Utilizing Curriculum Learning to Decrease Time to Threshold Performance of Deep Q Learning in a Simulated Highway Environment

Sean Hulseman

December 21, 2024

1 Introduction

Modern computers are capable of performing calculations at an astonishingly high speed—far beyond human comprehension. However, this raw computational power does not equate to intelligence or "common sense." In many computer science applications, the lack of cleverness is not a hindrance; the sheer processing speed compensates for it. Yet, when tasks become sufficiently complex—characterized by continuous actions, high dimensionality, and rare events—human cognition remains superior. For such tasks, like driving a car, it is impossible to "hard-code" instructions for every conceivable situation. Artificial intelligence (AI) agents, therefore, rely on specialized tools to make these tasks computationally feasible.

1.1 Challenges in Continuing Tasks

Self-driving systems, for instance, must process sensor data that is inherently uncertain and error-prone. Consider a scenario where a sensor detects an object signaling an imminent collision. The AI agent's response to such data—whether to stop or swerve—can involve unexpected and risky behavior, highlighting the complexity of decision-making in real-world environments.

Moreover, continuous variables pose additional challenges. [3]For example, in a driving task, a car that is 1 meter away is not significantly different from one that is 1.1 meters away. Preprocessing these observations by binning and clipping features can simplify the agent's calculations without sacrificing critical information. Additionally, continuous tasks can be segmented into shorter episodes to generate training data more efficiently, thereby accelerating the learning process.

1.2 Goals of This Paper

The primary goal of this research is to develop a custom Deep Q-Network (DQN) to train a reinforcement learning (RL) agent capable of driving reliably and safely while minimizing computational costs. Success in this task implies that subsequent episodes take longer to train, as each action builds upon a series of successful, non-terminal decisions. This challenge underscores the exploration-exploitation tradeoff, a fundamental concept in reinforcement learning, which can lead to exponential computational costs when using off-policy methods like DQNs.

The secondary goal is to accelerate the training process by introducing curriculum learning. To achieve this, smaller and faster-to-simulate environments were created, each designed to mimic aspects of the final goal environment. A training regimen was developed to assess the agent's learning and automate the transition to the goal environment in each experiment.

1.3 Findings

The experiments demonstrate that curriculum learning is effective at reducing the time required for a DQN agent to reach a performance threshold in a highway driving task. By progressively training the agent in simpler environments before transitioning to the target environment, the computational cost of training was significantly reduced.

2 Background: Transfer Learning

Transfer learning is a powerful technique in machine learning that focuses on leveraging knowledge gained from one task (the source task) to improve learning efficiency and performance in a related but distinct task (the target task). The concept is particularly valuable in reinforcement learning (RL), where the computational cost of learning from scratch can be prohibitively high.

2.1 Value Transfer

Taylor, Stone, and Liu's work on "[4]Transfer Learning via Inter-Task Mappings for Temporal Difference Learning" highlights the potential of transfer learning in RL:

- **Action-Value Function Transformation**: The transformation of an action-value function from the source task to the target task may not immediately outperform random actions in the target task. However, it biases the learner in a way that enables faster learning compared to training without transfer.
- **Policy Transfer**: By defining a functional transformation $\rho(Q)$ it becomes possible to apply a policy from the source task to the target task. This reduces the problem of policy transfer to transforming the action-value function.
- **Task Relatedness**: For transfer learning to be effective, the source and target tasks must be sufficiently related, and the relationship must be well-defined.

2.2 Evaluating Transfer Learning

Taylor also proposed several metrics for evaluating the effectiveness of transfer learning in RL:[4]

1. **Asymptotic Performance**: The performance of the learner after convergence in the target task.
2. **Initial Performance**: The learner's performance at the beginning of training in the target task.
3. **Total Reward**: The total accumulated reward during training in the target task.
4. **Area Ratio**: The area between the learning curves of transfer learning and non-transfer learning.
5. **Time-to-Threshold**: The time required to reach a predefined performance threshold in the target task.

This experiment evaluates the effectiveness of the value transfer with Time-to-Threshold.

2.2.1 Transfer Learning vs. Multitask Learning

Zhuang et al. provide further context by distinguishing transfer learning from multitask learning:[7]

- Multitask Learning: Simultaneously trains on multiple related tasks, leveraging inter-task relevance and differences to reinforce learning across tasks. Equal attention is given to all tasks.
- Transfer Learning: Focuses on transferring knowledge from related domains to a target task, with more emphasis on the target task than the source task.

Then challenges with multi-task Deep Reinforcement Learning are well documented[6]. When proposing this project the intention was around exploring transfer learning, but the process of mapping the features in the source to the target environments proved daunting.

2.3 Curriculum Learning

Curriculum Learning (CL) is inspired by the way humans learn, progressing from simpler tasks to more complex ones. This [5]"human-curriculum-like strategy" has shown to improve the efficiency and robustness of machine learning models. The primary motivations for applying CL in machine learning are twofold: guidance and denoising.

2.3.1 Motivations for Curriculum Learning

Guidance: Curriculum learning can improve both the training time and the performance of models, especially in challenging tasks where direct training often results in poor performance or slow convergence. By structuring the learning process, CL helps guide the model through an organized sequence of tasks or data, enabling it to build on prior knowledge incrementally.

A classic example is training models for difficult target tasks like self-driving vehicles or complex control systems. Direct training on such tasks often fails due to the high dimensionality and sparse reward signals. CL mitigates this by breaking the problem into manageable stages, each designed to progressively improve the model's capabilities.

Denoising: CL helps the model focus on cleaner, more relevant data first, which improves the robustness and generalizability of the final model. This approach is especially useful in domains where datasets are heterogeneous, noisy, or inconsistent.

A notable example is Neural Machine Translation (NMT), where datasets often vary in quality, difficulty, and noise levels. Sentences may differ in length, vocabulary, grammar, and annotation quality, making direct training challenging. CL helps by prioritizing simpler, cleaner translations early in training and gradually introducing more complex examples.

2.3.2 Defining Curriculum Learning

According to Xin Wang's A Survey on Curriculum Learning,[5] CL can be generalized as follows: Generalized Curriculum Learning: A curriculum is a sequence of training criteria over τ training steps. Each criterion Q_t encompasses various elements involved in training a machine learning model, such as:

- **Data/Tasks**: The progression of datasets or task complexity over time.
- **Model Capacity**: Adjusting the model's architecture or parameters as training progresses.
- **Learning Objective**: Dynamically refining the loss function to align with the current stage of learning.
- **Input Scheme**: Gradually increasing the complexity of input features.
- **Hypothesis Space**: Expanding the space of possible solutions as the model becomes more capable.

By iteratively modifying these criteria, CL provides a structured learning path that aligns with the model's evolving capabilities.

2.3.3 Applications of Curriculum Learning

The principles of CL are widely applicable, particularly in tasks with:

- Sparse or noisy rewards: Tasks where learning signals are rare or obscured by irrelevant data.
- High-dimensional input spaces: Environments with complex observations that require gradual abstraction.
- Heterogeneous datasets: Domains like NMT, where the quality and difficulty of data vary significantly.

In the context of Reinforcement Learning (RL), CL has been shown to significantly reduce training time while achieving superior performance. By structuring the learning process into stages, CL ensures that the model acquires foundational skills before tackling more advanced tasks, leading to better generalization and robustness.

3 Background Deep Q Networks[2]

The DQN algorithm introduced several innovations to address the challenges of reinforcement learning in complex tasks like playing Atari games. Here is the algorithm:

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**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ, select a random action a_t
    Otherwise, select $a_t = \arg\max_a Q^*(\phi(s_t), a; \theta)$
    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 $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

3.1 Experience Replay:

The agent stores its experiences at each timestep, $e_t = (s_t, a_t, r_t, s_{t+1})$, in a replay memory $D = \{e_1, e_2, \ldots, e_N\}$. These experiences are pooled over many episodes and sampled randomly during training. This technique:

- Reduces the correlation between consecutive samples.
- Stabilizes learning by breaking the sequence of states.

3.1.1 Stochastic Gradient Descent:

The DQN algorithm applies stochastic gradient descent to minimize the difference between predicted and target Q-values, enabling the agent to approximate optimal action-value functions even in high-dimensional state spaces.

The DQN paper demonstrated that RL agents could learn to play a wide range of Atari games directly from raw pixel inputs and achieve human-level performance in many cases. This success highlighted the potential of RL and deep learning to address complex, high-dimensional tasks and inspired subsequent advances in the field.

3.2 Target Q-values

In reinforcement learning, the goal is to estimate the action-value function $Q(s, a)$, which represents the expected return (reward) from taking action a in state s. The Deep Q-Network (DQN) algorithm uses the Bellman equation to compute the target Q-value for each transition. This target Q-value represents the ideal value that the agent should aim to approximate during training.

The target Q-value is computed using the Bellman equation:

$$Q_{\text{target}}(s, a) = r + \gamma \max_{a'} Q_{\text{target}}(s', a')$$

Where:

- r is the immediate reward.
- γ is the discount factor.
- $\max_{a'} Q_{\text{target}}(s', a')$ is the maximum predicted Q-value for the next state s' across all possible actions a'.

The target Q-value $Q_{\text{target}}(s, a)$ serves as a "supervised" target for the Q-learning update rule. It provides the agent with a reference for how much reward it should expect from taking action a in state s, considering both immediate rewards and the expected future rewards. The Bellman equation thus enables the agent to propagate the value of future rewards backward through time, which is critical for learning in environments with delayed rewards.

3.3 Loss Function

The loss function in DQN is used to measure the difference between the predicted Q-values (from the neural network) and the target Q-values (computed using the Bellman equation). The goal of training the Q-network is to minimize this loss, which in turn improves the agent's ability to estimate the optimal action-value function.

The loss function for DQN is typically the Mean Squared Error (MSE) between the target Q-values and the predicted Q-values:

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^{N} (Q_{\text{target},i} - Q_{\text{eval},i})^2$$

Where:

- N is the batch size.
- $Q_{\text{target},i}$ is the target Q-value for the i-th transition.
- $Q_{\text{eval},i}$ is the predicted Q-value for the i-th transition.

The update rule for the Q-network is based on the gradient descent algorithm, which adjusts the weights of the neural network to minimize the loss. By repeatedly applying this update process over many episodes, the Q-network gradually learns to approximate the optimal action-value function, ultimately enabling the agent to make better decisions in the environment.

4 Highway Environment and RL Variables

The highway environment used in this study was implemented using the `gymnasium` library[1], a popular toolkit for developing and comparing reinforcement learning (RL) algorithms. The specific environment, `highway-env`, is designed to simulate highway driving scenarios, offering a rich testbed for RL agents to learn and perform tasks such as lane-changing and speed control.

The environment provides the agent with observations and actions based on the current state of the simulation. The key variables involved in this environment are described below.

4.1 States - Kinematic Observations

The state space is defined by a set of kinematic features that describe the relative position and velocity of surrounding vehicles in the environment. These features are represented as a vector of 5 elements:

- **Presence (binary)**: Indicates whether a car is present or not. A value of zero means no vehicle is present, and all other features are zero as well.
- **Vx (continuous)**: The horizontal component of the velocity of the detected vehicle.
- **Vy (continuous)**: The vertical component of the velocity of the detected vehicle.
- **X (continuous)**: The relative distance between the ego car and the detected vehicle in the forward/backward direction (i.e., how far ahead or behind the vehicle is).
- **Y (continuous)**: The lateral distance of the detected vehicle from the ego car (i.e., how far to the left or right the vehicle is).

These kinematic observations provide the agent with information about the surrounding traffic, enabling it to make decisions based on the relative positions and velocities of other vehicles.

4.2 Actions

The action space in this environment consists of discrete meta-actions that control the movement of the ego vehicle. The actions available to the agent are:

- **0: 'LANE_LEFT'**: Move the vehicle one lane to the left.

Figure 1: Test images to show the smaller environments. Registering the custom environment is needed for the simulation to work as intended.

- **1: 'IDLE'**: Keep the vehicle in its current lane and speed.
- **2: 'LANE_RIGHT'**: Move the vehicle one lane to the right.
- **3: 'FASTER'**: Increase the speed of the vehicle.
- **4: 'SLOWER'**: Decrease the speed of the vehicle.

By limiting the action space, the complexity of the environment is reduced, making it more manageable for the RL agent to learn and explore. This simplification is particularly useful in experimental settings where the goal is to evaluate the effect of curriculum learning and other transfer strategies on the agent's learning efficiency.

4.3 Rewards

The standard highway environment calculates rewards based on two primary objectives.

1. Progressing quickly on the road: This is represented by a velocity term, which rewards based on the current speed of the ego-vehicle, normalized. The max and minimum velocities can be adjusted as needed. a=0.4 was standard and found to be effective
2. Avoiding collisions: A penalty is applied for collisions, scaled by a coefficient b. b=-1 was found to be affective as well so long as the rewards were normalized.

The overall reward function combines these terms:

$$R(s,a) = a \cdot \frac{v_{\max} - v_{\min}}{v - v_{\min}} - b \cdot \text{collision}$$

5 Making Custom Environments

The Gymnasium tool enables the creation of custom environments, utilizing the same framework as the standard environments. However, the computational resources required to run the simulations proved to be prohibitive. After testing various approaches, I found that the most reliable method for adjusting key variables—such as the number of lanes, the number of other vehicles, and vehicle behavior—was to develop three distinct environments. These environments were then used as components within the curriculum learning strategy.

5.1 Goal Environment Configuration

The goal environment used in this study is designed to simulate a highway driving scenario with specific parameters that affect the agent's learning and behavior. Below are the key configuration settings for the environment:

- **Duration**: 40 seconds (`duration` = 40). The simulation runs for a fixed duration of 40 seconds per episode.
- **Lanes Count**: 4 lanes (`lanes_count` = 4). The environment consists of 4 lanes. The early enviroments in the curricula have fewer lanes to reduces complexity.
- **Vehicles Count**: 6 vehicles (`vehicles_count` = 6). This setting defines the number of vehicles for which the Agent has kinematic values.
- **Simulation Frequency**: 5 Hz (`simulation_frequency` = 5). In this study, we reduced the frequency to make the simulations run faster.
- **Reward Speed Range**: [28, 32] (`reward_speed_range` = [28, 32]). The agent is rewarded for maintaining a speed within this range. The other cars are modeled as constant velocities ranging from 27,31
- **Off-road Terminal**: True (`offroad_terminal` = True). The agent's episode terminates if it drives off the road.

These configurations were carefully selected to create a challenging yet manageable environment for the agent, enabling faster simulations while isolating key variables in the experiment. Table 1 highlights the main differences between the custom environments, specifically the lane count and duration. These variables were deliberately chosen to isolate distinct tasks in the environment, such as maintaining velocity and heading, as well as avoiding collisions.

The theory is that driving on the highway is inherently simple: typically, you keep going straight and avoid steering off the road. The main challenge lies in responding to traffic conditions, which may require acceleration (positive or negative) to avoid collisions. After extensive testing, I found that the reward normalization method used in the standard highway environment maxed out around 350-360 in the goal configuration.

Custom Environments and Curricula

Experiment	Curriculum	Threshold(s)	Epsilon start	Lane Count	Duration (s)
4 (baseline)	'env1'	[300]	[0.5]*	4	40
3	['env2', 'env1']	[150, 300]	[0.5, 0.1]	[2, 4]	[20, 40]
2	['env3', 'env1']	[150, 300]	[0.5, 0.1]	[3, 4]	[30, 40]
1	['env2', 'env3', 'env1]	[150, 220, 300]	[0.5, 0.1, 0.01]	[2, 3, 4]	[20, 30, 40]

Table 1: Curriculum configurations used in the experiments. Lists are ordered left to right. Exp. 1 starts in 'env2' with initial epsilon of .5, two lanes, and a duration of 20 seconds. After reaching threshold of a 10-window rolling average of 150, the 'env2' is closed and 'env3' is created. The agent continues training retains the same Q-networks in 'env3' but the epsilon-greedy exploration value is reset to .1 to moderate exploration as desired

6 Experimental Results

The agent's performance in the baseline test (depicted by the red line in Figure 2) highlights the inherent difficulty of the task. As the episode progresses, the agent must make more non-terminal decisions to reach the end state. Random exploration at the tail end of a long sequence of decisions introduces the risk of reaching a new terminal state, and the rewards propagated back from this terminal state can lead to the learning of non-optimal behaviors.

In the baseline DQN, the rolling average value reaches 200 but then plateaus, signaling diminishing returns as training time increases. This demonstrates the need for strategies that can accelerate learning in DQN-based tasks.

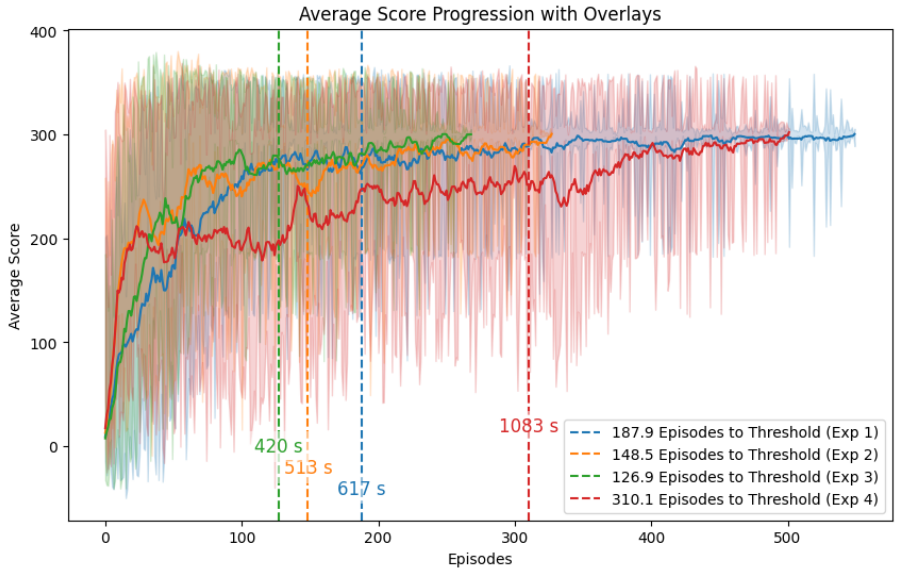

Figure 2: Plot of average scores by episode for each run. The average time to threshold is used to label the thresholds for each experiment, which may not align with the episode count. The episode number on the X-axis provides a more detailed view of the agent's learning progress. However, time is a valuable resource, and curriculum learning seems to reduce the computational cost of training for continuous tasks.

In contrast, the curricula exhibit slower initial learning (averaged across 10 runs) but do not suffer from the same premature plateauing issue. The results, as shown in the table below, indicate that Experiment 3 is the fastest in terms of average time to reach the performance threshold. All three curricula converged faster than the standard DQN.

Metric	**Experiment 1**	**Experiment 2**	**Experiment 3**	**Basic**
Average Times (s)	616.59	513.23	419.96	1083.38
Std. Dev. of Times (s)	620.5	390.8	308.6	503.6
Average Episodes	187.9	148.5	126.9	310.1
Std. Dev. of Episodes	159.3	105.7	82.2	147.0

Table 2: Summary of average times, standard deviations, and episodes for each experiment.

7 Conclusion

The results of this experiment clearly demonstrate the benefits of employing curriculum learning for complex tasks, such as the continuous task of driving on the highway. The curricula-based approach led to faster convergence, suggesting that curriculum learning can be an effective tool for improving training efficiency in reinforcement learning.

However, this experiment was limited in rigor. The testing was costly, and the conclusions are based on a relatively small number of runs for each curriculum. Hyperparameters were tuned and then applied uniformly across the different curricula, with limited fine-tuning during the process.

Additionally, the potential benefits of curriculum learning may have been influenced by the epsilon resetting method employed in this experiment. Future work with more time and resources could explore curriculum mapping further, potentially leading to even greater computational savings with minimal trade-offs in generalizability.

References

[1] Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] Bartow Sutton. *Reinforcement Learning*. Westchester Publishing Services, 2020.

[4] Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8:2125–2167, 2007. Submitted 11/06; Revised 4/07; Published 9/07.

[5] Xin Wang, Lichao Liu, Jing Yu, Li Zhang, and Yunchao Yang. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2022.

[6] Zhiyuan Xu, Wei Zhang, Tianyu Yang, Jure Xu, and Yifan Yu. Knowledge transfer in multi-task deep reinforcement learning for continuous control. *arXiv preprint arXiv:2010.07494*, 2020.

[7] Fuzhen Zhuang, Zhiqiang Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Heng Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.