|s_t))\right] = \sum_s \rho_\pi(s) \frac{1}{2} \sum_a \pi(a|s)(1 - \pi(a|s))$

# MDP vs. Soft MDP vs. Sparse MDP




| Problem | Original MDP (Deterministic) | Soft MDP (Stochastic) | Sparse MDP (Stochastic) |
|---|---|---|---|
| Objective function | $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$ | $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right] + \alpha H(\pi)$ | $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right] + \alpha W(\pi)$ |
| $Q(s,a)$ | $r(s,a) + \gamma \sum_{s'} V(s')P(s'\|s,a)$ | | |
| $V(s)$ | $\max_a Q(s,a)$ | $\alpha \log\left(\sum_{a'} \exp\left(\frac{Q(s,a')}{\alpha}\right)\right)$ | $\alpha\left(\frac{1}{2}\sum_{a' \in S(s)}\left(\frac{Q(s,a')}{\alpha}\right)^2 - \tau^2 + \frac{1}{2}\right)$ |
| $\pi(a\|s)$ | $\operatorname{argmax}_a Q(s,a)$ | $\frac{1}{Z}\exp\left(\frac{Q(s,a)}{\alpha}\right)$ | $\max\left(\frac{Q(s,a)}{\alpha} - \tau, 0\right)$ |

# Value Iterations: MDP, Soft MDP, Sparse MDP

| Algorithm | Value Iteration | Soft Value Iteration | Sparse Value Iteration |
|---|---|---|---|
| Bellman Operation | $U(x) = \max q(s,\cdot)$ | $U(x) = \alpha \log\left(\sum_{a'} \exp\left(\frac{q(s,\cdot)}{\alpha}\right)\right)$ | $U(x) = \alpha \text{ spmax}\left(\frac{q(s,\cdot)}{\alpha}\right)$ |
| | $q(s,a) = r(s,a) + \sum_{s'} x(s')T(s'\|s,a)$ | | |
| Method | $v_{i+1} = U(v_i)$ | | |

**Sparsemax Operation**

$$V(\text{s}) = \alpha \text{ spmax}\left(\frac{Q(s,\cdot)}{\alpha}\right) := \alpha\left(\frac{1}{2}\sum_{a' \in S(s)}\left(\frac{Q(s,a')}{\alpha}\right)^2 - \tau\left(\frac{Q(s,\cdot)}{\alpha}\right)^2 + \frac{1}{2}\right)$$

**Theorem**. Following inequalities hold:

$$\mathbb{E}_{\pi^*}[r(s,a)] - \frac{\alpha}{1-\gamma}\frac{|\mathcal{A}|-1}{2|\mathcal{A}|} \leq \mathbb{E}_{\pi^{sp}}[r(s,a)] \leq \mathbb{E}_{\pi^*}[r(s,a)]$$

$$\mathbb{E}_{\pi^*}[r(s,a)] - \frac{\alpha}{1-\gamma}\log(|\mathcal{A}|) \leq \mathbb{E}_{\pi^{soft}}[r(s,a)] \leq \mathbb{E}_{\pi^*}[r(s,a)]$$

where $\pi^*$, $\pi^{sp}$, and $\pi^{soft}$ are the optimal policy obtained by the original MDP, soft MDP, and sparse MDP, respectively, and $|\mathcal{A}|$ is the number of action.

# Wrap Up

- Generative adversarial networks
- NestedNet
- Deep reinforcement learning
- Sparse RL