# Lecture 8: Deep Reinforcement Learning

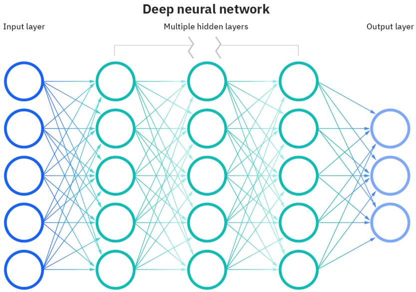
CS60077 : REINFORCEMENT LEARNING

Autumn 2023

# Outline

# Deep Neural Networks as Function Approximators

- Use Deep Neural Networks (DNN) to represent
  - State Value (**V**) Function
  - State-Action Value (**Q**) Function
  - Parameterized Policy
  - Model
- Optimize Loss Function by Stochastic Gradient Descent



# Q-Learning with Value Function Approximation

- Q-learning under Tabular Representation:
  - Converges to the optimal $Q^*(s, a)$
- Q-learning with Value Function Approximation (VFA):
  - Minimize MSE loss by stochastic gradient descent using a target Q estimate instead of true Q value
  - Q-learning with VFA can *diverge*
- Two of the issues causing problems:
  - Correlations between Samples
  - Non-Stationary Targets
- Deep Q-learning (DQN) addresses these challenges by
  - Experience Replay (removing correlations)
  - Fixed Q-Targets (improving stability)

# DQNs with Experience Replay

- To help remove correlations, store dataset $\mathcal{D}$ (called a **replay buffer**) from prior experience

| |
|---|
| $s_1, a_1, r_2, s_2$ |
| $s_2, a_2, r_3, s_3$ |
| $s_3, a_3, r_4, s_4$ |
| $\ldots$ |
| $s_t, a_t, r_{t+1}, s_{t+1}$ |

$\rightarrow \quad s, a, r, s'$

- To perform experience replay, repeat the following:
  - $(s, a, r, s') \sim \mathcal{D}$: sample an experience tuple from the dataset
  - Compute the target value for the sampled

$$s: r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'; \mathbf{w})$$

  - Use stochastic gradient descent to update the network weights

$$\Delta \mathbf{w} = \alpha \left( r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

- **Uses target as a scalar, but function weights will get updated on the next round, changing the target value**

# DQNs with Fixed Q-Targets

- To help improve stability, fix the **target weights** used in the target calculation for multiple updates
- Target network uses a different set of weights than the weights being updated
- Let parameters $\mathbf{w}^-$ be the set of weights used in the target, and $\mathbf{w}$ be the weights that are being updated
- Slight change to computation of target value:
  - $(s, a, r, s') \sim \mathcal{D}$: sample an experience tuple from the dataset
  - Compute the target value for the sampled

$$s : r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'; \mathbf{w}^-)$$

  - Use stochastic gradient descent to update the network weights

$$\Delta \mathbf{w} = \alpha \Big( r + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'; \mathbf{w}^-) - \hat{Q}(s, a; \mathbf{w}) \Big) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

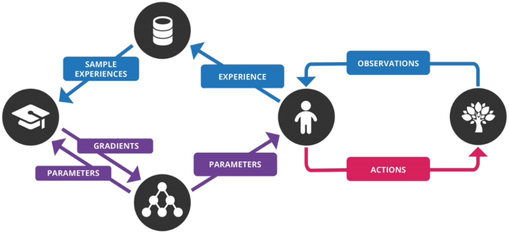
# DQN Strategy

DQN uses experience replay and fixed Q-targets

- Take action $a_t$ according to $\epsilon$-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$
- Sample random mini-batch of transitions $(s, a, r, s')$ from $\mathcal{D}$
- Compute Q-learning targets w.r.t. old, fixed parameters $\mathbf{w}^-$
- Optimise MSE between Q-network and Q-learning targets

$$\mathcal{L}_i(w_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_i}\left[\left(r + \gamma \max_{a\in\mathcal{A}} \hat{Q}(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

- Using variant of stochastic gradient descent



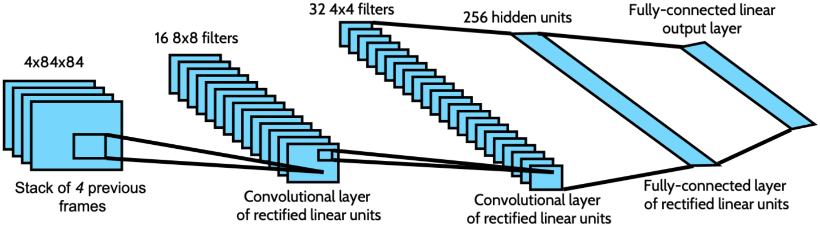
# DQN Algorithm

**Algorithm 1:** Deep Q-Learning with Experience Replay and Fixed Q-Targets

Initialize replay memory $\mathcal{D}$ to capacity $N$;
Initialize action-value function $q$ with random weights;
Initialize target action-value function $\hat{Q}$ with weights $\mathbf{w}^- = \mathbf{w}$;
**for** *episode* $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$;
  **for** $t = 1, T$ **do**
    Select $a_t = \begin{cases} \text{random action}, & \text{with probability } \epsilon \\ \arg\max_{a \in \mathcal{A}} Q(\phi(s_t), a; \mathbf{w}), & \text{otherwise} \end{cases}$;
    Execute action $a_t$ 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 $\mathcal{D}$;
    Set $y_j = \begin{cases} r_j, & \text{if episode terminates at step } (j+1) \\ r_j + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(\phi_{j+1}, a', \mathbf{w}^-), & \text{otherwise} \end{cases}$;
    Perform gradient descent over $\left(y_j - Q(\phi_j, a_j; \mathbf{w})\right)^2$ w.r.t. parameter $\mathbf{w}$;
    After every $C$ steps, reset $\mathbf{w}^- \leftarrow \mathbf{w}$;
  **end**
**end**

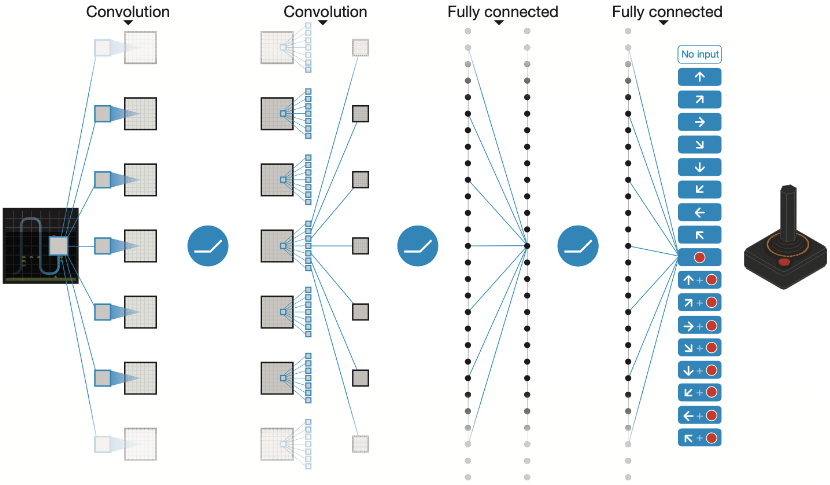
# DQN in Atari Games

- End-to-end learning of values $Q(s, a)$ from pixels $s$
- Input state $s$ is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



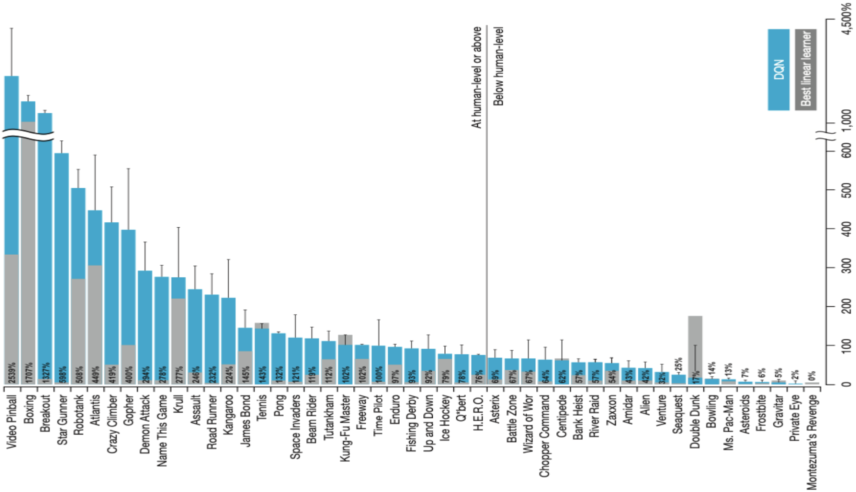

(Network architecture and hyperparameters fixed across all games)

# Human-level Control through DQN Models




**Ref:** Mnih et. al.; *"Human-Level Control through Deep Reinforcement Learning"*; Nature, 518:529-533, 2015.

# DQN Results in Atari Games




**Ref:** Mnih et. al.; *"Human-Level Control through Deep Reinforcement Learning"*; Nature, 518:529-533, 2015.

# Effects of Replay and Separating Target Q-network

| **Atari** | **DQN with Replay** | | **DQN without Replay** | | **Linear** |
|---|---|---|---|---|---|
| | **with Fixed Target Q** | **without Target Q** | **with Fixed Target Q** | **without Target Q** | |
| Breakout | 316.8 | 240.7 | 10.2 | 3.2 | 3.0 |
| Enduro | 1006.3 | 831.4 | 141.9 | 29.1 | 62.0 |
| River Raid | 7446.6 | 4102.8 | 2867.7 | 1453.0 | 2346.9 |
| Seaquest | 2894.4 | 822.6 | 1003.0 | 275.8 | 656.9 |
| Space Invaders | 1088.9 | 826.3 | 373.2 | 302.0 | 301.3 |

**Ref:** Mnih et. al.; *"Human-Level Control through Deep Reinforcement Learning"*; Nature, 518:529-533, 2015.

# Improvements over DQN Framework

- Double DQN (DDQN)
- Prioritized Replay DDQN
- Dueling DQN
- Distributional DQN
- Noisy DQN

# Double Deep Q-Networks (DDQN)

- Q-learning update for the (weight) parameters:

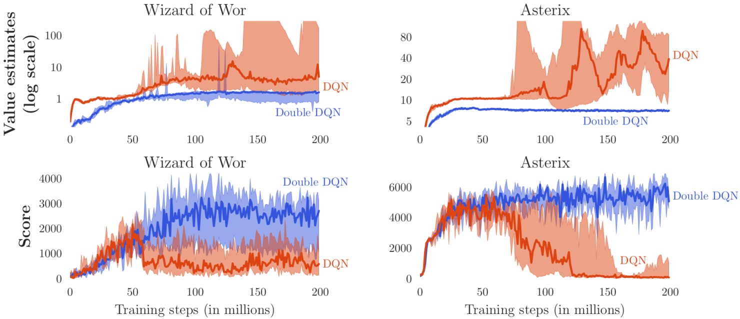
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha\big(y_t - Q(s_t, a_t; \mathbf{w}_t)\big)\nabla_{\mathbf{w}_t} Q(s_t, a_t; \mathbf{w}_t), \text{ where:}$$

  - (Standard) Q-learning target: $y_t = r_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \mathbf{w}_t)$
  - DQN target: $y_t = r_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \mathbf{w}_t^-)$
- Issue: There is an upward bias in $\max_{a \in \mathcal{A}} Q(s, a; \mathbf{w})$ estimates
- Double DQN maintains two sets of weight $\mathbf{w}$ and $\mathbf{w}^-$, so reduce bias by using:
  - $\mathbf{w}$ for selecting the best action
  - $\mathbf{w}^-$ for evaluating the best action
- Double DQN loss:

$$\mathcal{L}_i(w_i) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}_i}\Big[\Big(\underbrace{r + \gamma Q(s', \arg\max_{a' \in \mathcal{A}} Q(s', a'; w_i), w_i^-)}_{\text{Double DQN Target}} - Q(s, a; w_i)\Big)^2\Big]$$

# Performance of Double DQN

- Value estimates and scores by DQN and Double DQN on 6 Atari games



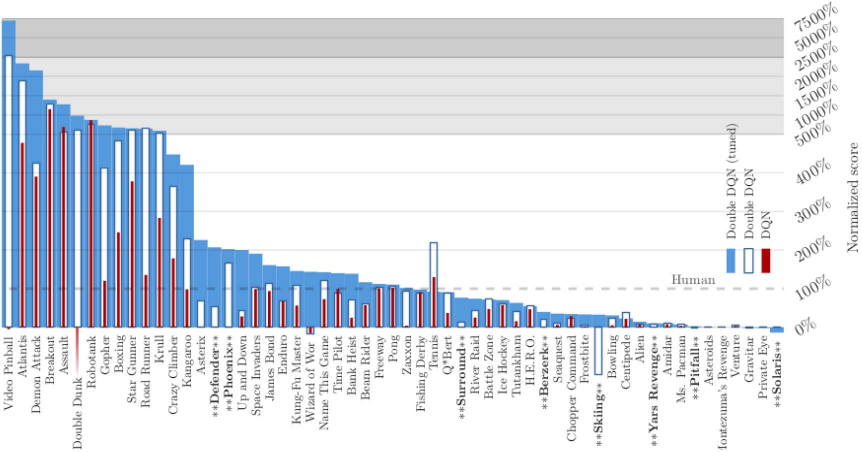

- Summary of normalized performance on 49 Atari games with human starts

| | upto 5 min of play | | upto 30 min of play | | |
|---|---|---|---|---|---|
| | DQN | **DDQN** | DQN | **DDQN** | **DDQN (tuned)** |
| Median | 93.5% | 114.7% | 47.5% | 88.4% | 116.7% |
| Mean | 241.1% | 330.3% | 122.0% | 273.1% | 475.2% |

**Ref:** Hasselt et. al.; *"Deep Reinforcement Learning with Double Q-learning"*; AAAI, pp. 2094–2100, 2016.

# Double DQN in Atari Games

Normalized scores on 57 Atari games, tested for 100 episodes per game with human starts




**Ref:** Hasselt et. al.; *"Deep Reinforcement Learning with Double Q-learning"*; AAAI, pp. 20942100, 2016.

# Prioritized Experience Replay (with DDQN)

- Replaying all transitions with equal probability is highly suboptimal
- Replay transitions in proportion to absolute Bellman error:

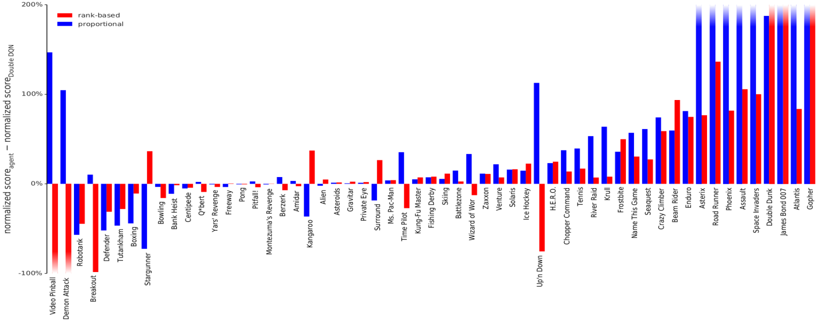
$$\left| r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \mathbf{w}^-) - Q(s, a; \mathbf{w}) \right|$$

- Leads to much faster learning
- Prioritized replay to DQN adds substantial improvements in (normalized) score on 41 out of 49 Atari games

| | **Double DQN** | | **Double DQN (tuned)** | | |
|---|---|---|---|---|---|
| | **Baseline** | **Rank-based** | **Baseline** | **Rank-based** | **Proportional** |
| Median | 48% | 106% | 111% | 113% | 128% |
| Mean | 122% | 355% | 418% | 454% | 551% |
| $>$ Baseline | – | 41 | – | 38 | 42 |
| $>$ Human | 15 | 25 | 30 | 33 | 33 |
| # Games | 49 | 49 | 57 | 57 | 57 |

# Prioritized Replay DDQN in Atari Games

- Substantial improvements in most games found through difference in normalized score (gap between random and human is 100%) on 57 games with human starts
- Comparing Double DQN with and without prioritized replay (rank-based as well as proportional variant)



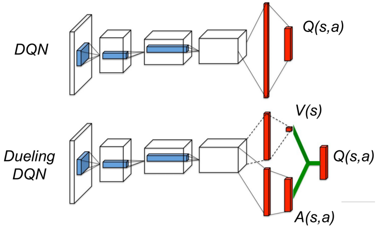

**Ref:** Schaul et. al.; *"Prioritized Experience Replay"*; arXiv preprint arXiv:1511.05952, 2016 (ICLR 2016 Poster).

# Dueling Deep Q-Networks

- Value-Advantage Decomposition of Q:

$$Q(s, a; \mathbf{w}, \alpha, \beta) = V(s; \mathbf{w}, \beta) + A(s, a; \mathbf{w}, \alpha), \text{ where}$$

  - $\mathbf{w}$ denotes convolutional layer parameters
  - $\alpha$ and $\beta$ are the FC-layer parameters of two streams



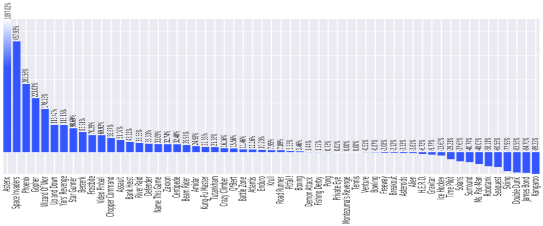

- *Identifiability Issue:* Given Q, cannot recover V and A uniquely
- To address this issue of identifiability, force advantage function estimator to have zero advantage at chosen action
  - $Q(s, a; \mathbf{w}, \alpha, \beta) = V(s; \mathbf{w}, \beta) + \left( A(s, a; \mathbf{w}, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \mathbf{w}, \alpha) \right)$
  - $Q(s, a; \mathbf{w}, \alpha, \beta) = V(s; \mathbf{w}, \beta) + \left( A(s, a; \mathbf{w}, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a' \in |\mathcal{A}|} A(s, a'; \mathbf{w}, \alpha) \right)$

# Dueling DQN in Atari Games

- Mean and median scores across all 57 Atari games, measured in percentages of human performance

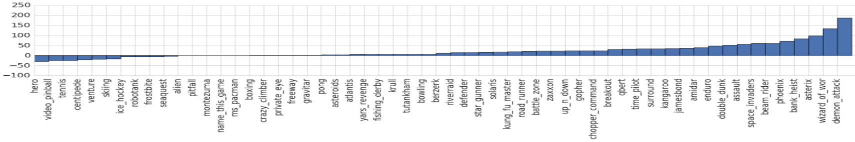
| DQN Variant | 30 no-ops | | Human Starts | |
|---|---|---|---|---|
| | Mean | Median | Mean | Median |
| Prior. Duel Clip | 591.9% | 172.1% | 567.0% | 115.3% |
| Prior. Single | 434.6% | 123.7% | 386.7% | 112.9% |
| Duel Clip | 373.1% | 151.5% | 343.8% | 117.1% |
| Single Clip | 341.2% | 132.6% | 302.8% | 114.1% |
| Single | 307.3% | 117.8% | 332.9% | 110.9% |
| Nature DQN | 227.9% | 79.1% | 219.6% | 68.5% |

- Improvements of dueling architecture over Prioritized DDQN baseline

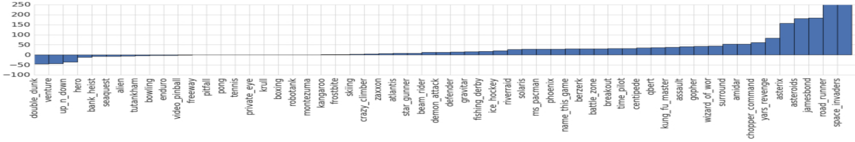



**Ref:** Wang et. al.; *"Dueling Network Architectures for Deep Reinforcement Learning"*; PMLR, 48:1995-2003, 2016.

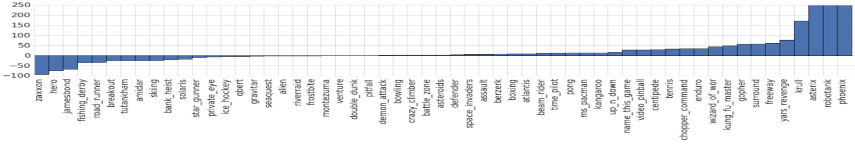
# Dueling DQN Performance in Atari Games




(a) Improvement in percentage of NoisyNet-DQN over DQN




(b) Improvement in percentage of NoisyNet-Dueling over Dueling

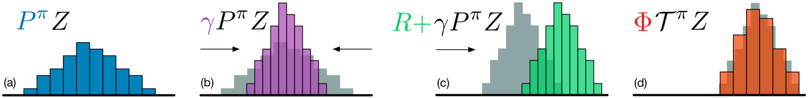



(c) Improvement in percentage of NoisyNet-A3C over A3C

**Ref:** Wang et. al.; *"Dueling Network Architectures for Deep Reinforcement Learning"*; PMLR, 48:1995-2003, 2016.

# Distributional Bellman Equations and Updates (no exam)

- Instead of learning expected return $Q(s,a)$, learn to estimate distribution of returns whose expectation is $Q(s,a)$
- The value function, $Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)]$, where $Z^\pi(s,a) = \sum_{t\geq 0} \gamma^t R(s_t,a_t)$ satisfies Bellman equation $Q^\pi(s,a) = R(s,a) + \gamma Q^\pi(s',a')$. $Z$ is the support of the return probability distribution (a categorical distribution)
- Distributional Bellman Equation:
  $Z^\pi(s,a) = R(s,a) + \gamma Z^\pi(s',a')$, where $s' \sim p(\cdot|s,a)$ and $a' \sim \pi(\cdot|s')$
- Distributional Bellman Operator: $T^\pi Z(s,a) = R(s,a) + \gamma Z(s',a')$



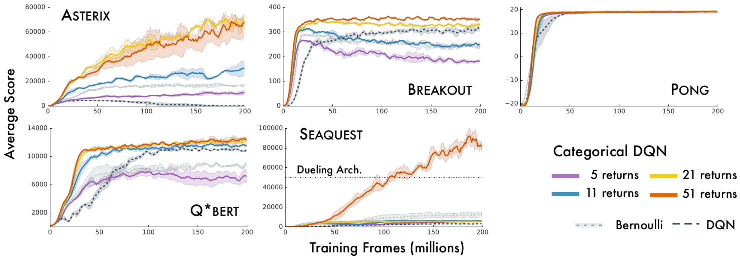

- Distributional Bellman Optimality Operator:
  $TZ(s,a) = R(s,a) + \gamma Z(s', \pi_Z(s'))$, where $\pi_Z(s') = \arg\max_{a'\in\mathcal{A}} \mathbb{E}[Z(s',a')]$

# Categorical/Distributional DQN Performance (no exam)

- Mean and median scores across all 57 Atari games, measured in percentages of human performance

| | DQN | Double DQN | Dueling DQN | Prior. DQN | Prior.Duel. DQN | Distributional DQN (C521) | UNREAL |
|---|---|---|---|---|---|---|---|
| Mean | 228% | 307% | 373% | 434% | 592% | 701% | 880% |
| Median | 79% | 118% | 151% | 124% | 172% | 178% | 250% |
| > Human | 24 | 33 | 37 | 39 | 39 | 40 | — |
| > DQN | 0 | 43 | 50 | 48 | 44 | 50 | — |

- Categorical DQN: Varying number of atoms in the discrete distribution. Scores are moving averages over 5 million frames.




**Ref:** Bellemare et. al.; *"A Distributional Perspective on Reinforcement Learning"*; PMLR, 70:449-458, 2017.

# Noisy Nets for Exploration (no exam)

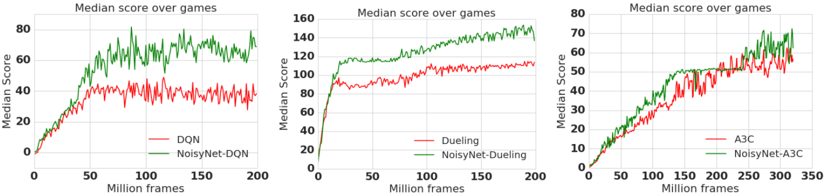
- Instead of $\epsilon$-greedy policy, Noisy Nets inject noise in the parametric space for exploration.
- Add noise to network parameters for better exploration
  - Use factorised Gaussian noise for DQN and Duelling
  - Use independent Gaussian noise for actor-critic method (distributed A3C)
- Noisy Nets are neural networks whose weights and biases are perturbed by parametric function of noise
  - *Standard Linear Layer:* $y = wx + b$, where $x \in \mathbb{R}^p$ is the layer input, $w \in \mathbb{R}^{q \times p}$ the weight matrix, and $b \in \mathbb{R}^q$ the bias
  - *Noisy Linear Layer:* $y = (\mu^w + \sigma^w \odot \epsilon^w)x + (\mu^b + \sigma^b \odot \epsilon^b)$, where $\mu^w \in \mathbb{R}^{q \times p}$, $\mu^b \in \mathbb{R}^q$, $\sigma^w \in \mathbb{R}^{q \times p}$ and $\sigma^b \in \mathbb{R}^q$ are learnable, whereas $\epsilon^w \in \mathbb{R}^{q \times p}$ and $\epsilon^b \in \mathbb{R}^q$ are random noise variables

# Noisy Nets in Atari Games (no exam)

- Mean and median scores across all 57 Atari games, measured in percentages of human performance

| Method | Baseline | | Noisy Net | | Improvement |
|---|---|---|---|---|---|
| | Mean | Median | Mean | Median | (On Median) |
| DQN | 319 | 83 | 379 | 123 | 48% |
| Dueling DQN | 524 | 132 | 633 | 172 | 30% |
| A3C | 293 | 80 | 347 | 94 | 18% |

- Comparison of learning curves of NoisyNet agent versus baseline according to median human normalised score




**Ref:** Fortunato et. al.; *"Noisy Networks for Exploration"*; arXiv preprint arXiv:1706.10295, 2017 (ICLR 2018).

# Rainbow: The Integrated DQN (no exam)

- *Distributional RL* estimate for return distribution than expected returns
- Replace 1-step distributional loss with *Multi-step Variant*
- This corresponds to a target distribution, $d_t^{(n)} = \left(R_t^{(n)} + \gamma_t^{(n)} \mathbf{z}, p_{\bar{\theta}}(s_{t+n}, a_{t+n}^*)\right)$, with the KL-loss as $D_{KL}(\Phi_{\mathbf{z}} d_t^{(n)} || d_t)$, where $\Phi_{\mathbf{z}}$ is a projection onto $\mathbf{z}$
- Combine distributional loss with multi-step *Double Q-Learning*
- Incorporate concept of *Prioritized Replay* by sampling transitions which are prioritized by KL loss
- Network architecture is a *Dueling Architecture* adapted for use with return distributions
- For each atom $\mathbf{z}^i$, corresponding probability mass $p_\theta^i(s,a)$ is calculated as below by using a dueling architecture:

$$p_\theta^i(s,a) \propto \frac{\exp\left(v_\eta^i(f_\xi(s)) + a_\psi^i(f_\xi(s),a) - \frac{1}{|\mathcal{A}|}\sum_{a'} a_\psi^i(f_\xi(s),a')\right)}{\sum_j \exp\left(v_\eta^j(f_\xi(s)) + a_\psi^j(f_\xi(s),a) - \frac{1}{|\mathcal{A}|}\sum_{a'} a_\psi^j(f_\xi(s),a')\right)}$$

- Finally incorporate noisy streams by *replacing all linear layers with their noisy equivalent*

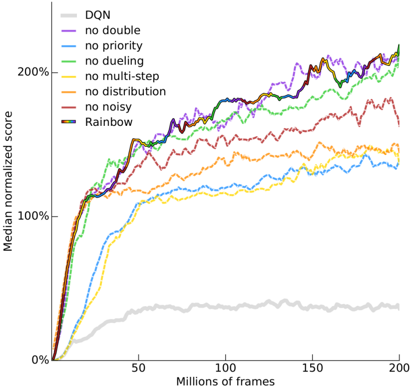
# Rainbow Performance Results (no exam)

- Median normalized scores of the best agent snapshots for Rainbow and baselines

| Agent | no-ops | human starts |
|---|---|---|
| DQN | 79% | 68% |
| DDQN | 117% | 110% |
| Prio. DQN | 140% | 128% |
| Duel. DQN | 151% | 117% |
| A3C | – | 116% |
| Noisy DQN | 118% | 102% |
| Dist. DQN | 164% | 125% |
| Rainbow | 223% | 153% |

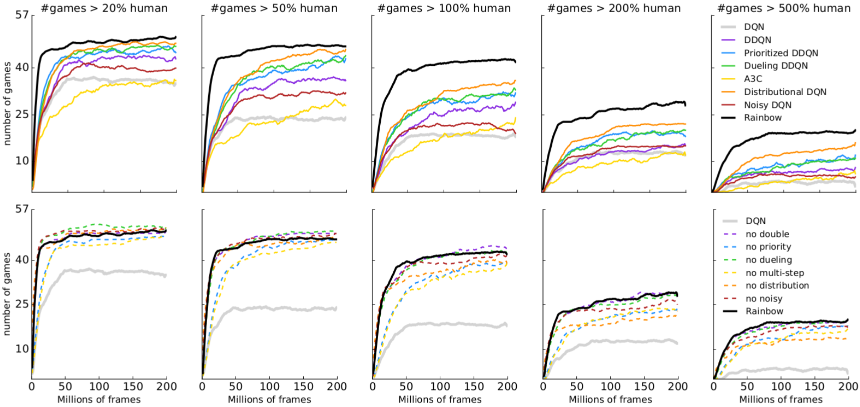
- **Demerit:** *Computationally intensive* – takes 10 days for 200M frames (full run) on 1 GPU (so 57 games would take 570 days on a single GPU)

- Median human-normalized performance across 57 Atari games, as a function of time




**Ref:** Hessel et. al.; *"Rainbow: Combining Improvements in Deep Reinforcement Learning"*; AAAI, 32(1):3215-3222, 2018.

# Rainbow Performance over Atari Games (no exam)




**Ref:** Hessel et. al.; *"Rainbow: Combining Improvements in Deep Reinforcement Learning"*; AAAI, 32(1):3215-3222, 2018.

# Thank You!

*Questions?*

*The only stupid question is the one you were afraid to ask but never did!* – Rich Sutton