Université de Montréal

Small batch deep reinforcement learning

par Johan Samir Obando Ceron

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

November, 2023

Université de Montréal
Faculté des arts et des sciences

Ce mémoire intitulé:

Small batch deep reinforcement learning

présenté par:

Johan Samir Obando Ceron

a été évalué par un jury composé des personnes suivantes:

Glen Berseth,	président-rapporteur
Marc Bellemare,	directeur de recherche
Pablo Samuel Castro,	codirecteur
Aaron Courville,	membre du jury

Mémoire accepté le: . . . .

Résumé

Dans l'apprentissage par renforcement profond basé sur la valeur avec des mémoires de relecture, le paramètre de taille de lot joue un rôle crucial en déterminant le nombre de transitions échantillonnées pour chaque mise à jour de gradient. Étonnamment, malgré son importance, ce paramètre n'est généralement pas ajusté lors de la proposition de nouveaux algorithmes.

Dans ce travail, nous menons une vaste étude empirique qui suggère que la réduction de la taille des lots peut entraîner un certain nombre de gains de performances significatifs ; ceci est surprenant et contraire à la pratique courante consistant à utiliser de plus grandes tailles de lots pour améliorer la formation du réseau neuronal. Ce résultat inattendu défie la sagesse conventionnelle et appelle à une compréhension plus approfondie des gains de performances observés associés à des tailles de lots plus petites.

Pour faire la lumière sur les facteurs sous-jacents, nous complétons nos résultats expérimentaux par une série d'analyses empiriques. Ces analyses approfondissent divers aspects du processus d'apprentissage, tels que l'analyse de la dynamique d'optimisation du réseau, la vitesse de convergence, la stabilité et les capacités d'exploration.

Le chapitre 1 présente les concepts nécessaires pour comprendre le travail présenté, notamment des aperçus de l'Apprentissage Profond (Deep Learning) et de l'Apprentissage par Renforcement (Reinforcement Learning). Le chapitre 2 contient une description détaillée de nos contributions visant à comprendre les gains de performance observés associés à des tailles de lots plus petites lors de l'utilisation d'algorithmes d'apprentissage par renforcement profond basés sur la valeur. À la fin, des conclusions tirées de ce travail sont fournies, incluant des suggestions pour des travaux futurs. Le chapitre 3 aborde ce travail dans le contexte plus large de la recherche en apprentissage par renforcement.

Mots-clés: Apprentissage par renforcement, Apprentissage par renforcement approfondi, Basé sur la valeur, Taille des lots

Summary

In value-based deep reinforcement learning with replay memories, the batch size parameter plays a crucial role by determining the number of transitions sampled for each gradient update. Surprisingly, despite its importance, this parameter is typically not adjusted when proposing new algorithms.

In this work, we conduct a broad empirical study that suggests *reducing* the batch size can result in a number of significant performance gains; this is surprising and contrary to the prevailing practice of using larger batch sizes to enhance neural network training. This unexpected result challenges the conventional wisdom and calls for a deeper understanding of the observed performance gains associated with smaller batch sizes.

To shed light on the underlying factors, we complement our experimental findings with a series of empirical analyses such as analysis of network optimization dynamics, convergence speed, stability, and exploration capabilities.

Chapter 1 introduces concepts necessary to understand the work presented, including overviews of Deep Learning and Reinforcement Learning. Chapter 2 contains a detailed description of our contributions towards understanding the observed performance gains associated with smaller batch sizes when using value based deep reinforcement learning algorithms. At the end, some conclusions drawn from this work are provided, including some exciting suggestion as future work. Chapter 3 talks about this work in the broader context of reinforcement learning research.

Keywords: Reinforcement Learning, Deep Reinforcement Learning, Value based, Batch Size

Contents

List of Figures

List of Abbreviations

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
TD	Temporal Difference
ALE	Atari Learning Environment
MSE	Mean Squared Error
HNS	Human-Normalized Score
MLP	Multi-layer Perceptron
ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
IQM	Interquartile Mean
BBF	Bigger, Better, Faster

Acknowledgments

First and foremost, I extend my gratitude to my family members—Isabel, Orlando, Maira, Marina, and Yuliana—as well as to the Obando-Ceron relatives and all my wonderful friends. They have been unconditionally supportive and they always encouraged me to figure out for myself what I wanted to do in life which is something I will forever be grateful for.

I am very grateful for the support I have received at Mila. I particularly wish to thank Marc Bellemare and Pablo Samuel Castro, who have taken immense amounts of time to introduce me to machine learning research and profoundly shaped my research career thus far. Thanks a lot Marc for giving me the opportunity to be your student, and be able to live this amazing academic experience. Pablo, once again, thank you, thank you and thank you for believing in me and your continuous support.

Infinite thanks to my roommate, Gopeshh Subbaraj. I have no words to thank you for everything. You are the best, thanks friend. I also wish to thank my coauthors, who have made my experience at Mila as productive as it has been: João, Max, Rishabh, Aaron, Marc and, of course, Pablo Samuel Castro.

I would also like to thank my other friends at Mila for interesting and fruitful conversations on machine learning and other topics, including the above as well as but not limited to Ghada, Riashat, Nizar, Albert, Aniket, Moskh, Naga, Adrien, Jesse, Evgenii, Pierluca, Juan Duque, Juan Ramirez, Mahan Fathi, and Zhixuan.

I am also extremely thankful for the astonishing computational resources provided to me at Mila, both through the Mila cluster and Compute Canada. Finally, the work reported in this thesis would not have been possible without the financial support from: Google research, Compute Canada, the Canada Research Chairs and CIFAR.

To my friends, family, and mentors who are no longer with us, RIP and thank you for everything. Love you all so much!

1 Introduction

In this chapter, we will present an overview of the fundamental concepts required to understand the contributions presented in this thesis. Our work resides at the intersection of the fields of deep learning and reinforcement learning, and we organize the overview accordingly. We first introduce important concepts in machine learning and we briefly discuss the emergence of deep learning in recent years and what characterizes the improvements it can bring to our work.

We introduce the field of reinforcement learning (RL), focusing on reinforcement learning techniques using neural networks, or deep reinforcement learning (DRL). We then briefly explain some value based DRL algorithms which are extensively used in this thesis. We assume knowledge of basic techniques and terminology in deep learning, including convolutional, feedforward, and recurrent neural networks, gradient-based optimization and regularization. A foundational understanding of conventional supervised learning tasks, such as classification and regression, is also presumed. For readers unfamiliar with these topics, we recommend the following books: *Pattern Recognition and Machine Learning* (Bishop and Nasrabadi, 2006) and *Deep Learning* (Goodfellow et al., 2016).

1.1 Deep Learning

In this section, we will introduce background material on representation learning in deep learning and the relationship with the amount of data sampled when training neural networks models. We begin with early developments in the field and preliminary studies on deep learning models and batch size. We wrap up with a discussion of recent discoveries, from which this work draws directly. It's important to note that this isn't intended to serve as an exhaustive overview of representation learning and batch size for supervised learning tasks in deep learning; we present only a few essential papers, leaving out numerous significant historical topics not essential to understanding the techniques employed in this thesis.

Machine Learning is the field of study focused on computational systems that possess the ability to learn and adapt from data without explicitly being programmed. Although the terminology has remained consistent since Samuel's Checkers-playing program in 1959, the landscape of learning has evolved significantly. It entails the development of learning algorithms to tackle complex tasks where it is challenging (or practically impossible) to construct explicit step-by-step instructions. Some of these tasks were detecting intricate patterns, extracting information, performing reasoning, and decision-making, among others. As any intelligent agent should be capable of adapting its behavior based on observations and experiences, the study of machine learning plays a crucial role in the advancement of artificial intelligence.

Designing algorithms to solve challenging tasks using rule-based systems, where a predetermined set of rules is defined by humans, proves exceedingly difficult. Machine learning algorithms, on the other hand, leverage statistical regularities present in the data to learn from examples. The primary objective of such learning procedures is generalization, which sets them apart from template matching algorithms or look-up tables.

Deep learning is a subfield of machine learning that focuses on the development and application of artificial neural networks, which are inspired by the structure and function of the human brain. Deep learning algorithms automatically learn and extract higher-level features from raw large amounts of data, such as images, text, or audio. These have gained significant attention and achieved remarkable success in various complex domains such as natural language processing, speech recognition and image, and game playing.

Deep learning algorithms are composed of multiple layers of interconnected artificial neurons, forming what is known as a neural network. Each neuron receives inputs, performs a mathematical operation on those inputs, and produces an output that is passed on to the next layer of neurons. The network learns to adjust the parameters of its neurons based on the patterns in the data it is trained on, with the goal of optimizing its performance on a specific task, such as image recognition or natural language processing.

The core focus of deep learning is to learn multiple levels of representation of data and coming up with higher levels of abstraction (LeCun et al., 2015; Schmidhuber, 2015). This approach draws inspiration from studies in neuroscience, which suggest that the human brain processes information in multiple stages (McCulloch and Pitts, 1943). Moreover, the hierarchical structure aligns well with the inherent characteristics of the data itself. For example, in language, words combine to form sentences, and sentences construct paragraphs in a document. In vision, pixels contribute to edges, edges contribute to basic shapes, and these shapes collectively compose more complex structures in a natural image.

While deep learning does not impose strict assumptions about the learned representations, the concept of distributed representations plays an important role (Hinton, 1984). Here, as opposed to learning local representations, the key idea is to distribute information about data observations across multiple dimensions of the feature space.

By doing so deep learning neural networks can express highly complex relationships between features (*Expressiveness*). This allows the network to model intricate patterns and dependencies that may not be evident in individual features or local representations. The network can learn to combine features in non-linear ways, enabling it to capture the complex interactions and dependencies that exist within the data.

1.1.1 Neural networks

We use Artificial Neural Networks (ANNs) as multi-layered non-linear function approximators, drawing loose inspiration from the biological neural networks found in animal brains. ANNs are not algorithms per se, but rather structures composed of multiple layers of mathematical transformations applied to input values. A prominent example of a neural network architecture is the Feedforward Neural network or multi-layer perceptron (MLP) (Rumelhart et al., 1986). This model consists of one or more layers of arbitrary (usually, differentiable) functions. In general, a single layer is a linear transformation followed by a non-linear transformation.

Let's start with the standard case of a single hidden layer neural network as an example, where vector-valued inputs are mapped to vector-valued outputs, as in regression tasks. Let's consider:

$$h(x) = b + Wa(c + Vx)$$

where,

$\Rightarrow x$ is a d-dimensional vector (input)

$\Rightarrow V$ is a $k \times d$ matrix (input-to-hidden weights)

$\Rightarrow c$ is a k-dimensional vector (hidden units offsets or biases)

$\Rightarrow b$ is an m-dimensional vector (output units offset or biases)

$\Rightarrow W$ is an $m \times k$ matrix (hidden-to-output weights)

$\Rightarrow a$ is a threshold-like (non-linear) activation function differentiable almost everywhere (e.g. Tanh, or ReLU function) applied element-wise.

The vector-valued function $h(x) = a(c + Vx)$ is called the output of the hidden layer and we call hidden units the elements of the hidden layer. The neural network's learnable parameters consist of weights and biases. Essentially, the operation performed by $h(x)$ can be applied to $h(x)$ itself, but with different biases and weights (different parameters).

By extending the aforementioned approach, we can construct a deep neural network with two hidden layers. One can create a feed-forward multi-layer network by stacking additional layers, each with potentially different dimensions (represented as k in the previous example). Throughout this thesis, we will typically represent the complete set of learnable parameters in a function approximator using lowercase *Greek* letters, such as θ or ω.

A widely used variant in neural networks involves incorporating skip connections, allowing a layer to receive input not only from the preceding layer but also from select lower layers. Residual neural networks accomplish this by using skip connections to jump over some layers, for instance, ResNets (He et al., 2016) or DenseNets (Huang et al., 2017). Silver et al. (2017) and Espeholt et al. (2018) are examples of reinforcement learning methods using a deep residual network architecture. The parameters of each layer in the network are learnable, meaning they are modified by an algorithm until the network approximates the target function with a satisfactory degree of accuracy. This adjustment process, commonly termed learning or training, is a key aspect of neural networks which will be explained in later sections. Despite various algorithms having been introduced for training Multi-Layer Perceptrons (MLPs), the most prevalent approaches are based on stochastic gradient descent (Reddi et al., 2019; Ruder, 2017; Goodfellow et al., 2016).

1.1.2 CNN networks

Convolutional neural networks (CNNs) are a cascade of layers with each layer generating an abstract representation of input by applying transformations across interconnected layers (LeCun et al., 1989). These networks draw inspiration from the human brain's processing of visual information, mimicking its mechanisms (Patterson and Gibson, 2017). The primary feature of CNNs are the convolutional layers that make up the network. The convolutional layers consist of filters that act upon the input image x in a pixel-wise manner to produce an output y, at a given pixel position (i, j). Each of these filters, of dimensions M by N, has a corresponding set of learnable parameters—weights denoted as W, and biases, b, that the network learns.

Every convolutional layer generates feature maps by convolving the input with the corresponding layer's filter. The output feature maps vary in their abstraction levels based on the layer's position within the network. Lower level semantic features such as edges or colors are closer to the input layer, while mid to high level semantic features such as object parts or entire objects themselves can be determined near the output at the later layers.

In addition to the convolutional layers, a CNN is constructed by incorporating supplementary layers, each playing distinct roles in shaping the ultimate output of the network. Some of these common layers include pooling layers, unpooling layers, batch normalization layers (Ioffe and Szegedy, 2015), dropout layers (Srivastava et al., 2014), linear layers, and activation layers (Sharma et al., 2017).

Pooling layers serve to decrease the input's scale, reducing the number of network computations (Scherer et al., 2010). This downsampling reduces the feature map resolution. With the lower feature map resolution, spatial feature invariance is improved, making the network more robust to translational shifts. Max pooling and average pooling are typical methods employed in pooling, offering a means to extract salient information (Lee et al., 2016). Conversely, unpooling layers perform the approximate inverse function of the pooling layers by upsampling the input data.

Batch normalization layers compute normalized outputs across each dimension of the layer's input (Ioffe and Szegedy, 2015). This normalization occurs prior to the activation layers, serving to stabilize the activation distribution and acting as a regularizer technique. This regularization allows for much higher learning rates to be used in training, with less emphasis needed on the initialization of weights, while simultaneously enhancing accuracy and speeding up convergence time. Dropout Layers are similar to batch normalization layers, in that dropout layers also work to regularize the neural network (Srivastava et al., 2014). However, dropout layers do this regularization by introducing noise to the units within the neural network layer, and randomly removing some of these units. The removal of the units additionally aids to prevent overfitting of the network.

The relationship between the depth of a CNN and its performance has been studied through the development of VGGNet, revealing that deeper networks result in higher accuracy for image classification tasks (Simonyan and Zisserman, 2014). Additionally, this high performance from deep networks can be extended to other object recognition or localization tasks as well. Based on a standard CNN architecture, which usually has fewer than five convolutional layers, VGGNet extends the number of convolutional layers to form a 19-layer network.

1.1.3 Training

In the standard set-up for gradient descent, we address the general optimization problem of minimizing a loss function $\mathcal{L}$: $min_\theta \ \mathcal{L}(\theta)$. This function takes the neural network's parameters, denoted as $f(\theta)$, as input and returns a scalar value that quantifies how well the network represents the target function. Given access only to first-order evaluations of L, the parameters θ are iteratively updated by stepping in the opposite direction of the gradient until convergence: $\theta_{k+1} \leftarrow \theta_k - \alpha_k \bigtriangledown \theta_k L$. Here, k is the index of the update iteration and α_k is the learning rate.

We distinguish three variants of gradient descent which are called batch gradient descent, stochastic gradient descent, and minibatch gradient descent.

The first one follows a gradient computed on the whole dataset. The second follows a gradient computed on a single example sampled uniformly from the dataset. The third one, which is the one we will use in practice and sits in the middle of both, processes a small subset of the training examples at a time. *Minibatch gradient descent* is our method of choice because computational considerations, batch gradient descent demands fitting the whole dataset into memory (often not realistic), while stochastic gradient descent suffers from too much variance.

Minibatch gradient descent still has variance, albeit reduced compared to the stochastic version, and it has been shown that this remaining variance aids in exploring the parameter space, effectively serving as a form of regularization. In what follows, we employ two different gradient descent optimizers: Adam (Kingma and Ba, 2014) and RMSProp (Tieleman and Hinton, 2017). These optimizers adjust the learning rate α based on approximations of the first and second moments of past gradients.

Overall, the procedure used for computing gradients in the case of feed-forward multi-layer networks is called the back-propagation algorithm (Rumelhart et al., 1985; LeCun et al., 1988). In broad terms, diverse strategies for MLP training are guided by two core considerations: (i) training as efficiently as possible, which involves reducing training error while sidestepping local minima traps, and (ii) controlling the expressiveness of the neural network subject to the amount of training data so as to prevent overfitting and subsequently minimize generalization error.

1.1.4 Optimization

Thoughtfully configuring the neural network structure, fine-tuning hyperparameters, and defining an appropriate regularization for the loss function are essential factors to consider before the training of neural networks. Despite their significance, these crucial design choices can be nuanced and occasionally remain implicit in scientific literature. For example, the difficult task of training deep neural networks is notably influenced by the choice of network initialization (Goodfellow et al., 2016). A few widely adopted initialization techniques include (Glorot and Bengio, 2010) and (He et al., 2016), and all depend on the neural network design choices (network structure and activation functions).

Furthermore, various principles govern the establishment of the learning rate, and in many cases, a fixed value is not employed due to potential issues arising from excessively large or small gradients. By automatically adjusting the learning rate, one aims to maintain gradients within favorable ranges that evade problems such as error landscape plateaus (where gradients are too small) or may overstep minima, preventing the goal of finding good solutions (when gradients are too large). Adam optimization (Kingma and Ba, 2014) builds upon the idea of adaptive learning rates from AdaGrad (Duchi et al., 2011) and RMSProp (Tieleman and Hinton, 2017).

Unless specified, we use Adam as our optimization method in the rest of this thesis.

1.2 Reinforcement Learning

Reinforcement learning is a subfield of machine learning that studies how agents learn behavioral patterns, referred to as *policies*, to maximize an objective, denoted as *reward*, while interacting with an environment. In the standard reinforcement learning setting (see Sutton and Barto, 2018), agents interact with a Markov Decision Process (MDP), characterized by a set of states $\mathcal{S}$, a set of possible actions A, a set of possible rewards $\mathcal{R}$, a state transition distribution $p(s'|s,a)$ and a reward distribution $p(r|s',s,a)$ [1] (Puterman, 2014).

Agents interact with their environment by selecting actions based on a policy $\pi(a|s)$, which is a probability mass (or density, if $\mathcal{A}$ is infinite) function. Once an action is selected, agents observe a reward and the next state; this sequence of state-action-reward-state is often termed as a *transition*. Agents' interactions are frequently categorized into *episodes*, and are denoted as $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$. These episodes may be infinite, in which case the quantity of interest is the discounted sum of rewards (or *return*), defined as $G_t \triangleq r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{k=t}^{\infty} \gamma^{t-k} r_k$, where $\gamma \in [0,1)$ is a *discount factor* that causes the agent to prioritize nearer rewards (Sutton and Barto, 2018).

Agents frequently estimate these returns using a learned *value function*, $V : \mathcal{S} \to \mathbb{R}$, which is trained to approximate the true expected return of a state $v_\pi(s) \triangleq \mathbb{E}_\pi[G_t|s_t = s]$. Alternatively, many algorithms opt to estimate an action-conditioned variant of the value function, generally denoted as $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R} : Q^*(s,a) \triangleq \mathbb{E}[G_t|s_t = s, a_t = a]$. When actions are chosen based on a policy π, estimating Q is strictly more general than estimating V, given that a value function V can be recovered by calculating the expectation of Q over actions: $V(s) = \mathbb{E}_{a\sim\pi(a|s)}[Q(s,a)]$.

1. These take the form either of probability mass functions or probability density functions, depending on the finiteness of $\mathcal{S}$ and $\mathcal{R}$.

1.2.1 TD Learning

While numerous techniques exist to estimate Q and V, the methods used in this thesis are based on *temporal-difference* (TD) learning. In TD methods, agents learn from individual transitions of the form $(s_t, a_t, r_t, s_{t+1}) \in \mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S}$, updating their estimates of $V(s_t)$ or $Q(s_t, a_t)$ using their estimate for the value of s_{t+1}. In the context of value learning, the *TD error* of a transition is characterized as $\delta_t \triangleq r_t + \gamma V(s_{t+1}) - V(s)$. Under the premise of unlimited data and particular optimization assumptions (Robbins and Monro, 1951), the correct value function can be learned through iterative minimization of this error, resulting in convergence with a rate determined by γ (Sutton and Barto, 2018).

This work predominantly centers on Q-learning (Watkins and Dayan, 1992), a variant of TD learning used to jointly estimate the optimal policy π^*, defined as $\pi^* \triangleq \arg\max_\pi \mathbb{E}_{\pi,s_o}[G_0]$, and the optimal value function, expressed as $q^*(s,t) = \mathbb{E}_{\pi^*}[G_t|s_t = s, a_t = a]$. In scenarios where the MDP is perfectly known, and states and actions can be enumerated, an agent's estimate of Q can be iteratively improved by applying the Bellman optimality operator, defined as $Q_{i+1}(S_t, A_t) = \mathbb{E}[r_t + \gamma \max_a Q_i(S_{t+1}, a)]$ (Sutton and Barto, 2018); this operator converges to the optimal policy $\pi = \pi^*$ and Q function $Q = q^*$ at its fixed point. Conversely, when the MDP is not known, Q must be learned from data, generally done by minimizing a TD error corresponding to the operator $\delta_t \triangleq r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s,a)$.

If the state and action spaces are small, one can store all the Q-values in a table of size $|\mathcal{S}| \times |\mathcal{A}|$. For most problems of interest, however, state spaces are very large (and possibly infinite). In these cases, one can use a function approximator, such as a neural network, parameterized by θ: $Q_\theta \approx Q$.

1.2.2 Off-Policy Learning

The objectives established in TD learning involve multiple expectations: over the agent's policy, the environment's reward distribution, and the environment's transition function. In cases where the MDP in question is fully known, these expectations are not problematic, as they can be directly evaluated. However, real-world scenarios often deviate from this ideal, necessitating the approximation of these expectations through samples drawn from the agent's interactions with the environment.

Strategies for doing so can be categorized into two classes: *on-policy* and *off-policy*, determined by whether the data utilized is collected according to the current policy π or some other policy or set of policies. Off-policy methods offer notable theoretical advantages, enabling agents to learn from arbitrarily-collected data, but frequently suffer from instability, particularly when combined with TD learning and function approximators such as neural networks (Sutton and Barto, 2018).

In the context of Q-learning, the agent's policy often leans towards determinism. This allows the agent to learn what is strictly the optimal policy, it means that the agent will generally not select a wide-enough variety of actions leading to insufficient exploration of its environment. This limited exploration can result in situations where the best action for a given state remains unexplored. Consequently, in Q-learning, it's a common practice for agents to collect data in their environment according to stochastic exploration policy based on π but with added noise. As a result, data for Q-learning is *never* sampled according to the agent's actual policy (Schwarzer et al., 2021).

1.2.3 Deep Reinforcement Learning

In scenarios where states and actions can be explicitly enumerated, the learning process for Q involves constructing a matrix of dimensions $|\mathcal{S}| \times |\mathcal{A}|$, and V can be learned as an $|\mathcal{A}|$-dimensional vector. In broader contexts, however, function approximation must be used for Q and V. Among parametric function approximation methods, both linear regression and neural networks are frequently adopted. In cases where linear function approximation is employed, some guarantees of stability or performance are available (Sutton and Barto, 2018). Nonetheless, in practice nonlinear function approximation is commonly used in numerous instances, particularly those with visual inputs (for example, see Mnih et al., 2015).

When using parametric function approximation with Q-learning, as we do in this work, it's a common practice to minimize the subsequent objective using gradient descent on the parameters of Q:

$$\mathcal{L}_Q(s_t, a_t, r_t, s_{t+1}) = (Q(s_t, a_t) - \mathbb{E}[r_t + \gamma \max_a Q_t(s_{t+1}, a)])^2 \tag{1.1}$$

where Q_t is a separate "target" function reflecting an older version of Q. It is key that Q not be modified by gradients taken through the target Q_t, even when Q_t

is defined to be the same as Q; failing to do this can often lead the algorithm to arrive at incorrect solutions (Sutton and Barto, 2018). In practice, this objective is optimized using samples taken from the environment. The data gathered by the agent is stored in a *replay buffer*, a buffer of the agent's most recent experiences. It's typical for buffer sizes to range up to several million transitions.

Deep Q-Learning, also known as Deep Q Networks (DQN), has emerged as a particularly successful technique in reinforcement learning. This success is primarily attributed to the incorporation of numerous algorithmic enhancements that go beyond the foundational Q-learning algorithm; when combined, in Rainbow (Hessel et al., 2018), the resulting algorithm brings enormous benefits, improving by far efficiency and final performance over DQN agent (Mnih et al., 2015). We refer the reader to (Hessel et al., 2018) for a full summary of these improvements.

Deep reinforcement learning has witnessed notable algorithmic advancements, especially in the realm of distributional RL (Bellemare et al., 2017; Hessel et al., 2018). Distributional RL approaches go beyond traditional RL methods by considering not just the expected return but also the entire distribution of possible returns (Bellemare et al., 2023). Instead of estimating a single value for each state-action pair, distributional RL agents model the distribution of possible returns, providing a more comprehensive understanding of the uncertainty and variability associated with different actions.

Different ways of parameterizing return distributions were proposed in the form of IQN (Dabney et al., 2018b) and QR-DQN (Dabney et al., 2018a) algorithms. QR-DQN agent (Dabney et al., 2018a) computes the return quantile values for N fixed, uniform quantiles. This has no restrictions or bound for value, as the distribution of the random return is approximated by a uniform mixture of N Diracs: $Z_\theta(x,a) := \frac{1}{N}\sum_{i=1}^{N}\delta_{\theta_i(x,a)}$, with each θ_i assigned a quantile value trained with quantile regression. IQN agent uses implicit quantile networks (IQN) as the parameterization of the return distribution (Dabney et al., 2018b). IQN learns to transform a base distribution (typically a uniform distribution in $[0,1]$) to the quantile values of the return distribution.

1.2.4 Replay buffer and batch size

A replay buffer, also known as an experience replay buffer, is a data structure used to store and manage past experiences or transitions encountered by an RL agent. Each experience typically consists of the agent's current state, the action it took, the resulting reward, and the next state it transitioned to. By storing and randomly sampling experiences from the replay buffer, RL algorithms can break the sequential correlation of experiences and alleviate issues such as catastrophic forgetting.

Replay buffers are a valuable component that enhance the efficiency of algorithms by enabling the reuse of data for multiple training iterations, rather than discarding it immediately after collection. This reuse of data leads to improved sample efficiency. Additionally, the replay buffer contributes to the stability of the network during training.

Additionally, the replay buffer helps in creating a diverse and uncorrelated dataset for training. It collects experiences during the agent's interactions with the environment and stores them in memory. These experiences are then sampled randomly during the training process. By randomly sampling transitions from the replay buffer, the RL algorithm can learn from a more diverse set of experiences and reduce the bias introduced by sequential data (Mnih et al., 2015).

During training, the reinforcement learning algorithm updates its policy or value function using a batch of experiences rather than updating after each individual experience. The batch size refers to the number of experiences sampled from the replay buffer in each training iteration. The choice of an appropriate batch size depends on various factors, such as the complexity of the RL task, the available computational resources, and the specific algorithm being used. It is often determined through experimentation and balancing the trade-off between computational efficiency and learning stability.

The replay buffer is typically implemented as a circular buffer, where the oldest transition in the buffer is removed to accommodate the new data that was just collected. The most basic sampling strategy used is uniform sampling. In 2015, Schaul et al. (2015) introduce prioritized experience replay, a method that assigns priorities to experiences stored in the replay buffer based on their importance for learning. Other variations, such as a distributed experience replay buffer (Horgan et al., 2018), can be used.

1.2.5 Plasticity and Stability

Plasticity in reinforcement learning refers to the ability of an agent to adapt and update its behavior based on new experiences or changes in the environment. It involves modifying the agent's policy or value function to improve its performance over time. Plasticity is a crucial aspect of reinforcement learning as it allows an agent to learn from interactions and adjust its decision-making process accordingly.

Issues with plasticity and related phenomena have recently garnered attention in deep RL under a plethora of various terminologies. Lyle et al. (2022) show *loss of capacity* for fitting targets in online RL and Kumar et al. (2021a) demonstrate a related implicit underparameterization phenomenon caused by bootstrapping with a greater emphasis on the offline RL scenario.

Both of these studies employ the *feature rank* as a proxy measure for plasticity. Feature rank corresponds to an approximation of the rank of a feature embedding, which represents how well states can be differentiated by updating only the final layer of a neural network. It serves as a measure of the network's capacity to adapt rapidly to variations in the target function.

The works conducted by Sokar et al. (2023), Graesser et al. (2022) and, Abbas et al. (2023) center around the topic of neuron saturation during the training process, but Lyle et al. (2023) demonstrate that the phenomenon of plasticity loss cannot be entirely attributed to saturation alone. Nikishin et al. (2022) discuss the primacy bias in deep RL, a tendency to excessively train on early data damaging further learning progress, and propose a strategy of periodically resetting a section of the network to address this issue, while utilizing the replay buffer as a means of transferring knowledge. Earlier, Igl et al. (2020) had previously observed that deep RL agents might lose the ability to generalize due to non-stationarity and proposed to use distillation as a mitigation mechanism.

Balancing stability and plasticity is a challenge in DRL. Too much plasticity may result in the agent quickly forgetting previously learned knowledge, while too much stability may hinder the agent's ability to adapt to new situations or learn new strategies. Achieving the right balance often requires careful design choices.

1.2.6 The Arcade Learning Environment

The introduction of the Arcade Learning Environment (ALE) by Bellemare et al. (2013) marked the inception of a challenging suite for reinforcement learning, in which agents learn to play Atari games using visual inputs. This task is different from the conventional domains commonly explored in reinforcement learning in that nonlinear function approximation (e.g., neural networks) are key. Atari games are discrete control tasks, in which agents choose between up to 18 available actions at each step. To further the progress in the field, Machado et al. (2017) present some methodological best practices and a new version of the Arcade Learning Environment that supports stochasticity and multiple game modes. These modes were developed by game designers to make each game progressively more challenging for human players.

1.2.7 Evaluation in Atari

Performance on Atari is typically calculated as the human-normalized score, calculated separately on each game as $\frac{\text{agent\_score}-\text{random\_score}}{\text{human\_score}-\text{random\_score}}$. Bellemare et al. (2013) introduce *human_score* and *random_score* values with some simple metrics that help compare agents across a diverse set of domains. In contrast to numerous continuous control domains like DM Control, where algorithm hyperparameters are frequently tailored for individual tasks, the standard practice in deep reinforcement learning for Atari games has since Bellemare et al. (2013) been to use a uniform hyperparameter setting across all games. This practice significantly influences algorithmic design; by forcing methods to be successful on a wide range of games, approaches requiring finely-tuned hyperparameters are relatively disadvantaged, and encourages the development of methods that organically tune hyperparameters during training, such as Agent57 (Badia et al., 2020). Consequently, methods that deviate from this standard, like Sunrise (Lee et al., 2020), cannot be directly compared to those adhering to it.

Recently, Agarwal et al. (2021) introduce three tools for improving the quality of experimental reporting in the few-run regime, all aligned with the principle of accounting for statistical uncertainty in results. The tools are *Stratified Bootstrap Confidence Intervals, Performance Profiles*, and, *Robust and Efficient Aggregate Metrics*. For more rigorous and statistically meaningful analyses, we follow the guidelines suggested by them. Specifically, we normalize all runs by the human scores for each game, aggregate them, and report metrics with 95% stratified bootstrap CIs and sample efficiency curves.

1.2.8 Data Efficiency and Representations

Data efficiency is a major challenge in deep reinforcement learning. Despite the achievements of recent algorithms in solving complex tasks like DotA 2 (OpenAI et al., 2019), Starcraft 2 (Vinyals et al., 2019), and Atari (Badia et al., 2020), their success is often reliant on substantial amounts of experience. Consequently, there has been an increasing emphasis on enhancing the data efficiency of reinforcement learning, generally defined as improving performance with limited environment interaction time. Several methods have been proposed to tackle this issue, from *model-based* methods that aim to accelerate learning by learning an explicit model of environment dynamics and reward distributions (e.g., Kaiser et al., 2019) to modified versions of existing algorithms that were previously optimized for performance in large-data regimes (for example, Data-Efficient Rainbow from Van Hasselt et al., 2019).

Representation learning is an integral part of reinforcement learning algorithms, and the success of reinforcement learning algorithms in large-scale, complex tasks hinges on their ability to construct valuable representations of the environment with which the algorithms interact. Thus, improving the representations learned by the neural networks could improve data efficiency in Deep Reinforcement learning.

Feature selection and feature learning has long been an important subdomain of RL, and with the advent of deep reinforcement learning there has been much recent interest in comprehending and enhancing the representations acquired by RL agents. Much of the work in representation learning has taken place from the perspective of auxiliary tasks like self-supervised, contrastive or bisimulation metrics losses (Jaderberg et al., 2016; Anand et al., 2020; Mazoure et al., 2020; Castro et al., 2021)]; in addition to the primary reinforcement learning task, the agent may attempt to predict and control additional aspects of the environment. Auxiliary tasks shape the agent's representation of the environment typically via gradient descent on the additional learning objectives.

The idea behind using auxiliary tasks is to encourage the agent to learn useful representations or acquire specific skills that help the agent learn faster or improve its overall performance by providing additional learning signals.

Sample-Efficient RL on ALE: Sample efficiency has always been an import aspect of evaluation in RL, as it can often be expensive to interact with an environment. Kaiser et al. (2020) introduced the Atari 100K benchmark, which has proven to be useful for evaluating sample-efficiency, and has led to a number of recent advances.

Kostrikov et al. (2020) use data augmentation to design a sample-efficient RL method, DrQ, which outperformed prior methods on Atari 100K. Data-Efficient Rainbow (DER) (Van Hasselt et al., 2019) and DrQ(ϵ) (Agarwal et al., 2021) simply modified the hyperparameters of existing model-free algorithms to exceed the performance of existing methods without any algorithmic innovation.

Schwarzer et al. (2021) introduced SPR, which builds on Rainbow (Hessel et al., 2018) and uses a self-supervised temporal consistency loss based on BYOL (Grill et al., 2020) combined with data augmentation. SR-SPR (Schwarzer et al., 2021) combines SPR with periodic network resets to achieve state-of-the-art performance on the 100K benchmark. Ye et al. (2021) used a self-supervised consistency loss similar to SPR (Chen and He, 2021).

EfficientZero (Ye et al., 2021), an efficient variant of MuZero (Schrittwieser et al., 2020), learns a discrete-action latent dynamics model from environment interactions, and selects actions via lookahead MCTS in the latent space of the model. Micheli et al. (2023) introduce IRIS, a data-efficient agent that learns in a world model composed of an autoencoder and an auto-regressive Transformer.

Recently, Schwarzer, Obando-Ceron, et al. (2023) introduced a value-based RL agent, BBF (Bigger, Better, Faster), that achieves super-human performance in the Atari 100K benchmark. BBF relies on scaling the neural networks used for value estimation, as well as a number of other design choices that enable this scaling in a sample-efficient manner.

1.3 Offline deep Reinforcement Learning

Offline RL focuses on the challenge of developing a policy using a static dataset of trajectories, without any additional interactions with the environment. This setting is particularly valuable for leveraging extensive historical interaction data in real-world decision-making domains like robotics, recommendation systems, and healthcare. (Shortreed et al., 2011).

Offline reinforcement learning (RL) refers to a scenario where the algorithm operates without direct access to the MDP, instead relying on a pre-existing dataset of transitions $D = (s, a, s', r(s, a))$. The (unknown) policy that generated this data is referred to as a behavior policy π_B. Successful offline RL methods must handle distributional shifts, as well as data collected via processes that may not be representable by the chosen policy class. The challenges associated with offline RL are extensively examined and discussed by Levine et al. (2020).

Despite the offline RL benefits such as sample efficiency, data reusability or risk reduction, offline RL presents unique challenges. Offline RL algorithms typically face challenges such as distributional shift, where the data distribution in the dataset may differ from the distribution the agent would encounter during online interaction. This can lead to poor performance or even catastrophic divergence when standard RL algorithms are directly applied to offline data. Mitigating distributional shift and ensuring effective policy improvement are active areas of research in offline RL.

Researchers are developing various algorithms and techniques, such as Conservative Q-Learning (CQL) (Kumar et al., 2020), Behavior Regularized Offline Reinforcement Learning (BRAC) (Wu et al., 2019), DR3 (Kumar et al., 2021b) and others, to address the challenges and improve the stability and performance on a wide range of offline RL problems.

Although there is a large literature for offline RL algorithms, CQL still performs very well on diverse domains and it is widely used as a backbone for building new state of the art offline reinforcement learning algorithms. For instance, recently Kumar et al. (2022) demonstrate that with careful design decisions, offline Q-learning (CQL) can scale to high capacity models trained on large, diverse datasets from many tasks, leading to policies that not only generalize broadly, but also learn representations that effectively transfer to new downstream tasks and exceed the performance in the training dataset. The authors combine CQL (Kumar et al., 2020) with C51 (Bellemare et al., 2017) which leads to a drastic improvement in performance when using large neural networks architectures.

2 Small batch deep reinforcement learning

Authors: Johan Obando-Ceron, Marc G. Bellemare, and Pablo Samuel Castro.

This chapter presents a lengthened version of a joint work with Marc G. Bellemare and Pablo Samuel Castro. It was accepted to the conference track of the Thirty-seventh Conference on Neural Information Processing Systems 2023 (Ceron et al., 2023).

Contributions: I formulated the project, wrote code for and performed all of the experiments for Atari listed in the paper. Marc Bellemare and Pablo Samuel Castro provided advising and helped refine the paper.

Affiliation

— Johan Obando-Ceron, Mila, University of Montreal

— Marc G. Bellemare, Mila, University of Montreal

— Pablo Samuel Castro, Google Deepmind, University of Montreal

2.1 Introduction

One of the central concerns for deep reinforcement learning (RL) is how to efficiently make the most use of the collected data for policy improvement. This is particularly important in online settings, where RL agents learn while interacting with an environment, as interactions can be expensive. Since the introduction of DQN (Mnih et al., 2015), one of the core components of most modern deep RL algorithms is the use of a finite *replay memory* where experienced transitions are stored. During learning, the agent samples mini-batches from this memory to update its network parameters.

Since the policy used to collect transitions is changing throughout learning, the replay memory contains data coming from a mixture of policies (that differ from the agent's current policy), and results in what is known as *off-policy* learning. In contrast with training data for supervised learning problems, online RL data is highly *non-stationary.* Still, at any point during training the replay memory exhibits a distribution over transitions, which the agent samples from at each learning step. The number of sampled transitions at each learning step is known as the *batch size*, and is meant to produce an unbiased estimator of the underlying data distribution. Thus, in theory, larger batch sizes should be more accurate representations of the true distribution.

Some in the supervised learning community suggest that learning with large batch sizes leads to better optimization (Shallue et al., 2019), since smaller batches yield noisier gradient estimations. Contrastingly, others have observed that larger batch sizes tend to converge to "sharper" optimization landscapes, which can result in worsened generalization (Keskar et al., 2017); smaller batches, on the other hand, seem to result in "flatter" landscapes, resulting in better generalization.

Learning dynamics in deep RL are drastically different than those observed in supervised learning, in large part due to the data non-stationarity mentioned above. Given that the choice of batch size will have a direct influence on the agent's sample efficiency and ultimate performance, developing a better understanding of its impact is critical. Surprisingly, to the best of our knowledge there have been no studies exploring the impact of the choice of batch size in deep RL. Most recent works have focused on related questions, such as the number of gradient updates per environment step (Nikishin et al., 2022; D'Oro et al., 2023; Sokar et al., 2023), but have kept the batch size fixed.

In this work we conduct a broad empirical study of batch size in online value-based deep reinforcement learning. We uncover the surprising finding that *reducing* the batch size seems to provide substantial performance benefits and computational savings. We showcase this finding in a variety of agents and training regimes (subsection 2.2.1), and conduct in-depth analyses of the possible causes (subsection 2.2.2). The impact of our findings and analyses go beyond the choice of the batch size hyper-parameter, and help us develop a better understanding of the learning dynamics in online deep RL.

2.2 Experimental results

2.2.1 The small batch effect on agent performance

In this section we showcase the performance gains that arise when training with smaller batch sizes. We do so first with four standard value-based agents (§2.2.1.1), with varying architectures (§2.2.1.2), agents optimized for sample efficiency (§2.2.1.3), and with extended training (§2.2.1.4). Additionally, we explore the impact of reduced batch sizes on exploration (§2.2.1.5) and computational cost (§2.2.1.6).

Experimental setup: We use the Jax implementations of RL agents, with their default hyper-parameter values, provided by the Dopamine library (Castro et al., 2018) [1] and applied to the Arcade Learning Environment (ALE) (Bellemare et al., 2013). [2] It is worth noting that the default batch size is 32, which we indicate with a **black** color in all the plots below, for clarity. We evaluate our agents on 20 games chosen by Fedus et al. (2020) for their analysis of replay ratios, picked to offer a diversity of difficulty and dynamics. To reduce the computational burden, we ran most of our experiments for 100 million frames (as opposed to the standard 200 million). For evaluation, we follow the guidelines of Agarwal et al. (2021).

1. Dopamine code available at https://github.com/google/dopamine.
2. Dopamine uses sticky actions by default (Machado et al., 2017).

Specifically, we run 3 independent seeds for each experiment and report the human-normalized *interquantile mean (IQM)*, aggregated over the 20 games, configurations, and seeds, with the 95% stratified bootstrap confidence intervals. Note that this means that for most of the aggregate results presented here, we are reporting mean and confidence intervals over 60 independent seeds. All experiments were run on NVIDIA Tesla P100 GPUs.

2.2.1.1 Standard agents

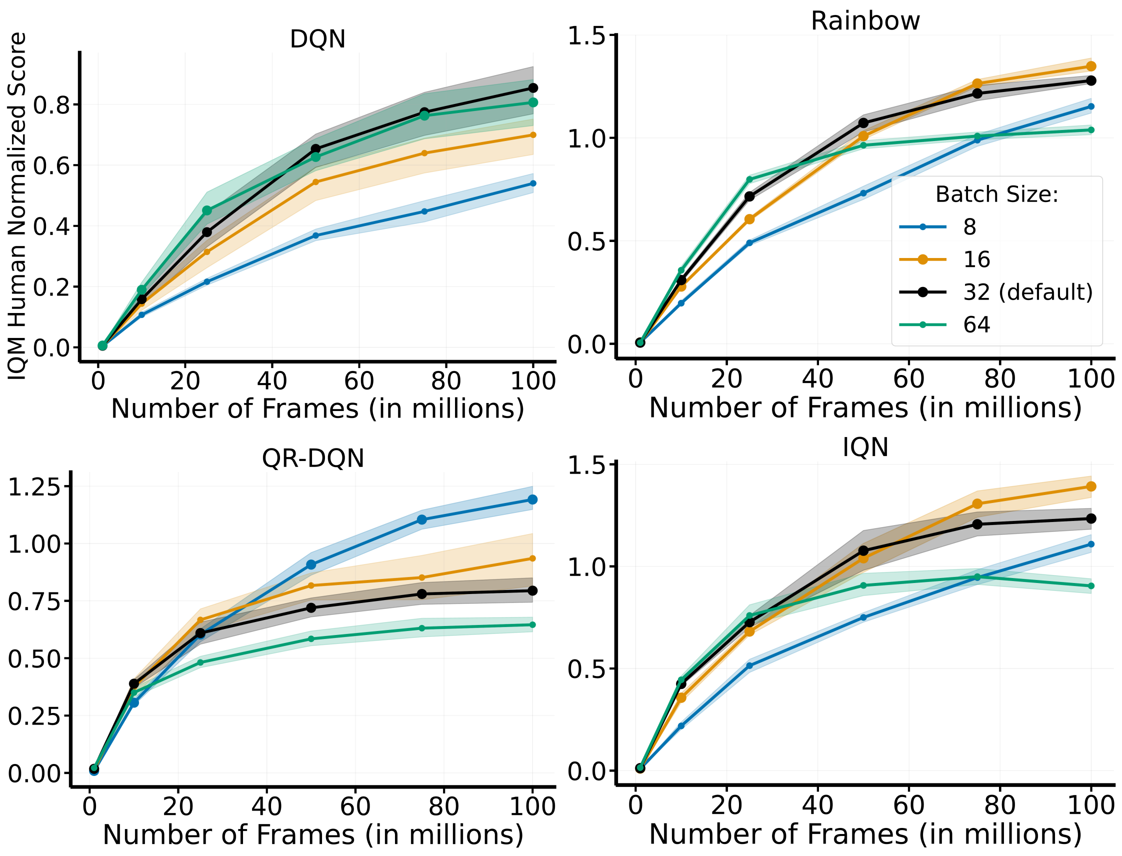

Figure 2.1 – IQM for human normalized scores for DQN, Rainbow, QR-DQN, and IQN. All games run with 3 independent seeds, shaded areas representing 95% confidence intervals.

We begin by investigating the impact reducing the batch size can have on four popular value-based agents, which were initially benchmarked on the ALE suite: DQN (Mnih et al., 2015), Rainbow (Hessel et al., 2018) (Note that Dopamine uses a "compact" version of the original Rainbow agent, including only multi-step updates, prioritized replay, and C51), QR-DQN (Dabney et al., 2018a), and IQN (Dabney et al., 2018b). In Figure 2.1 we can observe that, in general, reduced batch size results in improved performance. The notable exception is DQN, for which we provide an analysis and explanation for why this is the case below. To verify that our results are not a consequence of the set of 20 games used in our analyses, we ran QR-DQN (where the effect is most observed) over the full 60 games in the suite and report the results in Figure 2.2. Remarkably, a batch size of 8 results in significant gains on 38 out of the full 60 games, for an average performance improvement of 98.25%. We provide complete results for all games using QR-DQN agent in Figure 2.3. In Figure 2.4, we can observe that using a smaller batch size value yields to a 37% of improving *Aggregate Interquantile Mean* over all 60 Atari 2600 games using QR-DQN (Dabney et al., 2018a).

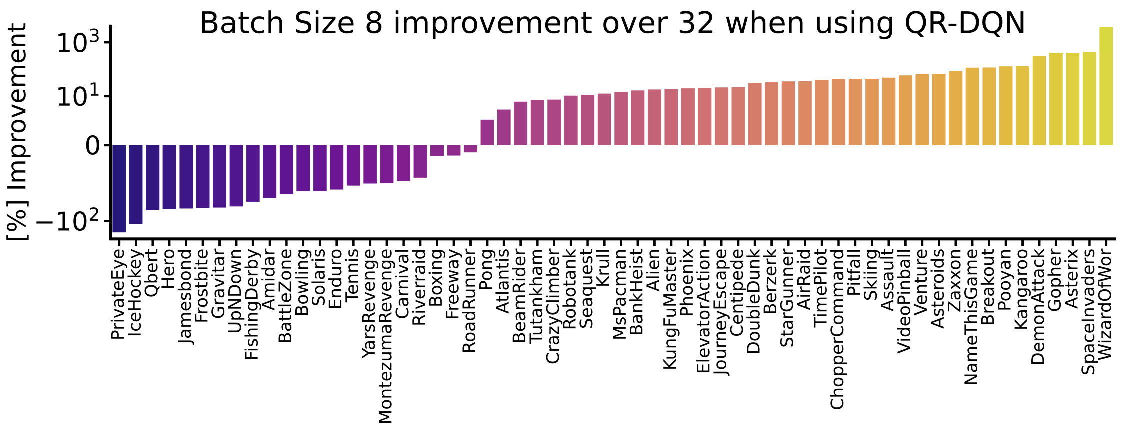

Figure 2.2 – Evaluating QR-DQN (Dabney et al., 2018a) with varying batch sizes over all 60 Atari 2600 games. Average improvement obtained when using a batch size of 8 over 32 (default); All games run for 3 seeds, with shaded areas displaying 95% stratified bootstrap confidence intervals.

2.2.1.2 Varying architectures

Deep neural networks have proven to be effective in extracting relevant features from data for a variety of downstream tasks. Recently, researchers have become interested in understanding the impact of scaling neural network architectures. It has been observed that increasing the size of models often leads to significant performance improvements in applications such as language modeling and computer vision.

Building upon these promising findings, the deep reinforcement learning community has started exploring the effects of scaling the model size in function approximation. By increasing the capacity of the neural network used to approximate the value or policy functions in RL algorithms, researchers aim to enhance the agent's learning and decision-making capabilities.

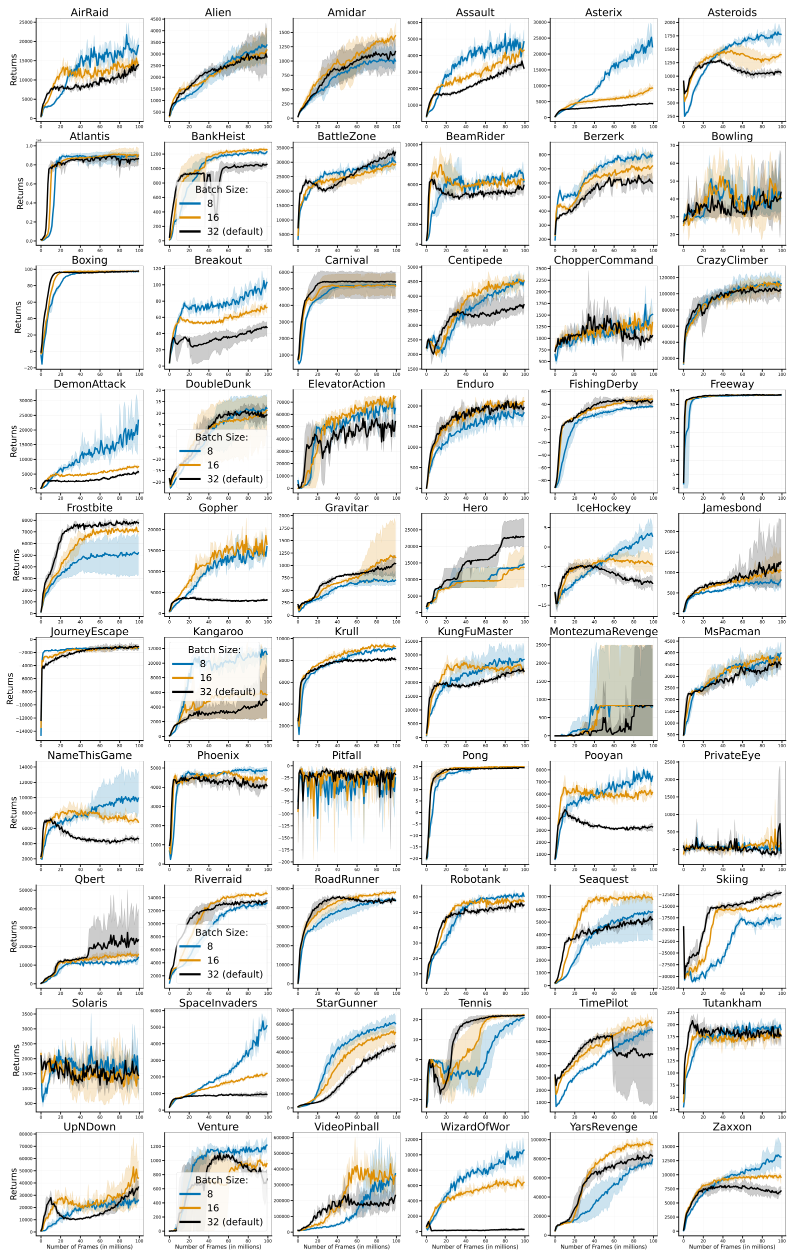

Figure 2.3 – Training curves for QR-DQN agent. The results for all games are over 3 independent runs.

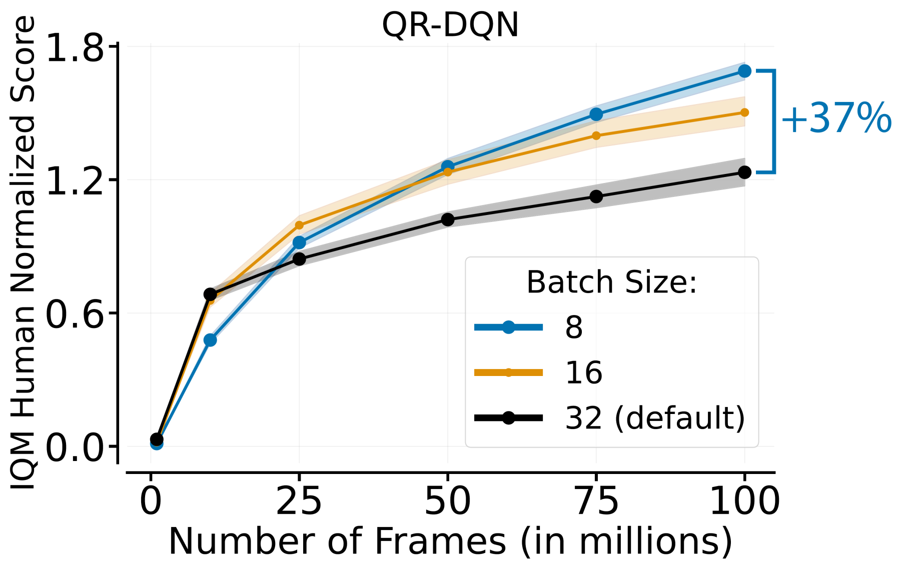

Figure 2.4 – Aggregate Interquantile Mean (Agarwal et al., 2021) of human normalized scores over all 60 Atari 2600 games using QR-DQN (Dabney et al., 2018a). All games run for 3 seeds, with shaded areas displaying 95% stratified bootstrap confidence intervals.

Although the CNN architecture originally introduced by DQN (Mnih et al., 2015) has been the backbone for most deep RL networks, there have been some recent works exploring the effects of varying architectures (Espeholt et al., 2018; Agarwal et al., 2022; Sokar et al., 2023; Schwarzer et al., 2023). We investigate the small batch effect by varying the QR-DQN architecture in two ways: **(1)** expanding the convolutional widths by 4 times (resulting in a substantial increase in the number of parameters), and **(2)** using the Resnet architecture proposed by Espeholt et al. (2018) (which results in a similar number of parameters to the original CNN architecture, but is a deeper network). In Figure 2.5 we can observe that not only do reduced batch sizes yield improved performance, but they are better able to leverage the increased number of parameters (CNNx4) and the increased depth (Resnet). We include the learning curves for each game when varying batch size and using CNNx4 (see Figure 2.6) or Resnet (see Figure 2.7) architecture.

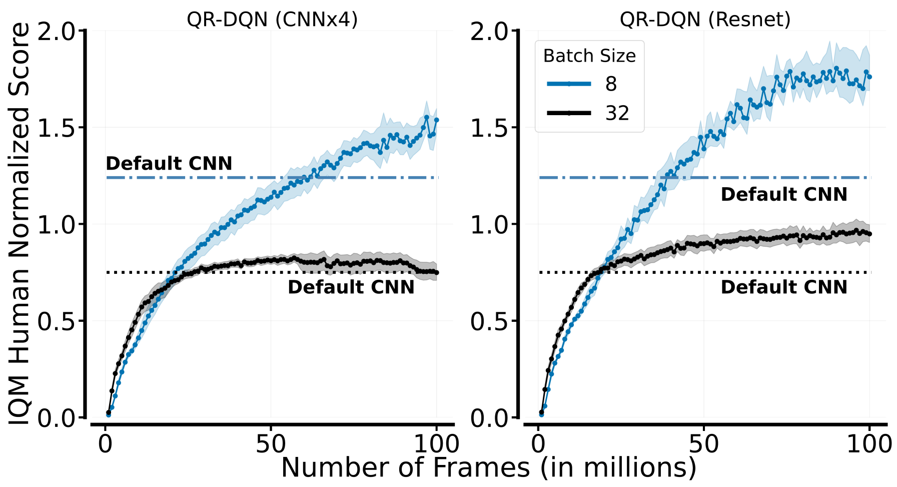

Figure 2.5 – IQM for human normalized scores with varying neural network architectures over 20 games, with 3 seeds per experiment. Shaded areas represent 95% stratified bootstrap confidence intervals.

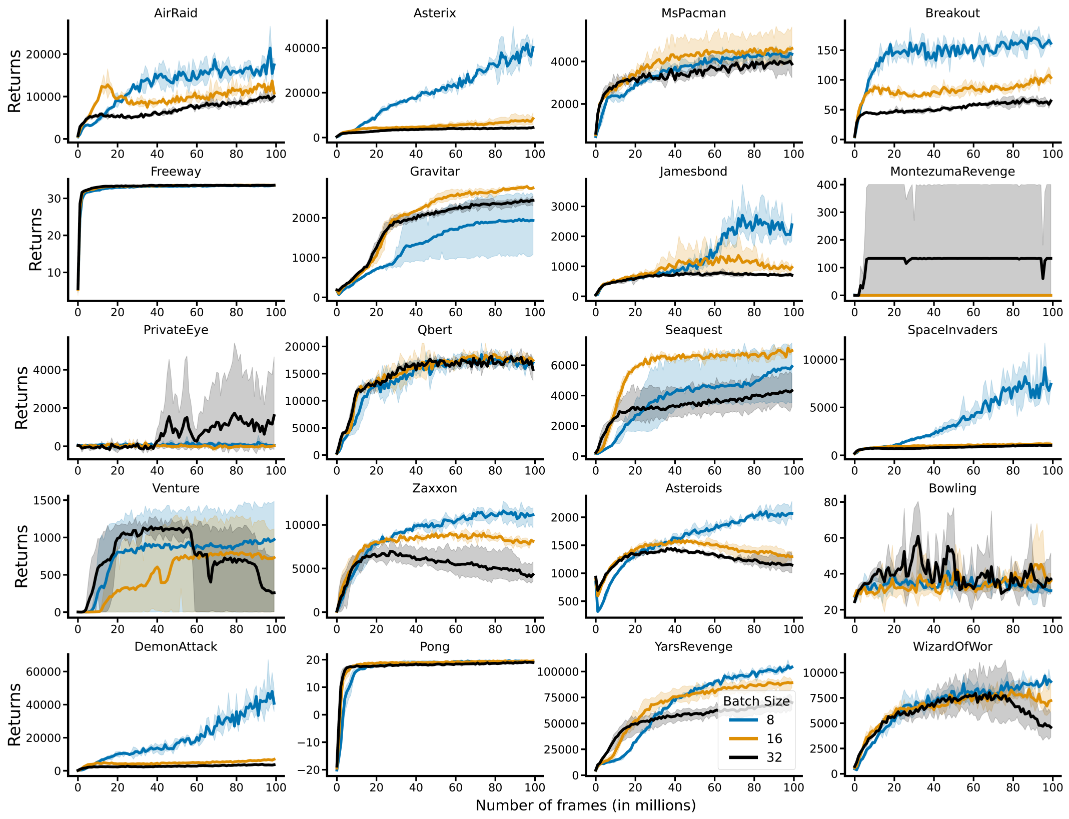

Figure 2.6 – Evaluating the effect of CNNx4 to QR-DQN, learning curves for all games.

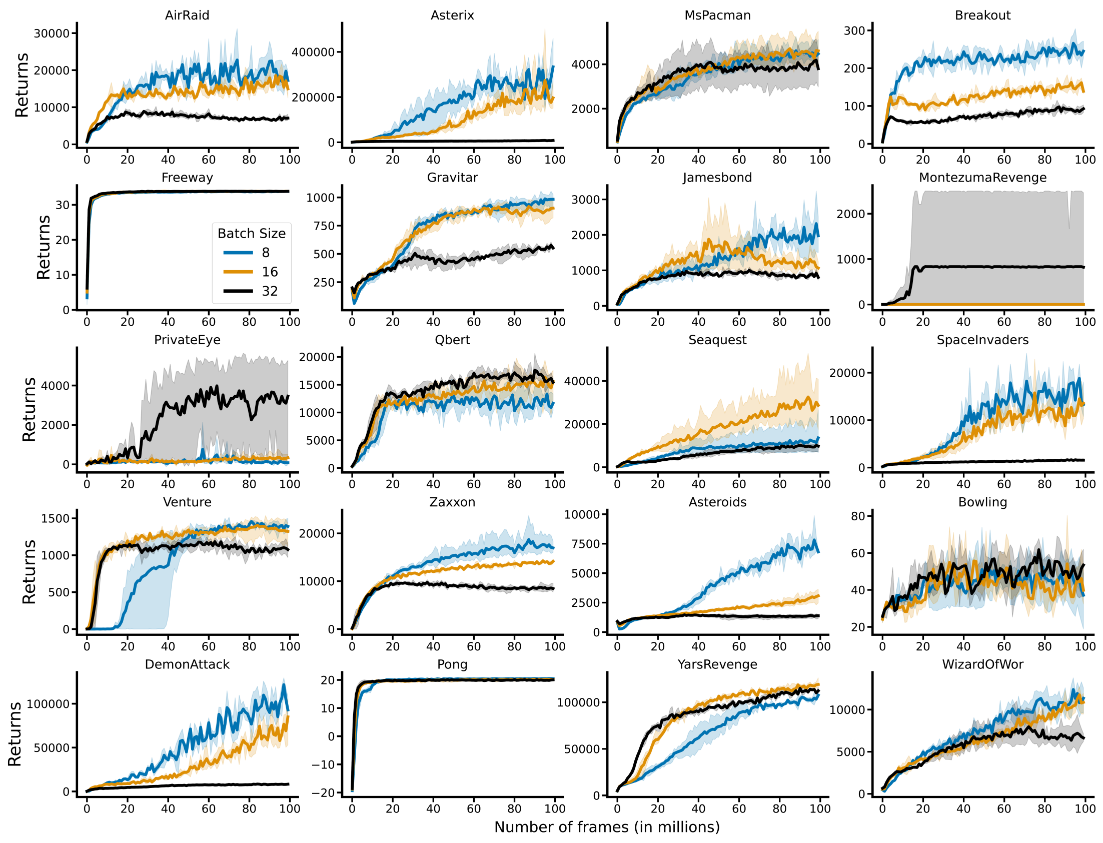

Figure 2.7 – Evaluating the effect of Resnet to QR-DQN, learning curves for all games.

2.2.1.3 Atari 100k agents

There has been an increased interest in evaluating Atari agents on very few environment interactions, for which Kaiser et al. (2020) proposed the 100k benchmark [3]. We evaluate the effect of reduced batch size on three of the most widely used agents for this regime: Data-efficient Rainbow (DER), a version of the Rainbow algorithm with hyper-parameters tuned for faster early learning (van Hasselt et al., 2019); DrQ(ϵ), which is a variant of DQN that uses data augmentation (Agarwal et al., 2021); and SPR, which incorporates self-supervised learning to improve sample efficiency (Schwarzer et al., 2020). For this evaluation we evaluate on the standard 26 games for this benchmark (Kaiser et al., 2020), aggregated over 6 independent trials.

3. Here, 100k refers to agent steps, or 400k environment frames, due to skipping frames in the standard training setup.

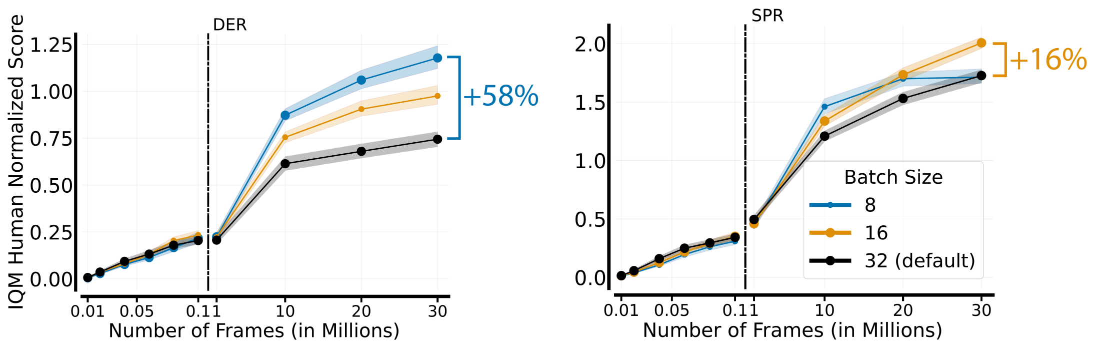

Figure 2.8 – Measured IQM of human-normalized scores on the 26 100k benchmark games, with varying batch sizes, of DER and SPR. We evaluate performance at 100k agent steps (or 400k environment frames), and at 30 million environment frames, run with 6 independent seeds for each experiment, and shaded areas display 95% confidence intervals.

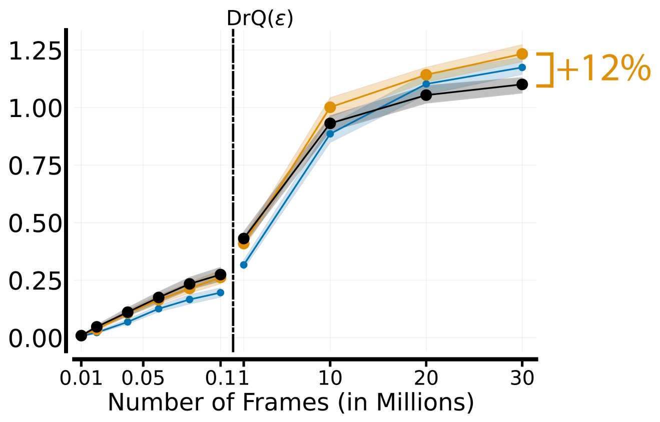

Figure 2.9 – Measured IQM of human-normalized scores on the 26 100k benchmark games, with varying batch sizes, of DrQ(ϵ). We evaluate performance at 100k agent steps (or 400k environment frames), and at 30 million environment frames, run with 6 independent seeds for each experiment, and shaded areas display 95% confidence intervals.

In Figure 2.8 and Figure 2.9, we include results both at the 100k benchmark (left side of plots), and when trained for 30 million frames. Our intent is to evaluate the batch size effect on agents that were optimized for a different training regime. We can see that although there is little difference in 100k, there is a much more pronounced effect when trained for longer. This finding suggests that reduced batch sizes enables continued performance improvements when trained for longer.

2.2.1.4 Training Stability

In deep reinforcement learning, it is not uncommon for the performance to plateau or reach a point where there is minimal improvement despite continued training. Several factors can saturate the agent's learning process as overfitting, limited exploration or insufficient training data. Overcoming the performance plateau in deep RL can be challenging, and it often requires a combination of exploration strategies, architectural adjustments, and careful experimentation.

To further investigate whether reduced batch sizes enables continual improvements with longer training, we extend the training of QR-DQN up to the standard 200 million frames. In Figure 2.10 we can see that training performance tends to plateau for the higher batch sizes. In contrast, the smaller batch sizes seem to be able to continuously improve their performance.

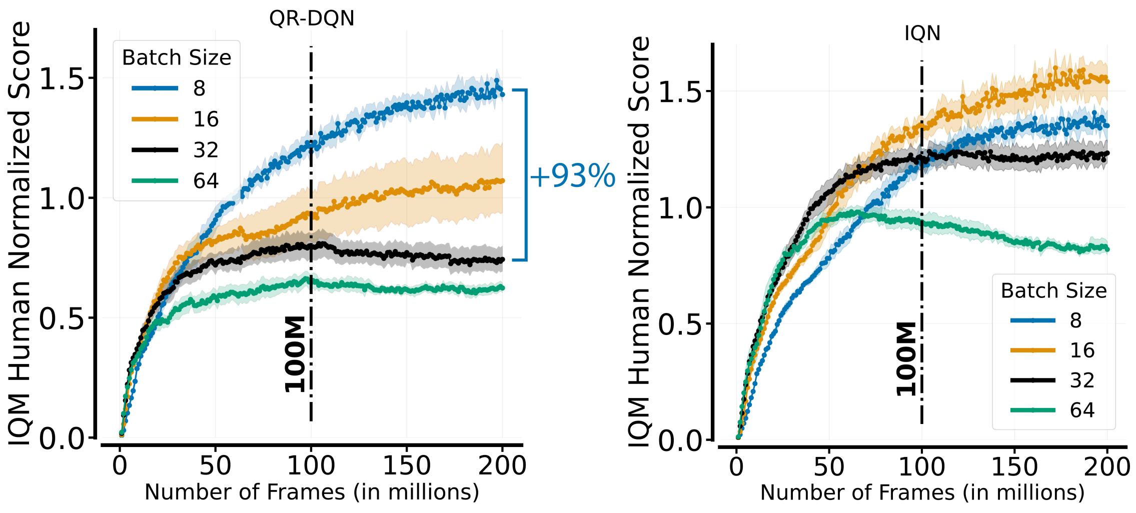

Figure 2.10 – Measuring IQM for human-normalized scores when training for 200 million frames using **Left:** QR-DQN (Dabney et al., 2018a) and **Right:** IQN (Dabney et al., 2018b). Results aggregated over 20 games, where each experiment was run with 3 independent seeds and we report 95% confidence intervals.

We include the learning curves for each game when varying batch size and using QR-DQN (see Figure 2.11) or IQN (see Figure 2.12) algorithm. We report the scores when training for 200 million frames.

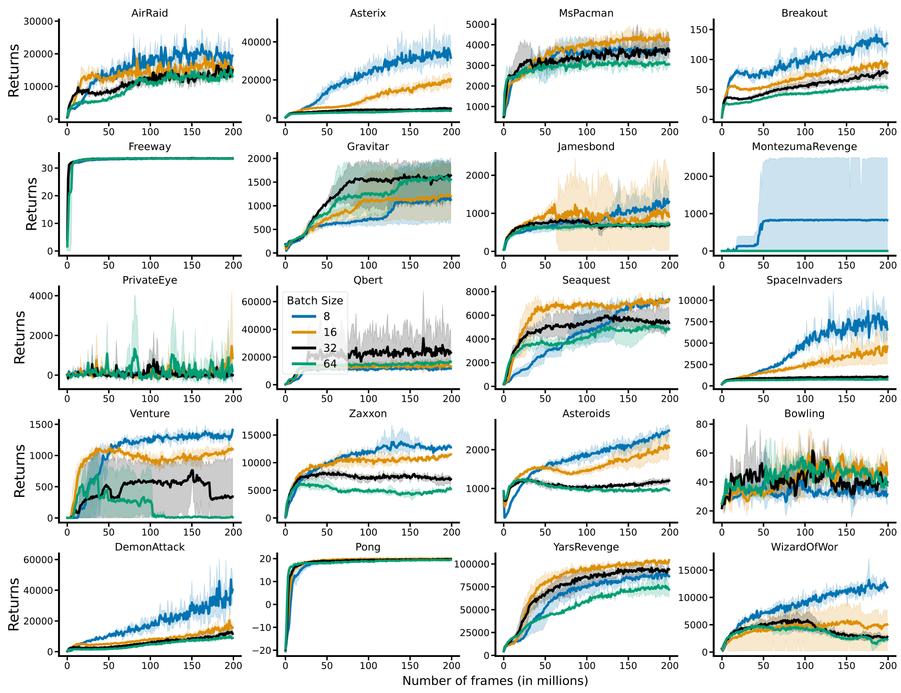

Figure 2.11 – Learning curves for individual games, when trained for 200 million frames using QR-DQN (Dabney et al., 2018a). Results aggregated over 3 seeds, reporting 95% confidence intervals.

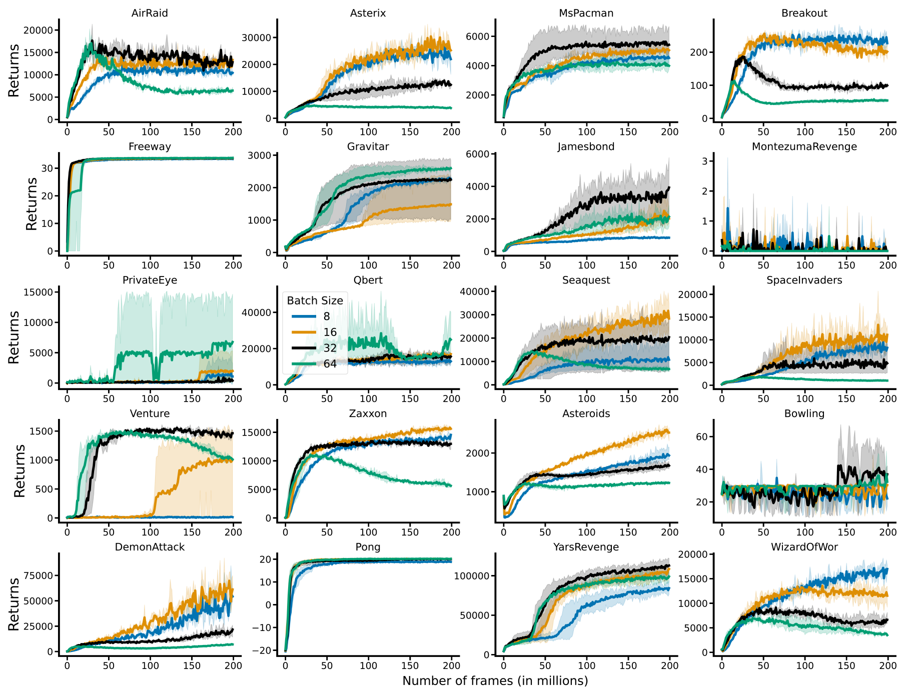

Figure 2.12 – Learning curves for individual games, when trained for 200 million frames using IQN (Dabney et al., 2018b). Results aggregated over 3 seeds, reporting 95% confidence intervals.

2.2.1.5 Impact on exploration

Exploration is a fundamental aspect of deep reinforcement learning that involves the agent actively seeking out and gathering information about the environment in order to discover new and potentially better policies. Effective exploration strategies are crucial for learning optimal or near-optimal policies in complex RL tasks. Without adequate exploration, the agent may get stuck in suboptimal policies and fail to discover better solutions.

The simplest and most widely used approach for exploration is to select actions randomly with a probability ϵ, as opposed to selecting them greedily from the current Q_θ estimate. The increased variance resulting from reduced batch sizes (as we will explore in more depth below) may also result in a natural form of exploration. To investigate this, we set the target ϵ value to 0.0 for QR-DQN [4]. In Figure 2.13 we compare performance across four known hard exploration games (Bellemare et al., 2016; Taiga et al., 2020) and observe that reduced batch sizes tends to result in improved performance for these games.

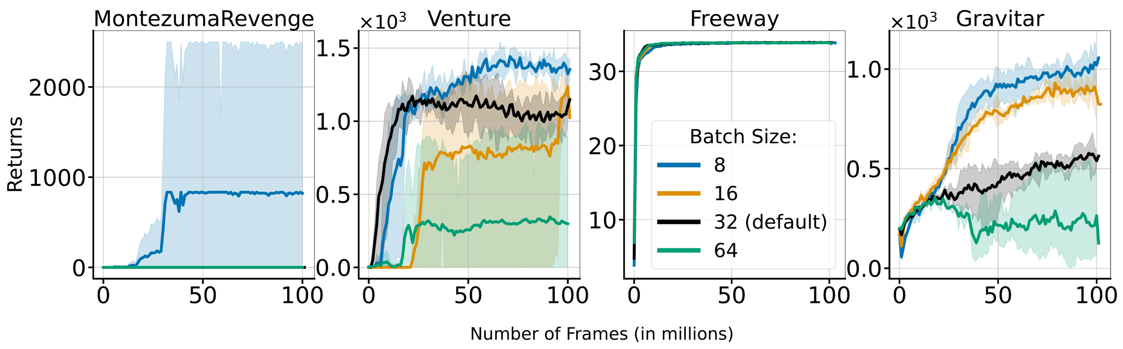

Figure 2.13 – Performance of QR-DQN on four hard exploration games with a target ϵ value of 0.0, and with varying batch sizes.

We include the IQM of human-normalized scores for the 20 games when varying batch size and using $\epsilon = 0$ (see Figure 2.14). We report the scores when training for 100 million frames.

4. Note that we follow the training schedule of Mnih et al. (2015) where the ϵ value begins at 1.0 and is linearly decayed to its target value over the first million environment frames.

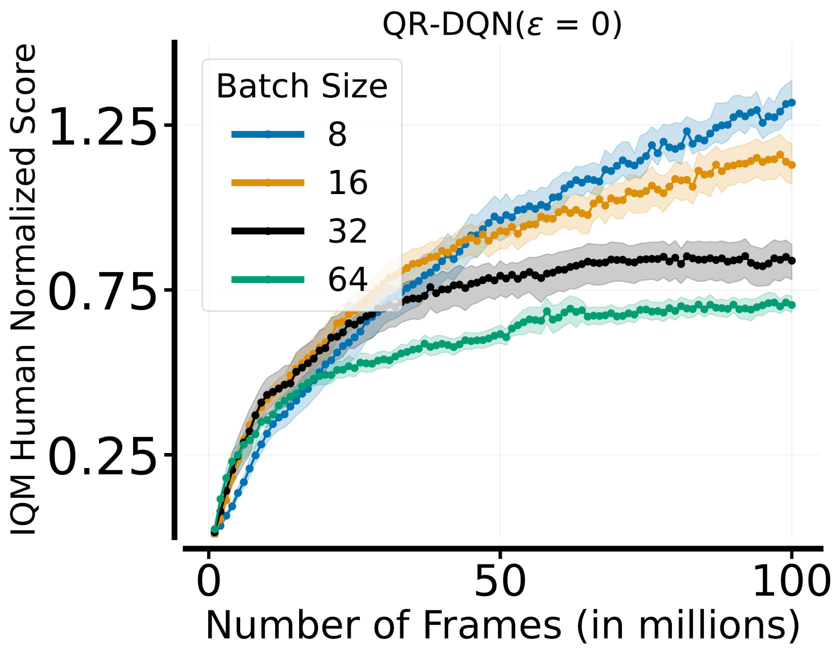

Figure 2.14 – Aggregate IQM of human-normalized scores over 20 games with a target ϵ value of 0.0. In all the plots 3 independent seeds were used for each game/batch-size configuration, with shaded areas representing 95% confidence intervals.

2.2.1.6 Computational impact

In the last few years, deep learning has made remarkable advancements across a wide range of applications. Notable examples include protein structure prediction (Jumper et al., 2021, AlphaFold), text-to-image synthesis (Ramesh et al., 2021, DALL-E), text generation (Brown et al., 2020, GPT-3), deep RL algorithms for solving complex games (Dota2, Berner et al., 2019) and (GATO, Reed et al., 2022). These achievements have been primarily driven by the strategy of scaling up deep learning models to extremely large sizes and training them on massive amounts of data.

While scaling of deep learning models has paved the way for significant progress, training large models has become extremely expensive. For example, GPT-3 training was estimated to cost $1.65 million with Google v3 TPUs (Lohn and Musser, 2022) and inefficient/naive development of a transformer model would emit carbon dioxide (CO2) equivalent to the lifetime carbon footprint of five cars (Strubell et al., 2019). Concerningly, deep learning has still not reached the performance level required by many of its applications. For example, deploying fully autonomous vehicles in the real world demands human-level performance, which has not been reached yet. Increasing the sizes of models and datasets to achieve such required performance will render current training strategies financially, environmentally, and otherwise unsustainable.

Considering the unsuitable increase in computational requirements, progress with deep learning demands more compute-efficient training methods. A natural direction is to eliminate algorithmic inefficiencies in the learning process, aiming to reduce the time, cost, energy consumption, and carbon footprint associated with training deep learning models (Bartoldson et al., 2023).

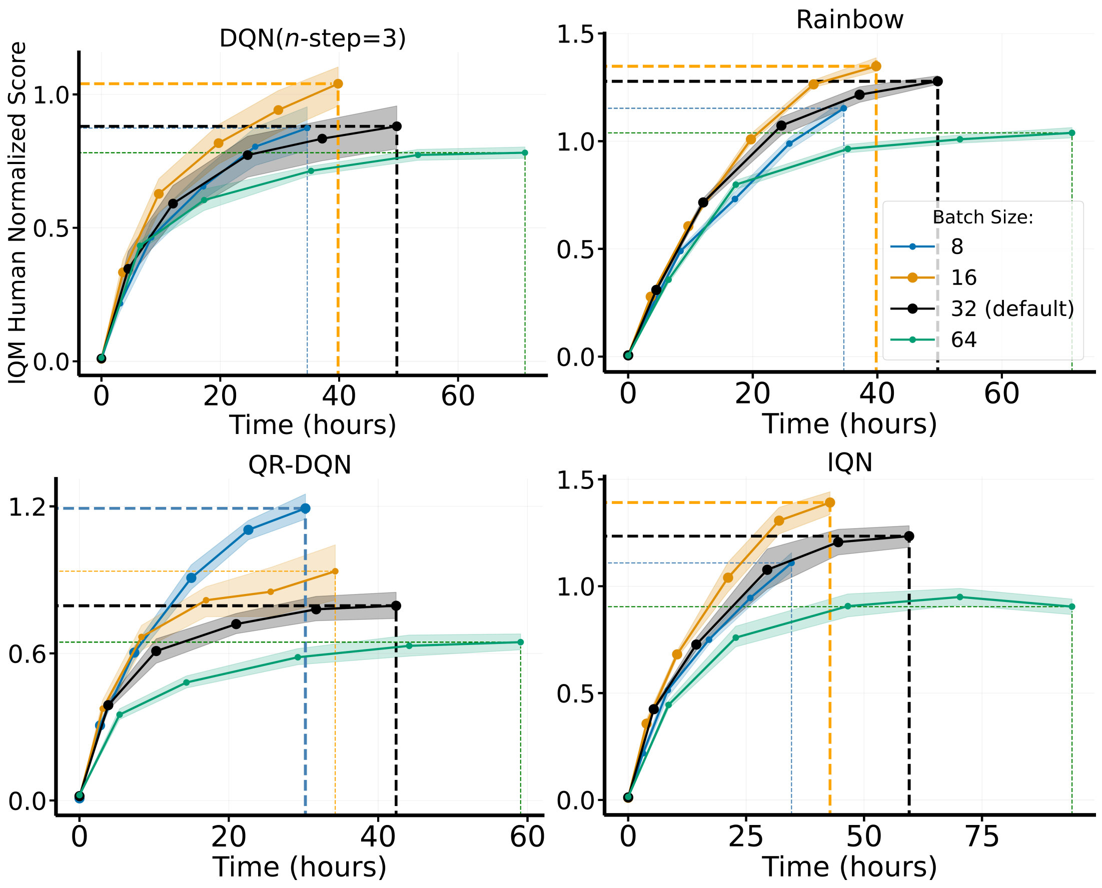

Figure 2.15 – Measuring wall-time versus IQM of human-normalized scores when varying batch sizes in DQN (with n-step set to 3), Rainbow, QR-DQN, and IQN. Each experiment had 3 independent runs, and the confidence intervals show 95% confidence intervals.

Empirical advances in deep reinforcement learning are generally measured with respect to sample efficiency; that is, the number of environment interactions required before achieving a certain level of performance. It fails to capture computational differences between algorithms. If two algorithms have the same performance with respect to environment interactions, but one takes twice as long to perform each training step, one would clearly opt for the faster of the two. This important distinction, however, is largely overlooked in the standard evaluation methodologies used by the DRL community.

We have already demonstrated the performance benefits obtained when reducing batch size, but an additional important consequence is the reduction in computation wall-time. Figure 2.15 demonstrates that not only can we obtain better performance with a reduced batch size, but we can do so at a fraction of the runtime. As a concrete example, when changing the batch size of QR-DQN from the default value of 32 to 8, we achieve both a 50% performance increase and a 29% speedup in wall-time.

It may seem surprising that smaller batch sizes have a faster runtime, since larger batches presumably make better use of GPU parallelism. However, as pointed out by Masters and Luschi (2018), the speedups may be a result of a smaller memory footprint, enabling better machine throughput.

Key observations on reduced batch sizes:

— They generally improve performance, as evaluated across a variety of agents and network architectures.

— When trained for longer, the performance gains continue, rather than plateauing.

— They seem to have a beneficial effect on exploration.

— They result in faster training, as measured by wall-time.

Takeaway: The impact of reducing batch size on performance enhancement is consistently evident when assessed across various agents and network architectures. Interestingly, extending the training duration reveals a continual progression of performance gains, in contrast to the typical plateau observed in other scenarios. Notably, smaller batch size value influence exploration, fostering a more thorough and effective learning process.

Many methods have been proposed to address the exploitation-exploration dilemma, and some techniques emphasize exploration by adding noise directly to the parameter space of agents (Fortunato et al., 2018; Hao et al., 2023; Plappert et al., 2017; Gupta et al., 2018) which inherently adds variance to the learning process. Noise perturbation is another approach that has been taken to induce exploration (Eberhard et al., 2022).

Like these works, our analyses show that increasing variance by reducing the batch size may result in similar beneficial exploratory effects, as the mentioned works suggest. It is difficult to isolate the direct impact on exploration; however, the improved performance observed on all the hard exploration games in Atari suggests that improved exploration may be an advantageous consequence of the variance induced by reduced batch size. We believe further work exploring the impact of variance injection in deep RL algorithms is necessary.

Beyond enhanced performance, another notable advantage is the reduction in training time, evident through faster convergence as measured by wall-time. When focusing on the aspect of reducing batch size, the benefits become apparent in terms of more efficient wall-time during training. While lowering the batch size might seem counterintuitive at first, it often leads to enhanced training speed due to the reduction in computation and memory requirements. Overall, these experiments highlight the positive impact of smaller batch when optimizing compute efficiency and agent's performance.

2.2.2 Understanding the small batch effect

Having demonstrated the performance benefits arising from a reduced batch size across a wide range of tasks, in this section we seek to gain some insight into possible causes. We will focus on QR-DQN, as this is the agent where the small batch effect is most pronounced (Figure 2.1). We begin by investigating possible confounding factors for the small batch effect, and then provide analyses on the effect of reduced batch sizes on network dynamics.

2.2.2.1 Relation to other hyperparameters

Learning rates The learning rate is a crucial hyperparameter in deep reinforcement learning that determines the step size at which the parameters of the neural network are updated during training. It plays a significant role in the convergence and stability of the learning process. For instance the Adam optimizer combines the

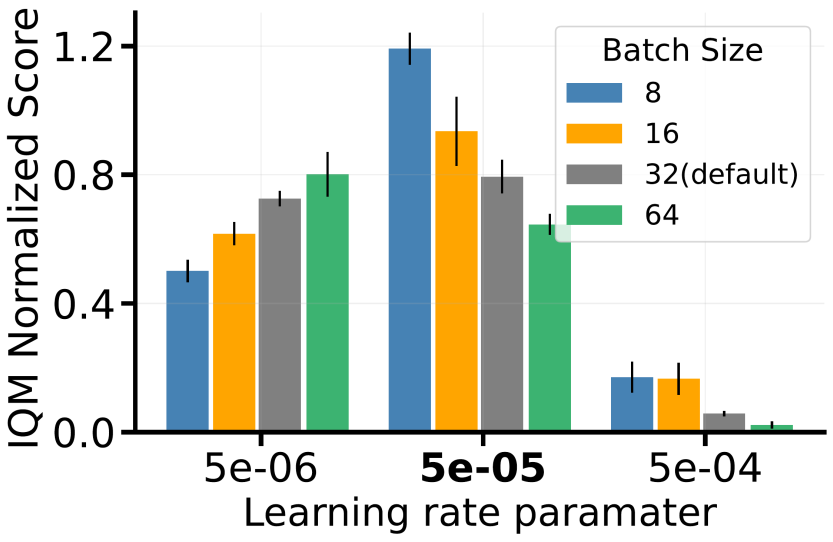

Figure 2.16 – Varying batch sizes for different learning values. Results aggregated IQM of human-normalized scores over 20 games for QR-DQN.

benefits of both adaptive learning rates and momentum, allowing to converge faster and more accurately, especially in high-dimensional parameter spaces (Kingma and Ba, 2015).

Therefore, it is natural to wonder whether an improved learning rate could produce the same effect as simply reducing the batch size. In Figure 2.16 we explored a variety of different learning rates and observe that, although performance is relatively stable with a batch size of 32, it is unable to reach the performance gains obtained with a batch size of 8 or 16.

Second order optimizer effects All our experiments, like most modern RL agents, use the Adam optimizer (Kingma and Ba, 2015), a variant of stochastic gradient descent (SGD) that adapts its learning rate based on the first- and second-order moments of the gradients, as estimated from mini-batches used for training. It is thus possible that smaller batch sizes have a second-order effect on the learning-rate adaptation that benefits agent performance. To investigate this we evaluated, for each training step, performing multiple gradient updates on subsets of the original sampled batch; we define the parameter *BatchDivisor* as the number of gradient updates and dividing factor (where a value of 1 is the default setting). Thus, for a *BatchDivisor* of 4, we would perform 4 gradient updates with subsets of size 8 instead of a single gradient update with a mini-batch of size 32. With an optimizer like SGD this has no effect (as they are mathematically equivalent), but we may see differing performance due to Adam's adaptive learning rates. Figure 2.18 and Figure 2.17 demonstrate that, while there are differences, these are not consistent nor significant enough to explain the performance boost observed.

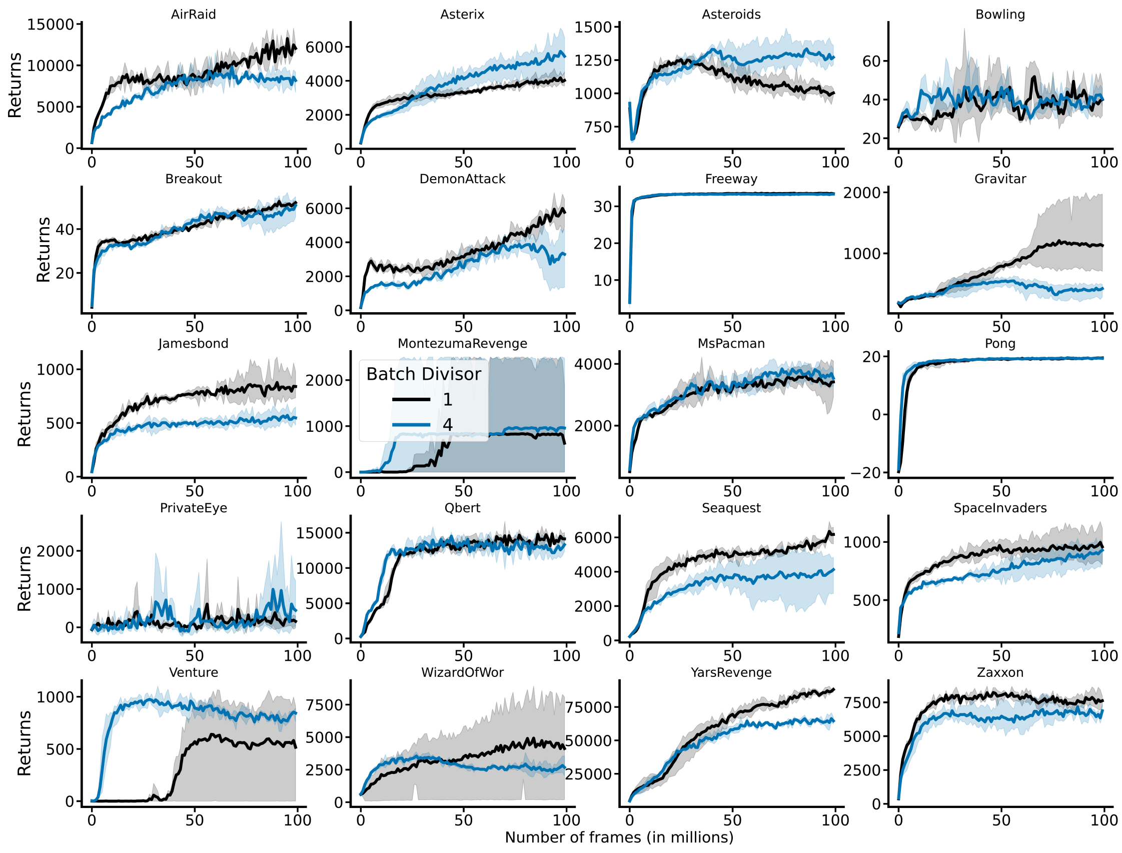

Figure 2.17 – Varying the number of gradient updates per training step, for a fixed batch size of 32. Performance of QR-DQN on tweenty games with different *BatchDivisor* value.

Relationship with multi-step learning In Figure 2.1 we observed that DQN was the only agent where reducing batch size did not improve performance. Recalling that the Dopamine version of Rainbow used is simply adding three components to the base DQN agent, we follow the analyses of Hessel et al. (2018) and Ceron and Castro (2021). Specifically, in Figure 2.21 (top row) we simultaneously add these components to DQN (top left plot) and remove these components from Rainbow (top right plot). Remarkably, batch size is inversely correlated with performance

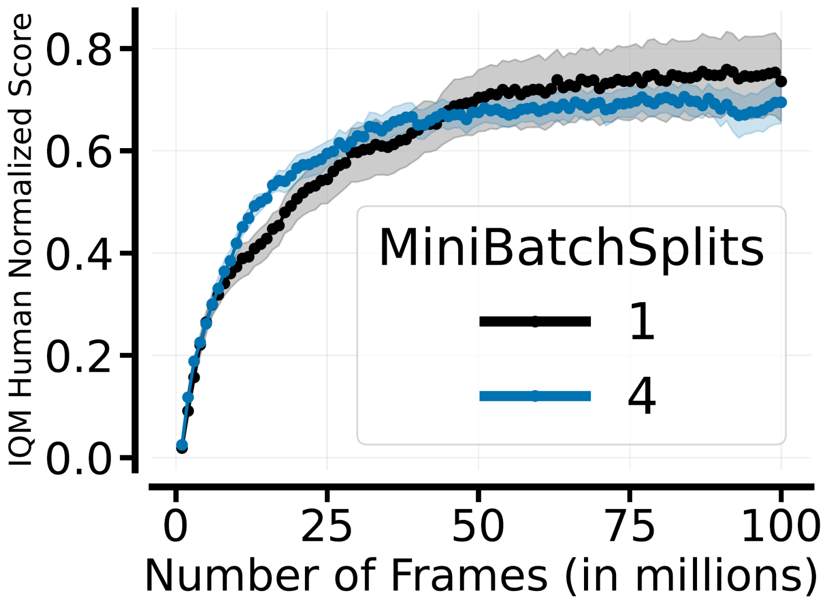

Figure 2.18 – Results aggregated IQM of human-normalized scores over 20 games for QR-DQN.

only when multi-step returns are used. Given that DQN is the only agent considered here without multi-step learning, this discovery explains the anomalous findings in Figure 2.1. Indeed, as Figure 2.20 shows, adding multi-step learning to DQN results in improved performance with smaller batch sizes. To further investigate the relationship between batch size and multi-step returns, in Figure 2.21 we evaluate varying both batch sizes and n-step values for DQN, Rainbow, and QR-DQN. We can observe that smaller batch sizes suffer less from degrading performance as the n-step value is increased.

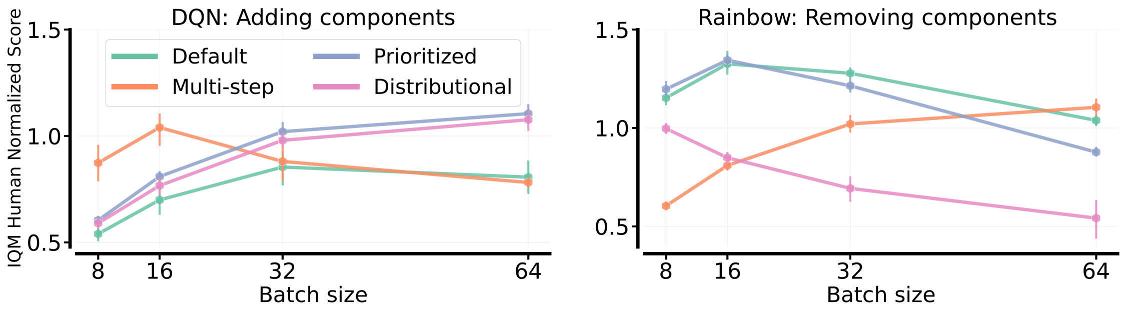

Figure 2.19 – Measured IQM human normalized scores over 20 games with 3 independent seeds for each configuration, displaying 95% stratified bootstrap confidence intervals. **Left:** Adding components to DQN; **Right:** Removing components from Rainbow.

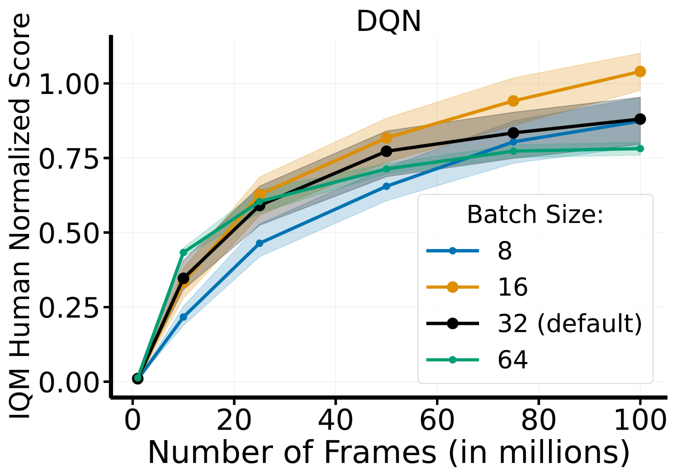

Figure 2.20 – Aggregate DQN performance with n-step of 3

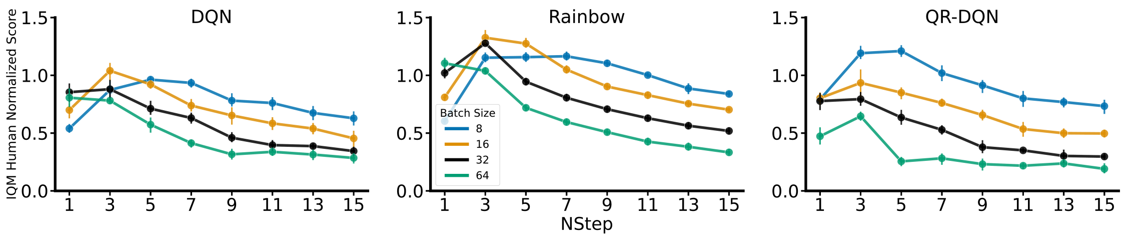

Figure 2.21 – Varying batch sizes and n-steps in DQN (left), Rainbow (center), and QR-DQN (right)

Key insights:

— The small batch effect does not seem to be a consequence of a sub-optimal choice of learning rate for the default value of 32.

— The small batch effect does not arise due to beneficial interactions with the Adam optimizer.

— The small batch effect appears to be more pronounced with multi-step learning.

— When increasing the update horizon in multi-step learning, smaller batches produce better results.

Takeaway: The previous observed gains (arising because of a smaller batch size) does not appear to stem from a sub-optimal selection of learning rate. The default learning rate used (5e-05) was one optimized by prior work for a batch size of 32. It has been previously shown that one should reduce learning rates when increasing batch sizes (Wilson and Martinez, 2003), which is consistent with our findings (e.g. reducing the learning rate can be beneficial when increasing the batch size).

Additionally, this effect remains distinct from any advantageous interactions with the Adam optimizer and interestingly, its presence becomes more prominent within the context of multi-step learning. Notably, as the update horizon in multi-step learning is larger, the advantages of using smaller batches become increasingly evident. This trend highlights that reducing batch size when using multi-step learning can yield improved outcomes, suggesting a nuanced interplay between batch size, learning dynamics, and optimization strategies.

2.2.2.2 Analysis of network optimization dynamics

In this section we will focus on three representative games (Asteroids, DemonAttack, and SpaceInvaders), and include results for more games in the supplemental material. In Figure 2.22 we present the training returns as well as a variety of metrics we collected for our analyses. We will discuss each in more detail below. The first column in this figure displays the training returns for each game, where we can observe the inverse correlation between batch size and performance.

Variance of updates Intuition suggests that as we decrease the batch size, we will observe an increase in the variance of our updates as our gradient estimates will be noisier. This is confirmed in the second column of Figure 2.22, where we see an increased variance with reduced batch size.

A natural question is whether directly increasing variance results in improved

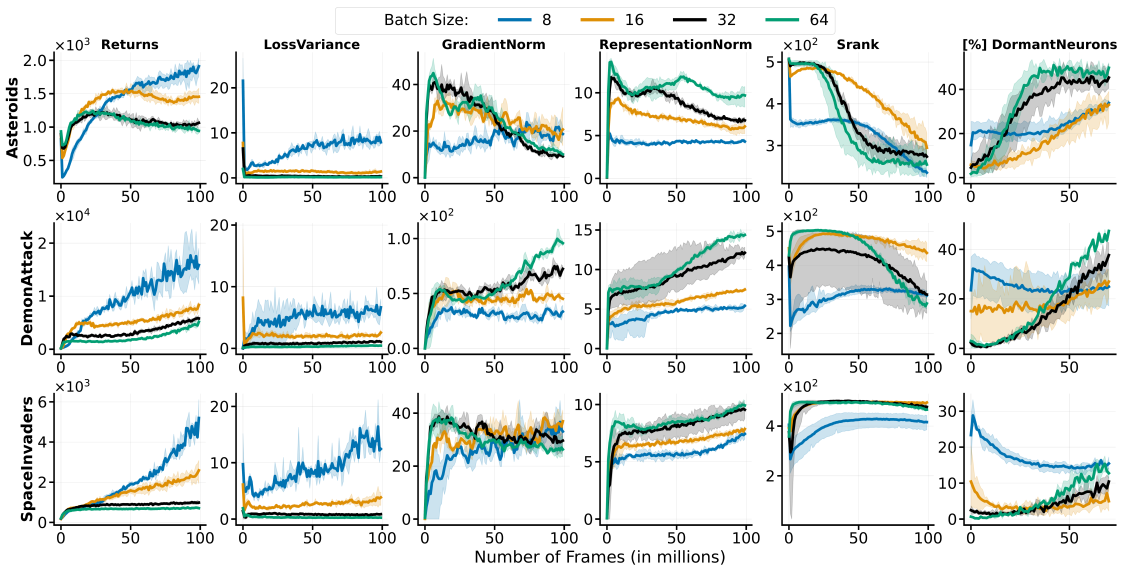

Figure 2.22 – Empirical analyses for three representative games with varying batch sizes. From left to right: training returns, aggregate loss variance, average gradient norm, average representation norm, *srank* (Kumar et al., 2021a), and dormant neurons (Sokar et al., 2023). All results averaged over 3 seeds, shaded areas represent 95% confidence intervals.

performance, thereby (partially) explaining the results with reduced batch size. To investigate, we added Gaussian noise (at varying scales) to the learning target $Q_{\bar{\theta}}$. As Figure 2.23 and Figure 2.24 demonstrates, simply adding noise to the target does provide benefits, albeit with some variation across games.

Gradient and representation norms Keskar et al. (2017) and Zhao et al. (2022) both argue that smaller gradient norms can lead to improved generalization and performance, in part due to less "sharp" optimization landscapes. In Figure 2.22 (third column) we can see that batch size is, in fact, correlated with gradient norms, which may be an important factor in the improved performance.

There have been a number of recent works suggesting RL representations, taken to be the output of the convolutional layers in our networks [5], yield better agent performance when their norms are smaller. Gogianu et al. (2021) demonstrated that normalizing representations yields improved agent performance as a result of a change to optimization dynamics; Kumar et al. (2021b) further observed that smaller representation norms can help mitigate feature co-adaptation, which can

5. This is a common interpretation used recently, for example, by Castro et al. (2021), Gogianu et al. (2021), and Farebrother et al. (2023)

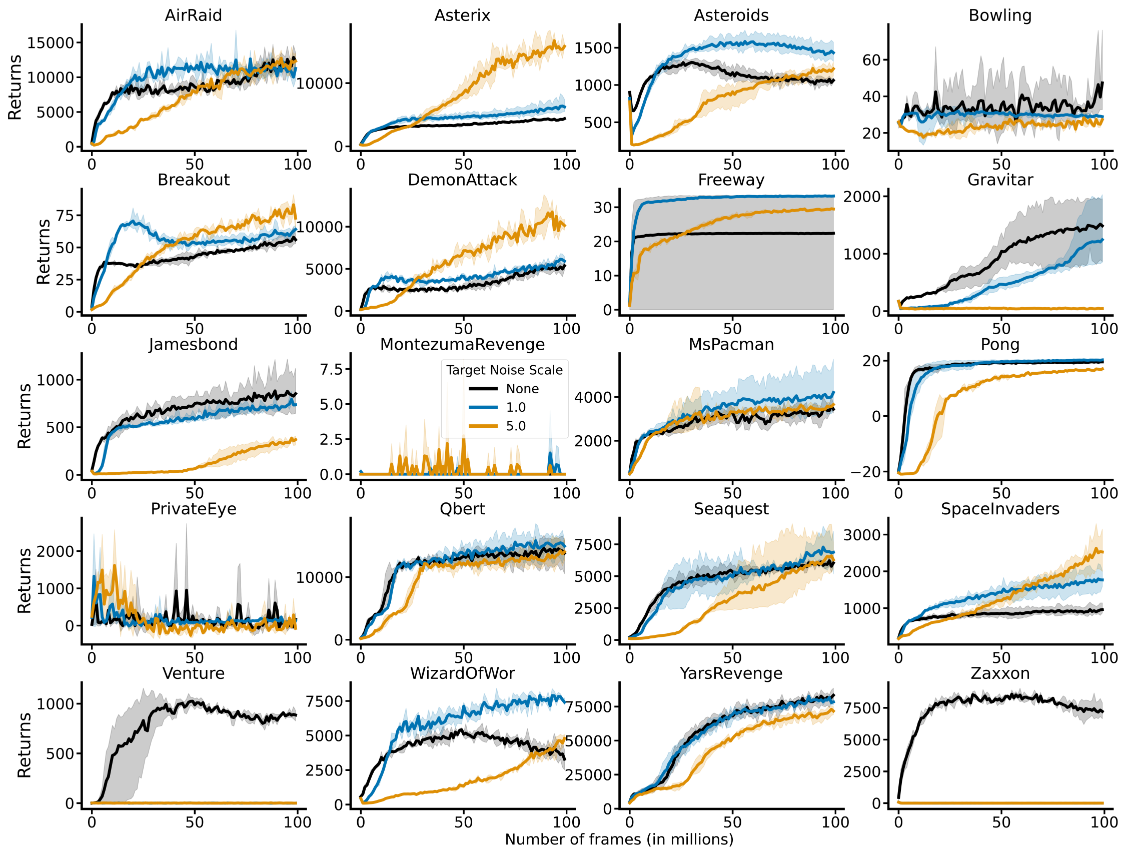

Figure 2.23 – Adding noise of varying scales to the learning target with the default batch size of 32. Performance of QR-DQN on twenty games with different target noise scale values.

degrade agent performance in the offline setting. As Figure 2.22 (fourth column) shows, the norms of the representations are correlated with batch size, which aligns well with the works just mentioned.

Effect on network expressivity and plasticity Kumar et al. (2021a) introduced the notion of the *effective rank* of the representation $srank_\delta(\phi)$ [6], and argued that it is correlated with a network's expressivity: a reduction in effective rank results in an implicit under-parameterization. The authors provide evidence that bootstrapping is the likeliest cause for effective rank collapse (and reduced performance). Interestingly, in Figure 2.22 (fifth column) we see that with smaller batch sizes *srank* collapse occurs earlier in training than with larger batch sizes. Given that there is mounting evidence that deep RL networks tend to overfit during training (Dabney et al., 2021;

6. δ is a threshold parameter. We used the same value of 0.01 as used by Kumar et al. (2021a).

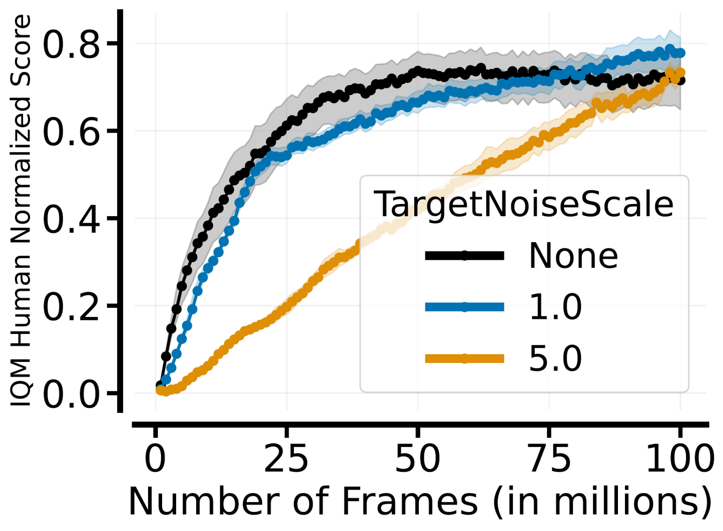

Figure 2.24 – Results aggregated IQM of human-normalized scores over 20 games for QR-DQN.

Nikishin et al., 2022; Sokar et al., 2023), it is possible that the network is better able to adapt to an earlier rank collapse than to a later one.

To further investigate the effects on network expressivity, we measured the fraction of *dormant neurons* (neurons with near-zero activations). Sokar et al. (2023) demonstrated that deep RL agents suffer from an increase in the number of dormant neurons in their network; further, the higher the level of dormant neurons, the worse the performance. In Figure 2.22 (rightmost column) we can see that, although the relationship with batch size is not as clear as with some of the other metrics, smaller batch sizes appear to have a much milder increase in their frequency. Further, there does appear to be a close relationship with the measured *srank* findings above.

Key insights:

— Reduced batch sizes result in increased variance of losses and gradients. This increased variance can have a beneficial effect during training.

— Smaller batch sizes result in smaller gradient and representation norms, which tend to result in improved performance.

— Smaller batch sizes seem to result in networks that are both more expressive and with greater plasticity.

Takeaway: Reducing batch sizes in training neural networks leads to an amplified variance of losses and gradients, which interestingly can yield advantageous outcomes during the learning process. The introduction of this heightened variance can be surprisingly beneficial, fostering exploration in the optimization landscape. Consequently, the utilization of smaller batch sizes yields diminutive gradient and representation norms, ultimately contributing to enhanced performance. Additionally, these reduced batch sizes appear to foster networks with augmented expressiveness and heightened plasticity, highlighting the potential for creating models that are both adaptable and capable of capturing intricate patterns in data.

The conclusions drawn from reducing batch size emphasize the importance of enhancing the stability of the loss landscape as a pivotal measure to foster plasticity. A more seamless loss landscape not only streamlines the optimization process but also demonstrates a propensity for superior generalization capabilities. These plasticity results are likely to have many ancillary benefits, presenting an exciting direction for future investigation.

2.2.3 Offline reinforcement learning

We next turn our attention to the offline reinforcement learning regime (Gulcehre et al., 2020; Levine et al., 2020), where we are given a dataset of sample transitions from which we would like to obtain a policy that performs well. Compared to the online regime, learning offline is more challenging as there is more room for overfitting to the fixed dataset, and there is no possibility for the agent to correct its estimation mistakes by interacting with the environment (as argued by Ostrovski et al. (2021)). We study the effect of using *smaller batch size* parameter for three offline algorithms: DQN, CQL+C51, and CQL+DR3. Except for DQN, these algorithms are specifically tailored to the offline regime, incorporating among other things a penalty to mitigate value overestimation. We follow the training scheme of (Kumar et al., 2021b): each agent is trained on 17 games from the ALE for 200 iterations (where each iteration consists of 62.5K gradient steps), and after each iteration the agent is evaluated for 125K steps on the environment. The offline dataset consists of the transitions experienced during the full training of a DQN agent (Agarwal et al., 2020).

Figure 2.25 illustrates the impact of jointly varying multi-step learning and batch

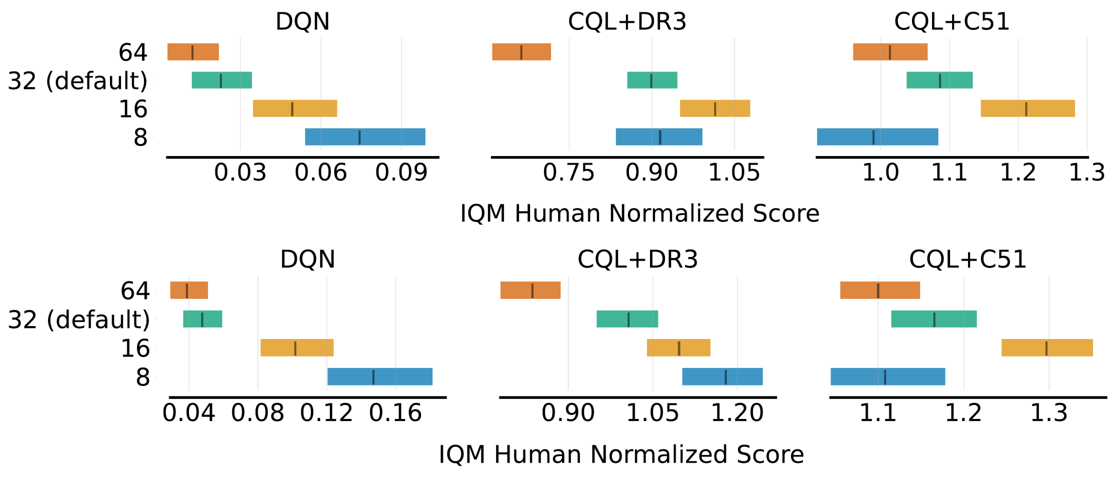

Figure 2.25 – Training curves for DQN, CQL+DR3 and CQL+C51 offline agents with multi-step learning **Top:** $n = 1$ and **Bottom:** $n = 3$. The results for all games are over 5 independent runs.

size as we do in section 2.2.2. In the case of CQL+C51 (the highest-performing method), it is clear that when n is increased from 1 to 3, it is beneficial to also reduce the batch size (from 32 to 16), in line with our previous findings. For DQN and CQL+DR3, the relative performance improvement is larger when using smaller batch size and using multi-step learning.

2.3 Related Work

There is a considerable amount of literature on understanding the effect of batch size in supervised learning settings. Keskar et al. (2016) presented quantitative experiments that support the view that large-batch methods tend to converge to sharp minimizers of the training and testing functions, and as has been shown in the optimization community, sharp minima tends to lead to poorer generalization. Masters and Luschi (2018) support the previous finding, presenting an empirical study of stochastic gradient descent's performance, and reviewing the underlying theoretical assumptions surrounding smaller batches. They conclude that using smaller batch sizes achieves the best training stability and generalization performance. Additionally, Golmant et al. (2018) reported that across a wide range of network architectures and problem domains, increasing the batch size yields no decrease in wall-clock time to convergence for either train or test loss.

Although batch size is central to deep reinforcement learning algorithms, it has not been extensively studied. One of the few results in this space is the work by Stooke and Abbeel (2018), where they argued that larger batch sizes can lead to improved performance when training in distributed settings. Our work finds the opposite effect: *smaller* batch sizes tends to improve performance; this suggests that empirical findings may not directly carry over between single-agent and distributed training scenarios.

Fedus et al. (2020) presented a systematic and extensive analysis of experience replay in Q-learning methods, focusing on two fundamental properties: the replay capacity and the ratio of learning updates to experience collected (e.g. the replay ratio). Although their findings are complementary to ours, further investigation into the interplay of batch size and replay ratio is an interesting avenue for future work. Finally, there have been a number of recent works investigating network plasticity (Nikishin et al., 2022; D'Oro et al., 2023; Sokar et al., 2023), but all have kept the batch size fixed.

2.4 Conclusion

In online deep RL, the amount of data sampled during each training step is crucial to an agent's learning effectiveness. Common intuition would lead one to believe that larger batches yield better estimates of the data distribution and yield computational savings due to data parallelism on GPUs. Our findings here suggest the opposite: the batch size parameter generally alters the agent's learning curves in surprising ways, and reducing the batch size below its standard value is often beneficial.

From a practical perspective, our experimental results make it clear that the effect of batch size on performance is substantially more complex than in supervised learning. Beyond the obvious performance and wall-time gains we observe, changing the batch size appears to have knock-on effects on exploration as well as asymptotic behaviour. Figure 2.16 hints at a complex relationship between learning rate and batch size, suggesting the potential usefulness of “scaling laws” for adjusting these parameters appropriately.

Conversely, our results also highlight a number of theoretically-unexplained effects in deep reinforcement learning. For example, one would naturally expect that increasing the batch size should increase variance, and eventually affect prediction accuracy. That its effect on performance, both transient and asymptotic, should so critically depend on the degree to which bootstrapping occurs (as in n-step returns; Figure 2.21) suggest that gradient-based temporal-difference learning algorithms need a fundamentally different analysis from supervised learning methods.

2.4.1 Future Work

Our focus in this paper has been on value-based online methods. This raises the question of whether our findings carry over to actor-critic methods, and different training scenarios such distributed training (Stooke and Abbeel, 2018). While similar findings are likely for actor-critic methods, the dynamics are sufficiently different in distributed training that it would likely require a different investigative and analytical approach.

Our work has broader implications than just the choice of the batch size hyper-parameter. For instance, our findings on the impact of variance on performance suggest a promising avenue for new algorithmic innovations via the explicit injection of variance. Most exploration algorithms are designed for tabular settings and then adapted for deep networks; our results in section 2.2.1.5 suggests there may be opportunities for exploratory algorithms designed specifically for use with neural networks.

Another interesting direction would be to study how large networks interact with RL algorithms when using smaller batch size. Lastly, our analysis on smaller batch size point towards stabilizing the loss landscape as a crucial step towards promoting plasticity. This approach is expected to yield numerous additional advantages, opening up exciting avenues for future research. A more seamless loss landscape offers several benefits, making optimization easier and leading to improved generalization.

Therefore, investigating the interplay between generalization and small batch size for reinforcement learning could be a promising direction for further exploration, as it can provide deeper insights into their complementary roles. We hope our analyses can prove useful for further advances in the development and understanding of deep networks for reinforcement learning.

3 Conclusion

Experimental design in reinforcement learning is no small task (Eimer et al., 2023; Araújo et al., 2021; Andrychowicz et al., 2021). Running good experiments requires attention to detail and at times significant computational resources. The large computational limitations and the complex interplay between Deep Learning models and RL algorithms make it hard to characterize the learning dynamics of DRL agents and the interplay among components. Despite the great potential of DRL in several domains, DRL is far from well understood. This impedes the development of more advanced algorithms and also prevents the deployment of DRL agents in broader real-world scenarios. Although the learning dynamics of RL agents has been studied with tabular and linear approximation, knowing the learning dynamics of DRL agents is challenging and still an open research problem.

Our work is a contribution in this direction. We investigate the critical role of the interplay between different components (batch size and multi-step learning) in the performance of deep RL algorithms, and explore how this affects the learning dynamics of DRL agents.

Despite recent progress (Sokar et al., 2023; D'Oro et al., 2023; Schaul et al., 2022), there is still much room for developing a better understanding of the many components, and their complex interactions, in training deep networks for RL. Increasing this understanding can aid in the development of new methods and in more rigorous methods for evaluating them. We encourage researchers to conduct experiments to obtain a deeper understanding of new methods so as to better prepare the community (both academic and applied) when building new deep RL agents.

It is worth emphasizing that the landscape of deep RL is marked by its rapid evolution. New methods are constantly being introduced, each with its unique set of promises and challenges. By engaging in experimental pursuits, researchers not only aid in demystifying these novel techniques but also contribute to their understanding and refinement when creating more capable deep RL algorithms.

Given the current level of design complexity achieved by deep reinforcement learning agents, any design choice may have a cascade effect whose consequences we do not predict in advance. Acknowledging this possibility and being aware of it may be an important step toward uncovering latent impacts, and potentially reshaping them into design choices with clearer consequences.

In a broader context, this paper delves into the utilization of a newfound insight to inform the development of forthcoming deep reinforcement learning algorithms. We view this endeavor as an illustration of how the advancement of proficient deep reinforcement learning methods should encompass more than just extending current algorithms or devising novel ones. It should also involve uncovering phenomena associated with deep RL systems, along with the development of strategies to harness them in order to enhance performance.

Bibliography

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv:2303.07507*, 2023. Cited on page 13.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. 2020. Cited on page 43.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021. Cited on pages vii, 14, 16, 20, 24, and 26.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Beyond tabula rasa: Reincarnating reinforcement learning. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. Cited on page 24.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari, 2020. Cited on page 15.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=nIAxjsniDzg`. Cited on page 48.

João Guilherme Madeira Araújo, Johan Samir Obando Ceron, and Pablo Samuel Castro. Lifting the veil on hyper-parameters for value-based deep reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021. Cited on page 48.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020. Cited on pages 14 and 15.

Brian R Bartoldson, Bhavya Kailkhura, and Davis Blalock. Compute-efficient deep learning: Algorithmic trends and opportunities. *Journal of Machine Learning Research*, 24:1–77, 2023. Cited on page 32.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/afda332245e2af431fb7b672a68b659d-Paper.pdf`. Cited on page 30.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 2013. Cited on pages 14 and 20.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 449–458, 2017. Cited on pages 11 and 17.

Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional reinforcement learning*. 2023. Cited on page 11.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. Cited on page 31.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. Cited on page 1.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. Cited on page 31.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL `http://arxiv.org/abs/1812.06110`. Cited on page 20.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34: 30113–30126, 2021. Cited on pages 15 and 40.

Johan Samir Obando Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1373–1383. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/ceron21a.html`. Cited on page 36.

Johan Samir Obando Ceron, Marc G Bellemare, and Pablo Samuel Castro. Small batch deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=wPqEvmwFEh`. Cited on page 18.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021. Cited on page 16.

W. Dabney, M. Rowland, Marc G. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *AAAI*, 2018a. Cited on pages vii, viii, 11, 21, 22, 24, 28, and 29.

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of*

Machine Learning Research, pages 1096–1105. PMLR, 2018b. Cited on pages viii, 11, 21, 28, and 29.

Will Dabney, Andre Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G. Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. Cited on page 41.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=OpC-9aBBVJe`. Cited on pages 19, 45, and 48.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. Cited on page 7.

Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022. URL `https://openreview.net/forum?id=imxyoQIC5XT`. Cited on page 34.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them, 2023. Cited on page 48.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018. Cited on pages 4 and 24.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=oGDKSt9JrZi`. Cited on page 40.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. *arXiv preprint arXiv:2007.06700*, 2020. Cited on pages 20 and 45.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alexander Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *Proceedings of the International Conference on Representation Learning (ICLR 2018)*, Vancouver (Canada), 2018. Cited on page 34.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. Cited on page 7.

Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: An optimisation perspective. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3734–3744. PMLR, 18–24 Jul 2021. Cited on page 40.

Noah Golmant, Nikita Vemuri, Zhewei Yao, Vladimir Feinberg, Amir Gholami, Kai Rothauge, Michael W Mahoney, and Joseph Gonzalez. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018. Cited on page 44.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016. Cited on pages 1, 4, and 7.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7766–7792. PMLR, 17–23 Jul

2022. URL https://proceedings.mlr.press/v162/graesser22a.html. Cited on page 13.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. Cited on page 16.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020. Cited on page 43.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018. Cited on page 34.

Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. Cited on page 34.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Cited on pages 4 and 7.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018. Cited on pages 11, 16, 21, and 36.

Geoffrey E Hinton. Distributed representations. 1984. Cited on page 3.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay, 2018. Cited on page 12.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. Cited on page 4.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020. Cited on page 13.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. Cited on page 5.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016. Cited on page 15.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. Cited on page 31.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *ICLR*, 2019. Cited on page 15.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *International Conference on Learning Representations*, 2020. Cited on pages 16 and 26.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. Cited on page 44.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=H1oyRlYgg`. Cited on pages 19 and 40.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited on page 7.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`. Cited on page 35.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020. Cited on page 16.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. Cited on page 17.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. 2021a. URL `https://openreview.net/forum?id=O9bnihsFfXU`. Cited on pages viii, 13, 40, and 41.

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. Dr3: Value-based deep reinforcement learning requires explicit regularization. *arXiv preprint arXiv:2112.04716*, 2021b. Cited on pages 17, 40, and 43.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022. Cited on page 17.

Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988. Cited on page 7.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. Cited on page 5.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015. Cited on page 2.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472. PMLR, 2016. Cited on page 5.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning, 2020. Cited on page 14.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. Cited on pages 17 and 43.

AJ Lohn and Micha Musser. Ai and compute: How much longer can computing power drive artificial intelligence progress? *Center for Securty and Emerging Technology*, 2022. Cited on page 31.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=ZkC8wKoLbQ7`. Cited on page 13.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks, 2023. Cited on page 13.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents, 2017. Cited on pages 14 and 20.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *ArXiv*, abs/1804.07612, 2018. Cited on pages 33 and 44.

Bogdan Mazoure, Remi Tachet des Combes, Thang Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *arXiv preprint arXiv:2006.07217*, 2020. Cited on page 15.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. Cited on page 3.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *To appear in The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=vhFu1Acb0xb`. Cited on page 16.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. Cited on pages 10, 11, 12, 19, 21, 24, and 30.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, pages 16828–16847. PMLR, 2022. Cited on pages 13, 19, 42, and 45.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. URL `https://arxiv.org/abs/1912.06680`. Cited on page 15.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=nPHA8fGicZk`. Cited on page 43.

Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach.* " O'Reilly Media, Inc.", 2017. Cited on page 5.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017. Cited on page 34.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014. Cited on page 8.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. Cited on page 31.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019. Cited on page 4.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. Cited on page 31.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. Cited on page 9.

Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017. Cited on page 4.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. Cited on page 7.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. Cited on page 3.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015. Cited on page 12.

Tom Schaul, André Barreto, John Quan, and Georg Ostrovski. The phenomenon of policy churn. *Advances in Neural Information Processing Systems*, 35:4235–4246, 2022. Cited on page 48.

Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010: 20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III 20*, pages 92–101. Springer, 2010. Cited on page 5.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. Cited on page 2.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. Cited on page 16.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020. Cited on page 26.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=uCQfPZwRaUu`. Cited on pages 10 and 16.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 30365–30380. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/schwarzer23a.html`. Cited on pages 16 and 24.

Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019. URL `http://jmlr.org/papers/v20/18-789.html`. Cited on page 19.

Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017. Cited on page 5.

Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84:109–136, 2011. Cited on page 17.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. Cited on page 4.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Cited on page 6.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. *arXiv preprint arXiv:2302.12902*, 2023. Cited on pages viii, 13, 19, 24, 40, 42, 45, and 48.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 2014. Cited on pages 5 and 6.

Adam Stooke and Pieter Abbeel. Accelerated methods for deep reinforcement learning. *CoRR*, abs/1803.02811, 2018. URL `http://arxiv.org/abs/1803.02811`. Cited on pages 45 and 46.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019. Cited on page 31.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`. Cited on pages 8, 9, 10, and 11.

Adrien Ali Taiga, William Fedus, Marlos C. Machado, Aaron Courville, and Marc G. Bellemare. On bonus based exploration methods in the arcade learning environment. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJewlyStDr`. Cited on page 30.

Tijmen Tieleman and G Hinton. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *Technical report*, 2017. Cited on page 7.

Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/1b742ae215adf18b75449c6e272fd92d-Paper.pdf`. Cited on page 26.

Hado P Van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019. Cited on pages 15 and 16.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019. Cited on page 15.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL `https://doi.org/10.1007/BF00992698`. Cited on page 9.

D.R. Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural networks : the official journal of the International Neural Network Society*, 16 10:1429–51, 2003. URL `https://api.semanticscholar.org/CorpusID:652801`. Cited on page 39.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. Cited on page 17.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021. Cited on page 16.

Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26982–26992. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/zhao22i.html`. Cited on page 40.