Deep Reinforcement Learning as Text Generator in Image Captioning

By

Farhad Amouzgar

A thesis submitted to Macquarie University

Masters of Research

Department of Computing

November 2019

MACQUARIE University

SYDNEY·AUSTRALIA

Typeset in LaTeX 2_ε.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed) _____ Date: _____

Farhad Amouzgar

Dedication

To all my loved ones.

Acknowledgements

I would like to thank my supervisors Dr. Amin Beheshti and A/Prof. Mark Dras for their support, instruction and excellent supervision throughout my research. Without whom it would not be possible to complete this work.

I am thankful to all the members of the Data Analytics Research Lab at Macquarie University for their friendship and helpful comments. Also, I would like to thank the administrative and technical staff of the Department of Computing for all their help during my research.

This thesis is dedicated to my wife, who has always been the source of strength, love and patience, my mother and my sister for their lifelong love and support and the memory of my father who taught me the power of curiosity and ignited the love of wisdom.

Abstract

Reinforcement learning (RL), as one of the oldest AI paradigms, has led to exciting results in recent years. Even though the research frontiers in the field are game playing and robotics, the natural language processing (NLP) community has also found many applications of RL as a solution for optimizing non-differentiable metrics in deep learning, including in text generation, image captioning and chatbots. However, current literature is mainly focused on the REINFORCE algorithm and its derivatives. REINFORCE is a robust algorithm, but it dates back to the 1990s and suffers from high variance compared to modern RL algorithms. To address this challenge, we study and analyze the recent state-of-the-art in RL. Taking image captioning as our specific NLP use case, we identify Proximal Policy Optimization (PPO) RL algorithms as suitable updates for REINFORCE, and propose methods for optimizing non-differentiable captioning metrics based on these. We experimentally evaluate them with respect to the REINFORCE-based standard and find that, while the static clipping mechanism of PPO is the key aspect of state-of-the-art results in game playing, it does not improve over REINFORCE in image captioning; rather, the actor-critic aspect of the algorithms has a more significant impact on the convergence of the model.

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

The field of artificial intelligence (AI) began with high hopes about the capabilities of symbolic representation and logical reasoning. When the initial ideas failed to deliver, it took a few decades for the scientific community to realize that there are more novel routes to intelligent behavior. These new methods, which were appropriated from established fields of study, such as statistics, neuroscience, and biology, are classified as machine learning (ML). As Arthur Samuel put it, ML is a field of science that tries to use statistical methods to develop systems that can operate without being explicitly programmed [7]. Learning is this iterative process of autonomous programming.

ML is a vast field of study, and one of the most fruitful and active topics in computer science. It is difficult to imagine a day without spam filters, search engines, map applications, voice recognition programs, automatic translation systems, face detection services, handwriting detection systems (OCRs), image captioning, ML-based controlling methods such as Autopilots and many more. An example of image captioning (the task that automatically generates descriptions for photos)

produced by the algorithm at the core of this thesis, is demonstrated in Figure 1.1.

FIGURE 1.1: Image Captioning. Caption: A man riding a wave on a surfboard in the ocean

There are many ways of characterizing ML algorithms, but one of the most informative methods is considering the effects of the outside world on the emergence of intelligence. By using this criterion, we can count at least three types of learning algorithms. The first is when learning is achieved by processing a labelled set of data, which is known as supervised learning. The second type is when the positive and negative feedbacks of the environment encourages and obstructs specific actions. This environment-oriented algorithm is called reinforcement learning (RL). Finally, when labels and feedbacks are unavailable, the data can be explored in the hope of finding interesting patterns, which creates the third class, unsupervised learning [8, 9].

RL emerged from different disciplines, sometimes fundamentally different from AI, such as psychology and optimal control [10]. In all of them, however, the primary goal is studying the adaptability of systems or organisms in their environments. RL matured in the 1990s and early 2000, but with the rise of deep learning, deep reinforcement learning began to show impressive results in robotics [3, 5] and video game playing [4, 6]. At first, combining deep neural networks and RL was challenging, but in 2013 the Deep Q-Network (DQN) paper proposed novel ideas that overcame the initial obstacles and achieved human-level performance in Atari games [4].

DQN's achievements ignited a wave of new methods in deep RL. Most of them are re-appropriated versions of older RL algorithms with ideas from deep learning. The DQN itself is a new version of Q-learning, a key subtype of RL known as off-policy, optimized with a neural network. DDPG [5], double DQN [11], dueling DQN [12] also originated in the Q-learning method [13].

In a parallel line of expansion, another subclass of RL referred to as on-policy policy gradient

algorithms also began to flourish. A very foundational and robust method in this scope is REINFORCE [14]. Ideas from REINFORCE and deep neural networks helped the research community to develop highly advanced on-policy methods such as PPO [3], TRPO [15], and A3C [16].

1.1 Motivation

Image captioning as an automatic generation of description for images is a challenging task that requires context comprehension and meaningful articulation. There were captioning techniques before the deep learning era that could not provide original and human-level captions due to their reliance on predefined templates [17] or object similarity in photos [18]. It has been recently shown, however, that methods based on deep neural networks such as convolutional neural networks, recurrent neural networks, and attention mechanisms can achieve outstanding performance in image captioning [19–21]. There are a variety of deep captioning techniques, but almost all of them utilize a convolutional neural network (CNN) network to detect the visual objects and a following recurrent neural network (RNN) to verbalize the perceptions, as shown in Figure 1.2.

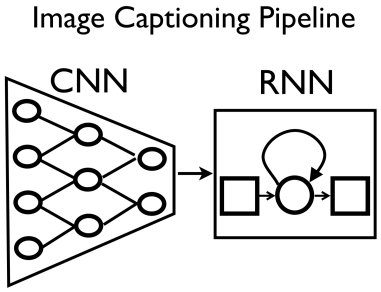

FIGURE 1.2: A deep learning architecture for image captioning

To facilitate the learning process, the NLP community invented several scoring metrics to assess the quality of produced captions against the human-generated ones such as BLEU, CIDEr, ROUGE, METEOR, and SPICE. One of the essential difficulties in training the network with these metrics is their non-differentiability, which implies that the gradient cannot backpropagate and optimize the model using them. The traditional approach to tackle this issue was the cross-entropy criterion in optimization, but this leads to relatively poor results. An insight from Daumé III et al.(2009) [22] was to use RL to get around the problem of non-differentiability. The principal candidate used for

this purpose has generally been the REINFORCE [14] algorithm [2, 23, 24].

REINFORCE dates back to the 1990s and particularly suffers from high variance. This problem is usually addressed by subtracting a baseline from the original REINFORCE. The way the baseline was calculated, however, influences the performance of the baselined REINFORCE. One of the state-of-the-art methods in image captioning that utilizes this technique is the Self-Critical Sequence Training (SCST) [24], which informs its baseline by its output from the test interval.

Even though these advances have achieved impressive results, they are still built on top of the REINFORCE. As mentioned, the rapid progress of RL in recent years provides a repository of methods that can replace REINFORCE and its derivatives. For instance, the Trust Reign Policy Optimization (TRPO) algorithm observed the poor performance of the policy gradient methods (e.g., REINFORCE) in their inability to consider past policies. Hence, solving the problem by adding more constraint to the update process. The Proximal Policy Optimization (PPO) recognizes the same discrepancy but proposes a more straightforward solution, replacing the constraint with a clipping mechanism. Some other solutions employ actor-critic approaches to enhance the outcome's performance.

1.2 Objective of the Study

In this research, we review and analyze the state-of-the-art in RL and image captioning to recognize what might constitute feasible alternatives to REINFORCE. Then, we investigate selected RL alternatives experimentally to understand the behavior of their various components. Finally, we propose a new method for image captioning that incorporates selected RL alternatives. We empirically evaluate our proposed method.

2

Background and Literature Review

Our research is built upon two lines of inquiry, image captioning, and deep reinforcement learning. First, we lay out the historical advancements of deep neural networks and the way they revolutionized other fields such as image captioning and reinforcement learning. Then, we explain the state-of-the-art in image captioning, and deep reinforcement learning and the way modern reinforcement learning algorithms might improve image captioning methods.

2.1 Artificial Neural Networks and Deep Learning

According to the connectionist model of mind, intelligence, behaviour, response, sensorimotor functions, language and ultimately consciousness all emerged out of a densely connected network of cells in the brain and other parts of the body. The computational view of neural networks was first proposed by Warren McCulloch and Walter Pitts in 1943. In their seminal work, they

demonstrated the capability of neural networks in computing logical and arithmetic expressions [25]. Their work signified a pivotal point in artificial intelligence and founded the field of artificial neural networks (ANNs).

After the introduction of neural networks as computational components, some discoveries shaped the ANN discipline. The perceptron was introduced in the 1950s by Rosenblatt as a linear classifier and the building block of neural networks. Perceptrons had limitations in function approximation which was pointed out by Minsky and Papert in 1969 that led to some stagnation in the field. Multi-layer perceptrons (MLP), however, later solved this weakness which was made possible by Werbos's backpropagation approach. Since the 1980s ANNs have been an active area of research and since the first decade of the 2000s and after the discovery of solutions to develop deep multilayer perceptrons, ANNs became one of the most successful methodologies in developing intelligent systems. This success also led to the rebranding of the field from artificial neural networks to deep learning [26].

As mentioned, the simplest type of neural network is a perceptron. As illustrated in Equation 2.1, a perceptron unit computes the weighted sum of its inputs $x_1 \dots x_n$, adding a bias term b, and then applies an activation function $f()$ such as Linear, Sigmoid, Hyperbolic tangent, ReLU, or ELU. This simple unit by itself can be used as a binary classifier or a simple predictor. By adding and stacking these elements in the right structure, highly complex functions can be learned by the neural network.

The mentioned activation functions transform the output of the neuron to a preferred non-linear output in a way to fit a desired range or shape. For instance, the Sigmoid function, which is inspired by the activation function used in biological neurons, is employed when a probability type output is expected as it ranges between zero and one. Tanh doubles the Sigmoid output range and is applied in many areas, including recurrent neural networks. ReLU and ELU are more efficient and can potentially prevent the vanishing and exploding gradient problem in deep networks [1, 26].

$$y = f(\sum_{i}^{n} x_{i=1} w_i + b) \tag{2.1}$$

In the rest of this section, we give brief overviews of three types of artificial neural networks that are important in deep learning and specifically in the work in this thesis, namely Fully Connected

Networks, Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNNs). More detail is available from sources such as [1, 26].

2.1.1 Fully Connected Networks

A fully connected network or a dense network consists of layers of parallel neurons, and each layer's neurons are connected to all of the neurons in the adjacent layers. There is no connection between neurons in one layer as it is illustrated in Figure 2.1. This structure, which resembles some biological neural networks, can become deeper and broader in order to better approximate complex functions. Intuitively the size of a network increases the learning time.

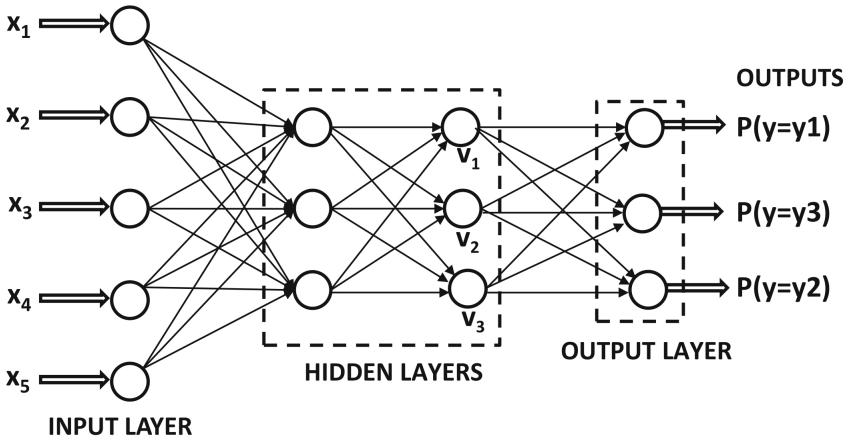

FIGURE 2.1: A Fully Connected Network [1]

Structure A fully connected network is categorized into three sections, the input layer which feeds the network with data, the output layer that passes the calculated prediction to the outside world, and hidden layers. Deeper networks have more hidden layers.

Training Neural networks operate in two modes, forward and backward. The forward mode moves the data from the input layer to the output layer to generate the prediction. This prediction is initially very inaccurate and to train the network, after computing the loss function (a function of the difference between the actual output and the prediction) the network switches to backward to optimize the network weights. The iterative process of prediction, computing the loss value,

and then backpropagation and optimization is the essence of learning in neural networks [26]. In a one-layer neural network, computing the error (or loss interchangeably) is straightforward as it depends on the weights of neurons in that layer. In deeper networks, however, the forward pass error depends on a complex combination of the weights in all the layers. The backpropagation algorithm addressed and solved this issue using a dynamic programming approach. It goes back in the network and based on the neuron's weight finds the role of each neuron in the generated error, and updates it accordingly [1].

2.1.2 Convolutional Neural Networks

Almost all artificial neural networks are biologically inspired; convolutional neural networks (CNNs) are specifically inspired by the receptive field of the brain in mammals. CNNs are highly capable of capturing spatial relations in grid structures such as images. This feature makes them a good choice for pattern recognition and feature extraction from high dimensional datasets, e.g., photos and videos [1].

The concept of filter (kernel) tremendously reduces the complexity of CNNs compared to fully connected networks, in terms of the number of parameters that need to be learned. A filter is a sliding window that traverses the image or a layer of CNN and connects certain parts of them (e.g., pixels, neurons) to the neurons in the next layer. The filter convolves those values with its predefined parameters, hence the name convolution [1, 26].

Two other essential components of CNNs are pooling layers and ReLU activation function. ReLU is a half linear function that returns $\max(0, \text{input})$ and can have a positive effect on efficiency. By using pooling layers (max-pooling for instance), a specific statistical value such as the maximum value of a certain section of the CNN replaces its adjacent values, which acts as a regularizer or a dimensionality reduction mechanism. The ReLU and similar activation functions also enhance the performance by avoiding the vanishing and exploding effects of saturating activation functions such as sigmoid and hyperbolic tangent [26].

One of the contributing factors in the rapid advancement of CNNs is the ImageNet competition (ILSVRC) which led to many deep and complex architectures such as AlexNet, ZfNet, VGG, GoogLeNet, and ResNet. Pre-trained CNNs are highly beneficial and are starting points for research groups to apply transfer learning and save time and computational power. As shown in Figure 2.2,

in the end, we typically connect the CNN to a fully connected or recurrent network for regular machine learning activities such as classification, prediction, or generation [1].

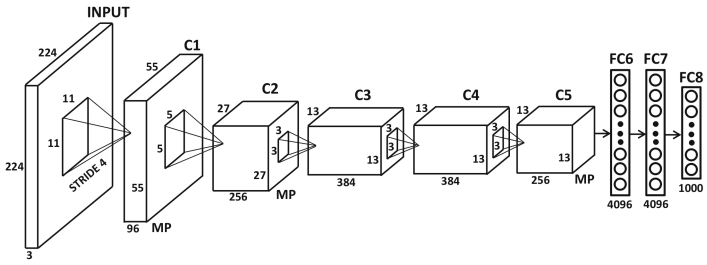

FIGURE 2.2: A CNN as the feature extractor with its following dense network for classification [1]

2.1.3 Recurrent Neural Networks

As noted, dense networks are valid options to deal with high dimensional data. CNNs have the same role, but they are most effective when the data is grid-like and spatially related. The assumption in these networks is the independence of features and the absence of time. Thus, to process, transform, or generate sequential datasets such as time-series, text, videos, and the like, a different type of neural network is required. Units that are used to handle sequentiality are called Recurrent Neural Networks (RNNs). There are different variations of RNNs. Currently popular ones are Elman RNNs [27], Bi-directional RNNs [28], Gated Recurrent Units (GRU) [29], and the Long-Short Term Memory (LSTM) [30]. LSTMs and the more recent GRUs can overcome the common issue of the vanishing gradient problems. LSTMs are also explicitly utilized in this thesis [1].

An RNN is structurally a multi-layer perceptron with a self-loop that allows the network to be informed about the earlier data, passed through the network in previous iterations. In each iteration, the RNN can potentially generate output and communicate a weight to the next iteration. The weight is a way of considering the hidden representation of the RNN with its current input. An abstract view of a typical RNN is depicted in Figure 2.3 [26, 30].

The hidden state of the network updates recursively using an activation function f (e.g. Hyperbolic Tangent), using a formula like $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$. The amalgamation of $\mathbf{h}_t$ and W_{hy} will generate the output $\mathbf{y}_t$ in each step.

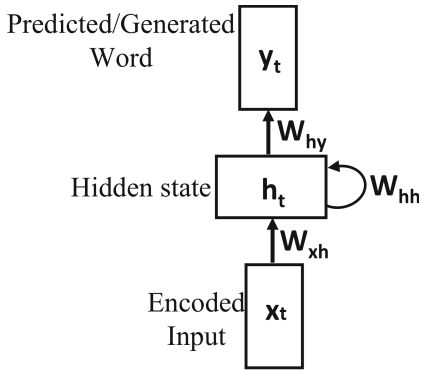

FIGURE 2.3: A typical RNN [1]

2.2 Image Captioning

Image captioning is the practice of generating textual contents to describe a picture, its constituting objects, and their interrelations. The interaction of language and image can be bi-directional; in other words, what is said can be imagined (generated), and what is shown can be described. Image captioning covers the latter as shown in Figure 1.1. Text-to-Image is another active field of research in Generative Adversarial Networks (GANs)[31], and it is not usually discussed in image captioning.

Image captioning has been one of the early challenges in artificial intelligence, image processing, and natural language processing. It can be argued that it dates back to 1960s when Marvin Minsky asked his student to make the first image captioning program by connecting a computer to a camera [32, 33]. One of the reasons that make image captioning complex is its interdisciplinary nature with different ways to approach the problem.

Historically, image captioning methods can be divided into three main categories which reflect three different perspectives: template-style, retrieval-style, and end-to-end [32]. The template-style algorithms provide templates for descriptors that will be filled by the features extracted from the image [17]. One of the advantages of these methods is their predictability and consistency of output. However, these methods are rigid due to their dependency on the pre-defined templates. The second type, the retrieval-style, uses the extracted features and search for similar images [18]. Once similar ones are found, the corresponding captions are copied or will be used to generate a relevant description. These methods are tightly coupled with the training datasets, which has

its advantages and shortcomings. For instance, a large and comprehensive dataset would lead to acceptable performance, whereas a small and unrelated one will generate poor captions. The third class, as shown in Figure 2.4, uses a combination of CNNs (Sec 2.1.2) and RNNs (Sec 2.1.3) in a sequential manner which resulted in promising outcomes, and is the focus of our study.

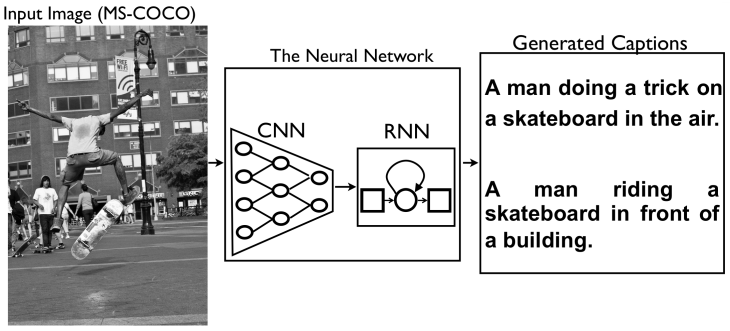

FIGURE 2.4: An end-to-end image captioning system

These methods rely on special ML techniques that matured since the dawn of deep learning such as CNNs (Sec 2.1.2), RNNs (LSTMs, GRUs and Bi-directionals) (Sec 2.1.3), and deep neural networks (Sec 2.1.1). The algorithms were supported by other tools and techniques such as massive improvements in computational power and its availability, optimization methods, more advanced methods of network initialization and regularization, transfer learning, better and more accessible libraries for deep learning, and bigger datasets for learning algorithms.

2.2.1 End-to-End Systems

The key idea of end-to-end systems in image captioning is to utilize the best characteristics of the two neural networks, namely CNNs and RNNs. The resulting architecture operates like an encoder-decoder [20, 34]. CNNs are best to extract features from images, which is the encoding phase. Many available solutions are built upon a prepared Residual Network (ResNet) [35] that has been trained by ImageNet [36] beforehand. The extracted and abstract features are fed into an RNN (LSTM or GRU) that in part encodes them into captions and generates the sequence of words.

Other approaches have also explored in the context of the encoder-decoder structures such as attention mechanisms [19, 37], and the application of reinforcement learning as a way of direct

optimization based on NLP metrics [24, 38, 39]. The latter, which is the focus of our research, will be discussed in more details in section 2.4.

2.2.2 Datasets

Many datasets can be considered when it comes to text and text generation. It is relatively easy to collect social media comments, technical reports, scientific papers, books, and the like since currently, the textual content is abundant. The challenge has been providing textual datasets that also include and relate the text to other forms of data such as images, videos, and time series. To address this issue in image captioning, many research groups and organizations have collected and annotated image datasets using online crowdsourcing platforms [40], which have led to benchmark repositories for image processing and captioning.

2.2.2.1 PASCAL VOC Project and Flicker Datasets

Visual Object Classes (VOC) challenge was a classification competition held between 2005 and 2012 [41], where researchers had to use their methods on a common dataset. The organizer was Pattern Analysis, Statistical Modeling, and Computational Learning (PASCAL), hence, released datasets were called PASCAL VOC. These datasets contained 6000 images collected from the photo-sharing website, Flicker, and captioned by the crowdsourcing service, Amazon Mechanical Turk [40].

In later challenges, the number of images increased to 8000 and 30000. Annotation mechanisms and methods also advanced. Thus, two new image classification and captioning benchmark datasets appeared called, Flicker8k and Flicker30k, respectively [42]. These datasets set the context for the next important dataset, Microsoft COCO.

2.2.2.2 Microsoft COCO

After the successes of deep learning in numerous fields of AI, specifically, computer vision, sequence generation, and translation, image captioning became a focus of attention again. PASCAL and older Flicker-based datasets defined a methodology for data gathering and annotation, but their sizes were not enough for the demands of researchers. Thus, in 2014, a new dataset was gathered

and introduced by Microsoft called Common Objects in Context or COCO [43].

In data gathering, like older datasets, Amazon Mechanical Turk was used. There are two datasets, MS COCO c5 and c40. In c5 images have five reference captions. C40 has forty sentences for 5000 randomly selected images. In COCO c5, there are 330,000 images with 80 object categories, and 1.5 million object instances. Object superpixel stuff segmentation and object segmentation are also considered in which the former simplifies segmentation by aggregating similar pixels in larger entities (superpixels), and the latter groups the entire object in one frame [44]. An evaluation server was also provided to compare the performance of different algorithms. The evaluation itself is in terms of NLP metrics such as BLEU, CIDEr, ROUGE, and METEOR [45].

2.2.3 Metrics and The Automatic Evaluation of Text

In evaluating the quality of a text, and its relevance to a particular context such as an image, text in another language or text summarization, human judgement is the benchmark. However, there are obstacles in employing humans for these assessments that considering an alternative would be highly beneficial. Four main problems are (1) the cost, (2) the speed (or lack of it) (3) non-reusability (4) fallibility [46]. Human evaluation is more expensive and slower than most computational techniques. It cannot be duplicated and multiplied and is subject to human error.

For all these reasons, six different text evaluation methods have been developed. Although they usually do not surpass human assessment, their estimates are consistent with human judgment [47]. The evaluation results can be used for various purposes. In our research, we consider them as rewards for the learning algorithm.

Most metrics use a similarity function to compare generated text in translation and summarization with the ground-truth sentences. BLUE [46], which is a precision score, examines texts based on a paramerized mean over the length of the n-grams. The size of the n-gram makes a different BLEU score as a result. METEOR and ROUGE utilize F-score and F-measures on n-grams, respectively. CIDEr employs TF-IDF values as a comparison measure of the generated and the target text. The most recent one, SPICE, is mainly aiming at assessing the text created in the process of image captioning, considering semantic parsing in the method. In this study, NLP scores are considered as an external reward system to the proposed methods [47].

2.3 Reinforcement Learning

As mentioned in the introduction, ML can be categorized into three different classes, supervised learning, unsupervised learning, and reinforcement learning (RL). RL emerged from psychology and optimal control as an adaptive and learning mechanism in the process of interaction with the environment. RL matured in the late 20th century, but when deep learning appeared in the late 2000's [1], the era of deep reinforcement learning began with impressive and human-level results in video games and robotics. One of the first notable results was the Deep Q-Network (DQN) algorithm that overcame the incompatibility of RL and deep neural networks [4].

2.3.1 Terminology of Reinforcement Learning

Two of the key concepts in RL are state and action. A state can be represented to the agent in various ways, and it depends on the problem that the agent is trying to solve. For instance, it might be a matrix of values for Chess or the AlphaGo game, a collection of pixels for Atari games, or a graph for a network navigation agent. Action is usually a set of predefined abilities given to the agent beforehand. The goal of learning is to find the right sequence of steps that maximize the total reward for the agent.

The reward signal r is also another fundamental notion in RL. The reward, which is usually represented by real value, is formally defined as a three-argument function such that $r : S \times A \times S \rightarrow \mathbb{R}$. It evaluates the quality of the action A, which changes one state to another [10]. Under normal circumstances, negative values indicate a negative reward that discourages the agent from that action, and positives values have the opposite effect. One of the known obstacles in training RL agents is the delayed reward problem. This is usually addressed by introducing auxiliary concepts such as value functions which act as a guiding signal before the arrival of the actual reward.

All these concepts can be formalized as classical sequential decision problems using the mathematically idealized models called the Markov Decision Process (MDP) [10]. RL algorithms try to solve MDPs by finding or approximating a policy. A policy, π, is a set of decision rules acquired by the agent that helps it in making proper decisions. π can be formalized as a function that maps states to actions in deterministic or action probabilities in stochastic environments; $\pi : S \rightarrow P(A)$

[48].

An optimal policy is determined in at least two ways. The first method is maximizing a state-action value ($Q^{\pi}(s,a)$) or state value function ($V^{\pi}(s)$) which returns the quality of a state or action-state under a particular policy. Secondly, the policy can also found by directly optimizing the policy itself. The former algorithms are foundations of many RL methods, especially the temporal difference (TD) methods and actor-critic approaches, and the latter forms the policy gradient class of RL methods [10].

Q and V are defined recursively. The recursiveness shows the dependence of the two on the discounted quality of the next states. They are mathematically expressed in Equation 2.2 and 2.3.

$$Q(s_t, a_t) \leftarrow \mathbb{E}[r + \gamma Q(s_{t+1}, a_{t+1})] \tag{2.2}$$

$$V(s_t) \leftarrow \mathbb{E}[r + \gamma V(s_{t+1})] \tag{2.3}$$

where s represents the state, a_t and a_{t+1} are current and future actions, respectively. r is the immediate reward obtained by the performed action, and γ is the discount factor for future values.

An essential distinction in RL is two different types of algorithms, on-policy and off-policy methods. On-policy methods such as REINFORCE use the current information available to the agent to update the policy, and that policy also employed to generate the agent's actions. In contrast, in off-policy algorithms like Q-learning and DQN not only past data is considered, but also two separate policies may exist, the one that is being updated and the policy that is doing the actions. In cases like that the two policies will merge after some iterations with delay [10].

Another important class of RL algorithms is actor-critics. The actor-critic paradigm emerged as a result of combining policy gradient methods such as REINFORCE (Sec2.3.7) and value-based methods like Q-learning (Sec2.3.2). Actor-critics often achieve higher performance since they aggregate the advantage of both approaches. These methods can be complicated, but the general idea is that the value-based method (the critic) returns some feedback about the performance of the policy method (the actor) and hence helps it to converge faster [10].

2.3.2 Q-learning

There are two fundamental methods in addressing the MDP-type problems, Dynamic Programming (DP) and Monte-Carlo (MC) methods. DP is a robust algorithmic technique that is used to solve a vast spectrum of problems. In RL, however, it is mainly used when the agent knows the model of the environment. This class of algorithms is also known as model-based. MC, on the other hand, is the opposite of DP, and it has no or minimal prior assumption on the way the environment behaves. Hence, MC is called a model-free method. DP and MC have their strengths and weaknesses, which motivated researchers to synthesize the two into a larger class of model-free methods called Temporal Difference (TD) [10].

TD absorbs positive aspects of the two and forms a stronger approach. The central idea of the TD method is utilizing the previous state to update the current one. This notion is mathematically expressed using Q and V functions in Equation 2.4 and 2.5. Similar to Equation 2.2 and 2.3, s, a, r, γ and V are the state, the action, the reward, the discount factor and the value function, respectively. α represents the learning rate in TD algorithms. The TD update mechanism revolves around computing the difference between the actual reward and the expectation of it, which can work as a loss function for the agent's success.

$$V(s_t) \leftarrow V(s_t) + \alpha(r + \gamma V(s_{t+1}) - V(s_t)) \tag{2.4}$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + (r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.5}$$

Q-learning can be derived from Equation 2.5, as shown in Equation 2.6. In Q-learning, after the random initialization of Q-values, the agent starts taking actions. The action is chosen by a random process called ϵ-greedy. ϵ-greedy balances the exploration-exploitation to facilitate a better learning outcome. This process continues until convergence [49].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + (r + \gamma max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2.6}$$

2.3.3 DQN

As noted before, the Deep Q-Network as a combination of deep neural networks and Q-learning was defined in the context of games [4], with its input being a sequence of images from a game. The DQN paper proposed a five-layer neural network architecture. The input to the network is the environment state, and its output is the set of Q-values for each action.

The first convolutional network interacts with the stream of images coming from the screen. The convolutional networks are mostly in charge of detecting objects in the stream of images. To enhance the capability of the algorithm, three convolutional networks have been added to the implementation. The result is processed by two fully connected networks. The other end of the network controls the agent actions.

After creating the Q network, the DQN algorithm selects an action using ϵ-greedy policy. Under this policy, the chosen actions are either the best actions with the probability of $1-\epsilon$ which maximize the Q-value ($a \leftarrow \arg\max_{a'}(Q(s,a';\theta))$), or they are random actions with probability of ϵ. To avoid infinite exploration, ϵ will decay over time.

The actions, states and the rewards will be saved for more efficient learning. This process is called experience replay. The space for saving the memory is called experience replay buffer which stores all the agent's experiences (actions, rewards, and states) for training the network.

The loss function calculates the difference between the target and predicted values, as in Equation 2.7, which originates in the Q-learning update mechanism as laid out in Equation 2.6. DQN optimizes the loss function with respect to the neural network parameters θ, using gradient descent. These steps will be repeated until the convergence of an optimal sequence of actions. The DQN pseudocode is in algorithm 2.

$$L(\theta) \leftarrow (r + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta))^2 \tag{2.7}$$

2.3.4 Enhanced DQNs

The DQN algorithm was relatively successful is demonstrating some degree of general intelligence, especially in the context of Atari games. However, it suffers from specific issues such as overoptimism in estimating the Q value [11], its inability to use relevant memories from the past [50],

and the narrowness and locality of Q values [12]. These problems led to three different variations of the DQN algorithm, which addressed and solved them.

In order to solve the overestimation of the Q values problem, the double DQN algorithm has emerged. The issue can be resolved by proposing two parallel separate Q networks, each learning independently. This architecture resembles the actor-critic paradigm (Sec 2.3.6), as learning is enhanced by utilizing two separate networks.

The matter can be simplified by keeping every part of the DQN algorithm, except that two neural networks are initialized instead of one, and some modifications are made to the DQN evaluations. Equation 2.8 shows the Double DQN update central update mechanism [11].

$$y_i^{\text{DoubleDQN}} \leftarrow r + \gamma Q(s, \arg\max Q(s, a; \theta^-); \theta') \tag{2.8}$$

where, as before, r, s, a, γ, and Q represent the reward, the state, the action, the discount factor, and the state-action value, respectively. The y_i value represents the target of the double DQN algorithm. θ^- signifies the target network parameters, whereas $\theta^{'}$ shows the current network.

The original DQN also did not incorporate any intelligence in its replay buffer sampling process: sampling selection was based on the uniform distribution. There are certainly different methods of prioritizing, but one of the most effective ones is to use the Temporal-difference (TD) error. This process values the highest TD errors the most, which means that whenever the agent is surprised by the environment, it assigns higher probabilities for more learning and memorizing. This method also aligns with animal and human memory functions [50].

2.3.5 Deep Deterministic Policy Gradient (DDPG)

The DQN algorithm and its derivatives were performing well in certain situations, but they were suffering from many shortcomings. The most important was their inability to tackle continuous action environments. The main reason for this is the curse of dimensionality present in these environments, which makes them slow even after discretization. Thus, the Deep Deterministic Policy Gradient (DDPG) was proposed in 2016 [5], which was an enhanced version of the original DPG paper [51]. DDPG is inherently model-free and actor-critic (Sec 2.3.1) [5].

DDPG borrowed the main algorithm from the DPG paper, but in architecture and learning

mechanism, it is following the footsteps of the DQN algorithm, namely using experience replay and the target network. The actor-critic nature of DDPG requires the initialization of two interacting neural networks. In each iteration, the actor-network generates an action according to the current policy and Ornstein-Uhlenbeck random process as its exploration noise. After performing the chosen action, observing the new state and receiving the reward, the critic network tries to minimize the loss function, which also influences the actor policy. The final step is to update the target network using the τ hyperparameter, which controls the amount of update. Instead of the delayed update method in DQN, DDPG instantly updates the target function using τ. The DDPG pseudocode is available in algorithm 3.

2.3.6 Asynchronous Advantage Actor-Critic (A3C)

Asynchronous methods were proposed for various reasons, but the most important ones are their ability to be distributed among different systems and devices and their lightweight architecture [6]. One of the biggest obstacles of offline learning algorithms such as DQN and DDPG is the memory that they consume to keep their replay memory. This issue becomes even more problematic when the complexity of the environment increases, and therefore, more and more space is required to preserve experiences.

In algorithms such as Asynchronous Advantage Actor-Critic (A3C), the lack of memory is balanced out by multiple parallel agents that learn and update the central policy in unison and thus collectively creating an implicit replay memory. Also, in DQN the role of replay memory is to decrease the correlation between current states. In asynchronous algorithms, several agents interacting with the environment and aggregating the central network has the same effect. A3C pseudocode details are in algorithm 4.

2.3.7 REINFORCE

Policy gradient (PG) methods, as a class of RL algorithms, try to learn the right policy by directly updating the model parameters instead of using a concept such as value (V function) or action-value functions (Q values and Q function) that assist the model in learning. Non-policy gradient algorithms are known to be more expensive or impossible in particular settings such as continuous

environments and infinite action spaces. PG methods, on the other hand, may need more data or iteration of training to achieve an acceptable level of performance.

The simplest PG algorithms are the vanilla policy gradient method or the REINFORCE method. REINFORCE is considered a Monte-Carlo method which tries to gather random returns from possible trajectories and utilize them retrospectively in the learning process, as illustrated in Equation 2.9. The update mechanism of REINFORCE is demonstrated in Equation 2.10.

$$G \leftarrow \sum_{i=t+1}^{T} \gamma^{i-t-1} r_i \tag{2.9}$$

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(a_t|s_t, \theta) \tag{2.10}$$

In Equation 2.10, G is the discounted return; θ represents model parameters that are being optimized; α is the update step which controls the learning rate; a and s are the taken action and the state, respectively; and π is the agent's policy that takes the state and network parameters and generates actions.

One of the issues of the vanilla REINFORCE is its high variance which diminishes its performance. The performance can be improved by subtracting the policy π from a static or dynamic baseline value b. This technique called baselining achieves better results [10, 14] and is used extensively in different areas such as the image captioning methods [2, 24]. REINFORCE is the foundation of later PG algorithms such as Trust Region Policy Optimization [15] and Proximal Policy Optimization [3].

2.3.8 Trust-Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO) is another policy gradient algorithm. As mentioned, the vanilla PG algorithms are in danger of policy divergence (Sec 2.3.7). A baselined PG can achieve better performance, but the other issue is that the old policy does not inform the step size of the algorithm, and thus, it might fail in finding the optimal policy. The solution is to add trust-region constraints to the optimization problem. The trust-region constraint will guarantee against the danger of moving too far. TRPO is the result of all these considerations. TRPO can be reduced to utilizing conjugate gradient optimization algorithm in solving the constraint optimization problem

in Equation 2.11 [15].

$$\begin{aligned} &\text{maximize } [L_{\theta_{\text{old}}}(\theta)] \\ &\text{subject to } D_{KL}^{\max}(\theta_{old}, \theta) < \delta \end{aligned} \tag{2.11}$$

where L is the loss function similar to REINFORCE; θ is the network parameters; D is the Kullback–Leibler (KL) divergence that measures the distance between the two policies; and δ is the hyperparameter that controls the trust-region or the distance between the two consecutive policies.

2.3.9 Proximal Policy Optimization (PPO)

TRPO introduced a robust solution for stabilizing policy gradient RL algorithms. TRPO stability stems from its optimization part. This is especially important in reinforcement learning because of the influence of an agent's actions on its later observations. However, TRPO is suffering from complexity in implementation and slow execution. The delayed convergence is attributed to two factors, the constraint in the optimization and the optimization algorithm (conjugate gradient). These two factors are both computationally complex.

Proximal Policy Optimization (PPO) emerged as the successor of TRPO, inheriting the main idea, but with a much simpler way of dealing with the trust region problem using only the first-order optimization. PPO's decoupled architecture makes it suitable for scalability and parallelism, which enhances its success in distributed and multi-agent environments. The method in PPO is to replace the complex TRPO loss function with a clipped one. The trust region constraint is embedded in the minimization step of the loss function formula, as shown in Equation 2.12. The actual PPO method has two more components, the value function update, shown in Equation 2.13, and the entropy. The full method is shown in Equation 2.14. PPO outperformed most deep RL algorithms in continuous spaces [3].

$$\begin{aligned} L^{CLIP}(\theta) \leftarrow \mathbb{E}\Big[\min(r(\theta)\hat{A}, clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}\Big], \\ r(\theta) \leftarrow \frac{\pi_\theta}{\pi_{\theta_{old}}} \end{aligned} \tag{2.12}$$

$$L^{VF}(\theta) \leftarrow (V_\theta(s) - V^{targ})^2 \tag{2.13}$$

$$L^{PPO}(\theta) \leftarrow \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta(s)]] \tag{2.14}$$

where π and π_{old} are the current policy and the old policy, respectively. $r(\theta)$ is the ratio of the two policies, and it has been introduced for readability purposes. $\hat{A}$ represents the advantage function which is calculated after the Monte-Carlo rollout and evaluates the quality of the taken action. ϵ is the clipping value, set to a number between 0 and 1, and controls the policy update. $L^{VF}(\theta)$ is the regression-like update of the value function used in the PPO loss function, which updates the value function prediction. They all combined in Equation 2.14. The $S[\pi_\theta(s)]$ terms quantifies the entropy from the policy to ensure enough exploration by the algorithm.

2.4 RL Approaches in Text Generation and Captioning

Daumé III et al. [22] proposed the first work that pointed out how to effectively use reinforcement learning to solve complex structured classification problems, introducing a 'meta-algorithm' called SEARN. Its main idea was to combine search and learning to reduce initial assumptions about the structure of the final model. What SEARN was proposing was to expose the model to its predictions, and thus producing a more informed series of actions afterward. It achieved state-of-the-art performance in text summarization at the time of its publication [2, 22]. The MIXER algorithm [2] picked up this idea and appropriated it from text summarization to text generation. Instead of cross-entropy, they used the REINFORCE algorithm [14] in their method. Their technique addressed the differentiation problem and the exposure bias that most metric-based approaches were trying to overcome.

The first issue arises when a non-differentiable function (for instance, all the NLP metrics discussed in Sec 2.2.3) used to assess the performance and the optimization algorithm alone cannot determine the gradient and hence carry out the backpropagation. Exposure bias occurs when the error aggregates as a result of the model never having been exposed to its outcome. These two were the main reasons that held back metric feedback systems in sequence generation and its derivatives such as image captioning [2].

In this context, the REINFORCE as a policy gradient RL algorithm behaves like an agent. The agent parameters determine the agent policy which chooses the right action or words during the word generation. The state is a combination of predicted words and a context vector representing an environment such as an image. As mentioned before, the reward is coming from an NLP metric or possibly another model that absorbed evaluative capabilities of metrics. Thus, all the elements of a Markov Decision Process (Sec 2.3.1) are present — namely, (S, A, P_a, R_a) — and the problem can be tackled with RL techniques [10]. The MIXER loss function is the negative of expected reward, as shown in equation 2.15 [2] where $w_{1..T}$ represents the generated words, and r is the reward assigned to it. p is the probability of the next produced word that corresponds to the notion of policy (π) in reinforcement learning when generating a word is interpreted as taking action.

$$L(\theta) \leftarrow -\sum_{w_{1..T}} p_\theta(w_{1..T}) r(w_{1..T}) \tag{2.15}$$

Mixed Incremental Cross-Entropy Reinforce (MIXER) method introduced the idea of incremental learning and a mixed loss function, which together can handle large action spaces. Rather than a random initialization, MIXER starts with a cross-entropy pre-training which reduces the perplexity factor of the algorithm. MIXER also uses the annealing schedule method to stabilize the generated output. It used two types of RNNs, LSTMs and standard Elman RNNs. In both cases, the usual recurrent processing takes place, and the algorithm learns to maximize the distribution over the next word given the currently generated ones, as in Equations 2.16 and 2.17 [2]. ϕ is the RNN function that processes a word and the output of the previous one.

$$\mathbf{h}_{t+1} \leftarrow \phi_\theta(w, \mathbf{h}_t, c_t) \tag{2.16}$$

$$w_{t+1} \sim p_\theta(w|w_t, \mathbf{h}_{t+1}) = p(w|w_t, \phi_\theta(w_t, \mathbf{h}_t, c_t)) \tag{2.17}$$

where w is the network's weights, $\mathbf{h}_t$ is the hidden state of the RNN, and c_t is an optional context vector. ϕ represents the hyperbolic tangent activation function which transforms the output. p is the probability of the next word in the sequence, and it is very close to the concept of policy in RL.

The architecture of MIXER was similar to earlier encoder-decoder methods and set the architecture that all the descendant methods followed. Figure 2.5 illustrates its schema.

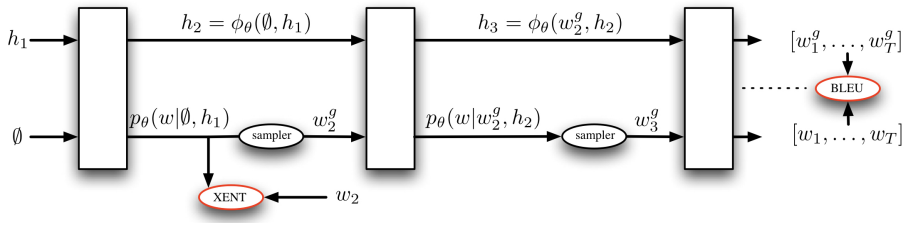

FIGURE 2.5: The MIXER Architecture [2]

2.4.1 Self-Critical Sequence Training

The Self-Critical Sequence Training (SCST) method emerged after the initial developments in end-to-end approaches such as Teacher-Forcing and Professor-Forcing. These initial methods were mainly utilizing the cross-entropy as the cost function, which led to exposure bias. SCST follows the REINFORCE-style algorithms in image captioning, such as MIXER. The rationale is the same as before, to overcome the exposure bias and the differentiation problem. The difference with previous methods is the moving baseline in REINFORCE that controls the variance of the process. SCST uses its test-time output to normalize the rewards it receives during the training time [24]. The SCST training reward is coming from the CIDEr score.

Two types of captioning models can be used in the image captioning context, the FC model and the Attention models. The FC model is static and merely encodes the image using a CNN. The CNN could be trained beforehand like a VGG or ResNet. The extracted features will be embedded through a dense layer. The vocabulary that is expressed with one-hot encoding is also embedded using a similar linear layer. These inputs will be given to an LSTM layer for the final caption generation. The output is a distribution over the next word after the final softmax layer. The loss function for backpropagation is the SCST algorithm. The attention method follows the same mechanism. The only difference is the dynamic re-weighting of the CNN features to concentrate on a different section of the image every step [24].

After each iteration of caption generation for one instance or a batch of them, we update our network using the loss function. As mentioned, cross-entropy is the traditional method for this

purpose, and it is still used as a pre-training step, but after the initial stages, we apply a modified version of a baselined REINFORCE algorithm, as shown in Equation 2.18. The reward in this equation is calculated by subtracting the reward received by the current model from the training reward, which has normalizing effects, as shown in Equation 2.19.

$$L(\theta) \leftarrow -\mathbb{E}_{w^s \ p_\theta}[r(w^s)\log p_\theta(w^s))] \tag{2.18}$$

$$R \leftarrow r(w^r) - r(\hat{w}) \tag{2.19}$$

SCST significantly outperformed its predecessor algorithm, the MIXER method, and set the ground for subsequent advancements. [38] used the same mechanism with an enhanced reward, and [52] combined it with an adversarial architecture. Although other methods were proposed since then, SCST remains the core the state-of-the-art. This SCST architecture is thus the one we use to explore the application of other RL methods.

2.5 Conclusion

In this chapter, we reviewed the challenging problem of image captioning. We explained deep neural networks and their novel solutions for the captioning problem [20, 24, 32]. RL, as a separate paradigm of machine learning, was also utilized in some parts of image captioning [2, 24]. REINFORCE [14] or its other varieties were the methods used in the literature. Although REINFORCE has been a successful method, recent advancements such as PPO [3] augment it with better converging mechanisms. Modern RL methods showed their strengths in robotics and video games [3, 4, 15, 53], so it is justified to move from REINFORCE towards one of these more modern methods. The review of this chapter shows that the PPO approach is both structurally similar to REINFORCE, as a policy gradient method, and one that produces state-of-the-art results in games applications, and so is a suitable candidate. SCST [24] is one of the state-of-the-art methods that set the context for more recent image captioning techniques such as [38, 52] and they all apply REINFORCE in their training phase. Our proposed method is therefore built on SCST and PPO but the RL aspect derives from our chosen model.

3

OpenAI Gym Experiments

In this chapter, we begin our experiments with reinforcement learning algorithms, focussing on REINFORCE and PPO as motivated by the previous chapter, to identify success factors in them and investigate the way they operate. More specifically, we would like to study the inner functions of PPO to facilitate the process of its incorporation in image captioning. These experiments are conducted in the context of one of the OpenAI Gym's environments. The chosen environment, the LunarLander, is among the games that are closer to image captioning in terms of discrete actions and vectorized state space.

The outstanding achievement of the PPO algorithm is shown in different areas such as robotics and video games, as benchmarked by [3] (graphs of performance reproduced in Appendix B). To test the performance of the Proximal Policy Optimization (PPO) algorithm, the effects of its three-term loss components, and its performance in a combined form with the REINFORCE algorithm (Equation 3.1) [14], we proposed and tested three separate methods.

3.1 Introduction

Reinforcement learning, as a branch of machine learning, has lots of shared properties with other topics, more specifically, with supervised learning. However, according to its definition as an interactive agent within an environment, makes its assessment complicated. RL algorithms can be directly applied to mechanical or physical systems such as a robot or drone as a controlling mechanism. This process, however, is subject to many problems such as high time-consumption, high cost and non-repeatability. These issues are more critical when we try to compare the performance of one algorithm against others. Thus, the RL community standardly uses common simulated environments to evaluate algorithms.

3.2 The Experimental Environment: The Lunar Lander V2

In recent years, there have been several proposed benchmark environments for reinforcement learning algorithms such as Arcade Learning Environment (ALE) [54] and RLLib benchmark [55]. OpenAI Gym combines these older environments and more in a unified and intuitive interface that reflects the MDP structure of RL problems. In other words, every environment, regardless of its underlying design, starts from an arbitrary zero state, takes the agent's action, and goes to the next state while generating the relevant reward. This process continues until the environment reaches the final state.

There are five general classes of environments in OpenAI Gym, namely, the classic control and toy texts, algorithmic, Atari, board games, and robotics. In our experiments, for reasons of similarity to text generation problems, we focused on one of the 2D robotic controlling environments, the Lunar Lander.

The Lunar Lander environment, as depicted in Figure 3.1, is an example of a classic controlling problem which is built on top of the Box2D physics engine [56]. The agent role is to guide the engines to land at the landing pad with zero speed. The landing pad is always at zero coordinates, and the right landing earns a reward of between 100 to 140 for the agent [57].

The episode finishes after crashing or receiving -100 or 100 points. There are also smaller

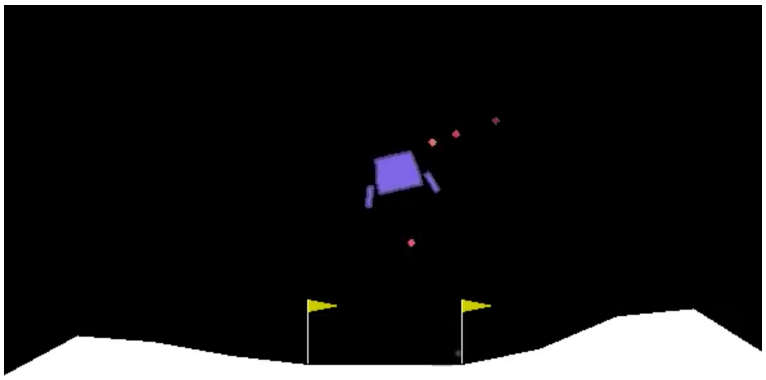

FIGURE 3.1: The LunarLander-v2 Environment

negative and positive rewards encouraging the agent for better and more efficient performance such as ten extra points for contacting each leg to the landing pad, and -0.3 for firing the main engine. The action space is discrete and consists of four actions: doing nothing, utilizing the left engine, utilizing the right engine, and firing the main engine [57].

3.3 Experimental Setup and Results

As mentioned, the PPO algorithm has a three-term loss function [3] inspired by the Trust Region algorithm [15] and the Policy Gradient method. Each component has a specific role in the optimization process with different impact, which we will investigate in this experiment. Restating them from Chapter 2, they are the clipped policy gradient (Equation 3.2), the value function update (Equation 3.3), and the entropy term.

The core component of PPO method is the clipped loss function ($L^{CLIP}(\theta)$, Equation 3.2). The value function ($L^{VF}(\theta)$, Equation 3.3) and the third part of the main PPO loss function (Equation 3.4), the entropy, are secondary. The reason is that the first term embodies the policy gradient section of the method, making the core decisions. Thus, it is plausible to think about other alternatives of PPO with lesser complexity, removing the second, third or both to examine

the role of each in the final output.

$$L^{REINFORCE}(\theta) \leftarrow \sum \log \pi_\theta(a_t|s_t) G_t \tag{3.1}$$

$$\begin{aligned} L^{CLIP}(\theta) \leftarrow \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \Big], \\ r_t(\theta) \leftarrow \frac{\pi_\theta}{\pi_{\theta_{old}}} \end{aligned} \tag{3.2}$$

$$L^{VF}(\theta) \leftarrow (V_\theta(s_t) - V_t^{targ})^2 \tag{3.3}$$

$$L^{PPO}(\theta) \leftarrow \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta(s)]] \tag{3.4}$$

Our experiments here differ from those of [3] in that we aim to identify the relative importance of the three components of PPO, and discover whether PPO and REINFORCE can be usefully combined. The combination of PPO and REINFORCE is a feasible solution due to their high bias and variance, respectively. The combined method may cancel each components' shortcomings and make a stronger solution.

3.3.1 Full PPO and the REINFORCE Algorithm

This method considers the REINFORCE algorithm and the whole PPO algorithm in an interpolated manner, as shown in Equation 3.5.

$$L^{PPO}(\theta) \leftarrow \lambda_{ppo} E[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta(s_t)]] + \lambda_{REINFORCE} E[L^{REINFORCE}(\theta)] \tag{3.5}$$

3.3.2 PPO with the Clipped and Value Function and REINFORCE

The method, represented by Equation 3.6, is similar to Equation 3.5 but with the elimination of the entropy part, which is in charge of introducing exploration to the equation; it thus allows an analysis of the importance of using entropy in the PPO method.

$$L^{PPO}(\theta) \leftarrow \lambda_{ppo} E[L^{CLIP}(\theta) - c_1 L^{VF}(\theta)] + \lambda_{REINFORCE} E[L^{REINFORCE}(\theta)] \tag{3.6}$$

3.3.3 Clipped-only PPO and REINFORCE

Finally, the third variety is the clipped PPO surrogate objective of Equation 3.2 and its combination with the REINFORCE algorithm 3.7; the clipped PPO surrogate is the component of the full PPO that is structurally most similar to the REINFORCE objective. This method is thus merely mixing two types of REINFORCE methods with high and low variance.

$$L^{PPO}(\theta) \leftarrow \lambda_{ppo} E[L^{CLIP}(\theta)] + \lambda_{REINFORCE} E[L^{REINFORCE}(\theta)] \tag{3.7}$$

3.4 Experiments

We tested five values of lambda for each proposed method as illustrated in Table 3.1. Then, we plotted them and compared the results. The RL community usually assesses the experiments in terms of two criteria, the reward gained and the speed of learning. Both are captured in these diagrams. By convention, the y-axis represents the reward gained by the RL agent and the x-axis is the epoch number. In these experiments, every integer point on the x-axis outlined the average rewards earned in every 50 consecutive episodes to enhance the readability of our results.

Algorithm / λs	PPO-clipped+REINF.	ISO PPO-Value+REINF.	PPO-Full+REINF.
$\lambda_{ppo}, \lambda_{REINFORCE}$	(0,1)	(0,1)	(0,1)
$\lambda_{ppo}, \lambda_{REINFORCE}$	(0.25,0.75)	(0.25,0.75)	(0.25,0.75)
$\lambda_{ppo}, \lambda_{REINFORCE}$	(0.5,0.5)	(0.5,0.5)	(0.5,0.5)
$\lambda_{ppo}, \lambda_{REINFORCE}$	(0.75,0.25)	(0.75,0.25)	(0.75,0.25)
$\lambda_{ppo}, \lambda_{REINFORCE}$	(1,0)	(1,0)	(1,0)
Charts	3.2	3.3	3.4

TABLE 3.1: The LunarLander experiment proportion of Loss Functions

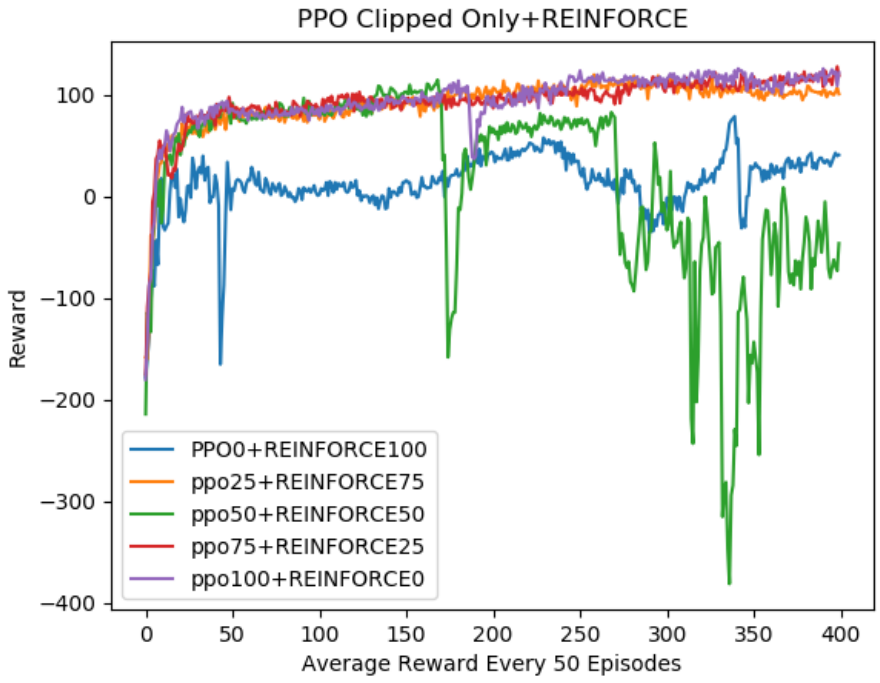

FIGURE 3.2: PPO-Clipped+REINFORCE Method (Equation 3.7)

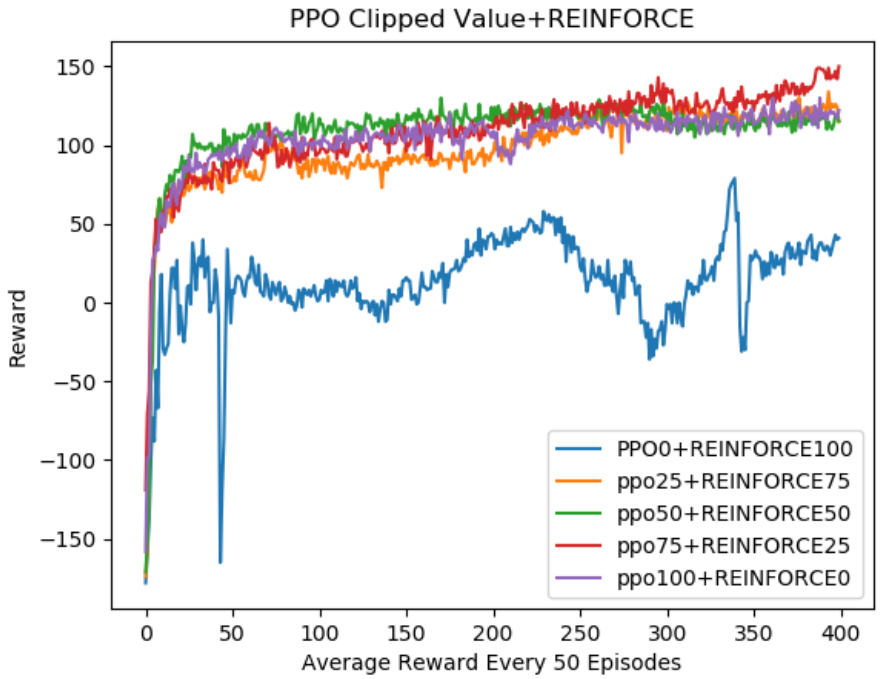

Figure 3.3: PPO-Value+REINFORCE Method (Equation 3.6)

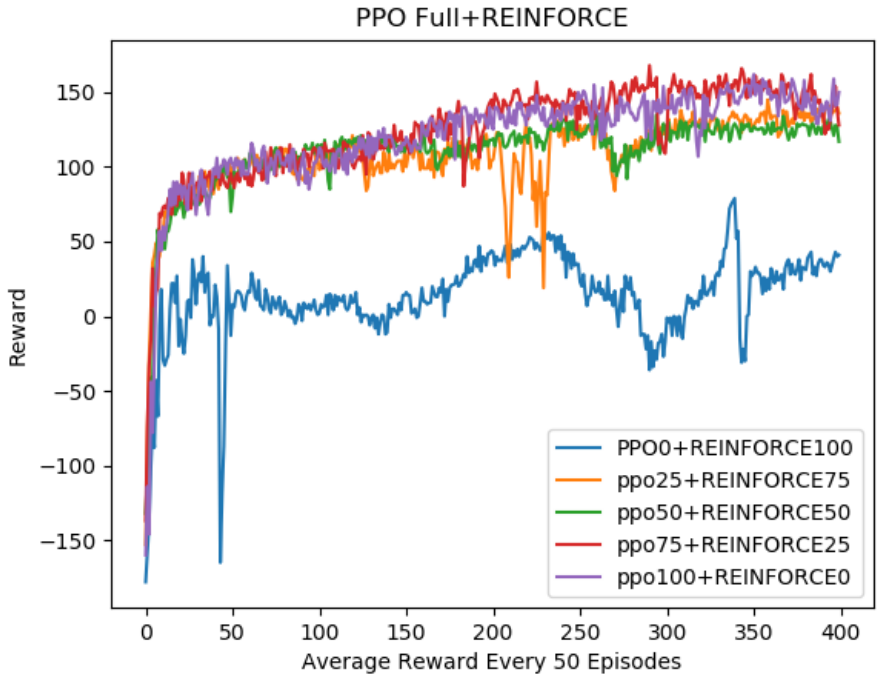

Figure 3.4: PPO-Full+REINFORCE Method (Equation 3.5)

3.5 Discussion and Conclusion

Looking at Figures 3.2 to 3.4, REINFORCE alone successfully trained the agent but with significantly lower performance than PPO or other varieties of the interpolated PPO and REINFORCE (except the 50 PPO-Clipped + 50 REINFORCE version). When we compare the full PPO algorithm (Figure 3.4), PPO first and second term (PPO-Clipped Value) (Figure 3.3), and PPO first term only (PPO-Clipped) (Figure 3.2), we observe that all are successful in training agent with no different significant rate. This fact indicates that the clipped policy gradient mechanism in PPO, as illustrated in Equation 3.2, is the main factor in the learning process, and it can be possible to be used alone when the environment has similar properties like the LunarLander such as discrete action space.

The interpolated version, however, had a slightly better performance in some cases than the PPO-only version, especially when it comes to the Clipped-only method (Figure 3.2). These classes of algorithms can become effective when we run such algorithms for a longer time or when the bias-variance trade-off is destabilizing the PPO-only due to its high bias.

Our experiments show that the main component of the Proximal Policy algorithm is the first term. The rest improves and smoothes the outcome, especially in continuous environments [3], which are not the subject of this study. Hence, for simplicity purposes, our proposed methods will benefit from the clipped ratio idea of the PPO algorithm.

Given that most of PPO varieties outperformed the vanilla REINFORCE algorithm, we are expecting to see the same pattern in other places that the REINFORCE algorithm has been used, such as the text generation task in image captioning. In the next chapter, we propose our method based on our findings in these initial experiments.

4

Proposed Method for Image Captioning

PPO has outperformed other state-of-the-art algorithms as it was shown in Appendix B. PPO is also significantly better than the REINFORCE family of algorithms in the game and physical controlling context, as we demonstrated in Chapter 3. We learned that PPO first-term clipping mechanism is the essential component of the method, and that is directly inter-substitutable with PPO. Chapter 3 also uncovered the possibility of combining the PPO or one of its varieties and REINFORCE to achieve better performance.

In this chapter, we lay out some augmentations on top of the state-of-the-art reinforcement learning algorithms used in image captioning and sequence generation. The proposed additions, in an autoencoder context, are network agnostic, and thus, in theory, are compatible with any existing or future neural networks used for this task. To do this, we analyze the main characteristics of current methods in image captioning and reinforcement learning and draw on the strengths of modern algorithms from the deep reinforcement learning community.

4.1 Introduction

There are two main issues in older methods such as cross-entropy that made researchers apply reinforcement learning instead, namely, exposure bias and the differentiation problem. The first methods, such as MIXER, were using an autoencoder network architecture and a simple reinforcement learning method such as REINFORCE. Even though REINFORCE addressed the aforementioned problems, it still suffered from high variance, which can lead to slow convergence, as explained in Section 2.3.7. Thus, the REINFORCE with a baseline was proposed, which has the same expectation, but with a smaller variance.

The REINFORCE with baseline itself can also become unstable as the context of the text or image is not considered in calculating the baseline. To correct this, the Self-Critical Sequence Training (SCST) proposed the utilization of its output signal as the normalizing factor of its rewards. As a result, SCST pushed up samples that surpass its current performance, and the negative ones will be suppressed. This architecture is comparable to the actor-critic methods in reinforcement learning since the reward estimation influences the policy network too. Our approach to modifying the SCST algorithm keeps this dynamic baseline and changes the core reward component to reflect the improvements of PPO over REINFORCE.

4.2 Proposed Algorithm

In machine learning, the goal is to minimize the loss function, which, in image captioning, used to be the cross-entropy method shown in Equation 4.1. This method tried to maximize the likelihood of the next word.

$$L(\theta) \leftarrow -\sum_{t=1}^{T} \log(p_\theta(w_t|w_{1:t-1})) \tag{4.1}$$

As mentioned, the loss function can be replaced by the requirement to minimize the collected negative reward by a reinforcement learning method as predicated in Equation 4.2. The reward signal is coming from an NLP metric such as CIDEr or BLEU score.

$$L(\theta) \leftarrow -E_{w^s \sim p_\theta}(r(w^s)) \tag{4.2}$$

where $w^s = (w_1^s, \ldots, w_T^s)$ and w_t^s are the sequences of words and word sampled from the model at the time step t, respectively. $r(w^s)$ is the reward given to that sequence. The goal is to minimize the loss function, $L(\theta)$ in order to maximize the earned rewards.

In Policy Gradient approaches with REINFORCE (Sec 2.3.7) such as MIXER, REINFORCE solves the non-differentiation reward problem via its computation formula. As shown in Equation 4.3, the gradient bypasses the reward signal [2].

$$\nabla L(\theta) \leftarrow -E_{w^s \sim p_\theta}(r(w^s) \nabla_\theta \log p_\theta(w)) \tag{4.3}$$

As mentioned in Section 2.3.7, the high variability of this equation can be addressed by the introduction of a baseline (Equation 4.4). Since the baseline is not a function of θ, it will disappear in the calculation of the gradient and only reduces the variance [14].

$$\nabla L(\theta) \leftarrow -E_{w^s \sim p_\theta}[(r(w^s) - b) \nabla_\theta \log p_\theta(w)] \tag{4.4}$$

The SCST method added an informed baseline to the formula from its output, as shown in 4.5. The new baseline represents the expectation of the network of the future reward. This change will help the algorithm to converge faster [24].

$$\nabla L(\theta) \leftarrow -E_{w^s \sim p_\theta}[(r(w^s) - r(w)) \nabla_\theta \log p_\theta(w)] \tag{4.5}$$

Equation 4.5 can be approximated by Equation 4.6 using the chain rule where s_t is representing the input to the softmax layer. Equation 4.6 is at the core of the self-critical algorithm [24]. It consists of two main components, the base-lined reward $(r(w^s) - r(w))$ and the policy (p_θ) that is represented by the neural network.

$$\frac{\partial L(\theta)}{\partial s_t} \leftarrow (r(w^s) - r(w))(p_\theta(w_t|h_t) - 1_{w_t^s}) \tag{4.6}$$

On the other hand, Proximal Policy Optimization [3] does the learning by introducing three component functions as its loss, and it uses the advantage function in the process. The first component is the policy gradient section with a clipping mechanism. The second part is the value

function cost function, and the third is a guarantee for entropy and enough exploration.

$$L_t^{\text{PPO}}(\theta) \leftarrow \mathbb{E}_t[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{VF} + c_2 S[\pi_\theta](s_t)] \tag{4.7}$$

In our method, the focus is on the first surrogate function, the L^{CLIP} as shown in Equation 4.8.

$$\begin{aligned} L_t^{\text{CLIP}}(\theta) \leftarrow \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t),\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \\ \text{such that } r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \end{aligned} \tag{4.8}$$

The clipping, which is a replacement for the constraint in the TRPO algorithm [15] has to control the policy based on its old value to remain in the 1-ϵ and 1+ϵ interval while keeping it computationally simple [3].

As explained, the self-critical method has two components, the baselined reward, and the policy gradient. The base-lined reward acts as the feedback signal for the policy network. The advantage function takes on the role of a controlling mechanism in Proximal Policy Optimization. They both implement two versions of the Monte-Carlo rollouts. Hence, in order to enhance the efficiency of the self-critical method, we propose the clipped self-critical algorithm (CLP-SC), as demonstrated in 4.9.

$$\begin{aligned} L_t^{CLP-SC}(\theta) \leftarrow -\mathbb{E}_t[\min(r_t(\theta)\Delta\hat{R}_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\Delta\hat{R}_t)] \\ \text{such that} \quad r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, \\ \Delta\hat{R}_t = r(w^s) - r(w), \\ 0 < \epsilon < 1 \end{aligned} \tag{4.9}$$

Based on the validation of the central role of L^{clip} in our initial experiments in chapter 3, we take this as the basis of our fundamental method, training the SCST system using this for the loss function.

We also consider two alternatives, given the observation in chapter 3 that REINFORCE and PPO can be combined. Specifically, we consider using CLP-SC as pre-training for the SCST, and interpolating CLP-SC and SCST. We describe these below.

4.2.1 Variant #1: Pre-training with Clipped-Self Critical Algorithm

Our first alternative is treating the clipped self-critical (CLP-SC) as a pre-training step after the initial pre-training with the cross-entropy. This variation of CLP-SC is expressed in algorithm 1.

Algorithm 1: Pre-training with Clipped-Self Critical Algorithm

Pre-training

Pre-train with the cross-entropy method

Determine the pre-training epoch PE

for *all the epochs* **do**

  if *epochs* $< PE$ **then**

    $L(\theta) \leftarrow -\mathbb{E}_t[\min(r_t(\theta)\Delta\hat{R}_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\Delta\hat{R}_t)]$

  else

    $L(\theta) \leftarrow -\mathbb{E}_{w^s \sim p_\theta}[(r(w^s) - b)\log p_\theta(w)]$

  end

  Optimize $\pi_{\theta_{new}} \leftarrow \pi_\theta + \nabla L(\theta)$

end

4.2.2 Variant #2: The Interpolated Clipped-SC with SCST

The final solution that we are going to investigate is the combination of the SCST method and our clipped one. This method is motivated by the experiments we described in Chapter 3.

In the interpolated version shown in Equation 4.10, we give λ coefficients to each loss function, and we test different values for these coefficients.

$$L^{Interpolated}(\theta) \leftarrow \lambda_{CLP-SC} E[L^{CLP-SC}(\theta)] + \lambda_{SCST} E[L^{SCST}(\theta)] \tag{4.10}$$

Our intuition that led to the interpolated method comes back to the fact that the REINFORCE algorithms and its derivatives have the high variance problem. The variance reduces when a static or dynamic baseline is used. These solutions, however, are limited in performance and applying a trust-region style method such as clipped policy used in PPO is a more robust way of fixing it.

5

Experimental Results for Image Captioning

The core of this chapter are the experiments evaluating the fundamental method and its variants as proposed in Chapter 4. As a precursor to that, we first explore the choice of reward to optimize.

5.1 Sources of Reward

The Self-Critical Sequence Training (SCST) paper advised the use of CIDEr score as the source of reward since CIDEr resulted in the best model. The MIXER algorithm, on the other hand, proposed BLEU and ROUGE-2 for its feedback system. To best of our knowledge, a synthesis reward system is still unexplored. Thus, to discover the role of each NLP score in the reward given to the SCST method, we arrange the following experiments.

We trained the SCST algorithm using the FC model (Sec 2.4.1) for 60 epochs, and plot the evaluations rendered by all six NLP metrics. As mentioned in Section 2.4.1, FC model is a static

end-to-end network consists of a pre-trained CNN for feature extraction from images and an LSTM to generate words. The first 30 epochs were pre-training phase using the cross-entropy algorithm [24]. Then, we trained our network using CIDEr and evaluate them with all the NLP metrics. We use as a baseline a widely used PyTorch implementation of the self-critical sequence training[1] [38]. The hyperparameter settings and other details of our experiments are given in Appendix D.[2] The results are illustrated in Figure 5.1 to Figure 5.8. The sudden change of trend in the 60th assessment (the equivalent of epoch 30, there are two assessments in every epoch) in these diagrams is an indication of switching from cross-entropy to SCST algorithm.

As it is shown in Figure 5.1 to Figure 5.8, BLEU-1 and CIDEr distinguished and valued SCST more than other metrics, and are more likely to provide these class of algorithms with proper rewards. Thus, we tested a linear composite reward consisting of CIDEr and BLEU-1. Although our experiments were not exhaustive, the four different coefficients of BLEU and CIDEr revealed some patterns of optimality, as shown in Figures 5.10, 5.12, 5.14 and 5.16.

It appears from the diagrams that a mixture of one unit CIDEr and 0.5 of BLEU score provides us with the highest score across all the metrics. As these are very expensive to train, we have not carried out repeated training to assess statistical significance, but we note that the pattern is quite consistent. We therefore use this for our RL-based image captioning algorithms. Following the SCST convention [24], the y-axis represents the NLP score (reward) gained by the method and the x-axis is the iteration number. In these experiments, every integer point on the x-axis outlined the reward earned every 6000 iterations to enhance the readability of our results.

[1] https://github.com/ruotianluo/self-critical.pytorch

[2] For clipping, we used the best ϵ from [3], but note that the performance was not strongly dependent on the value of ϵ.

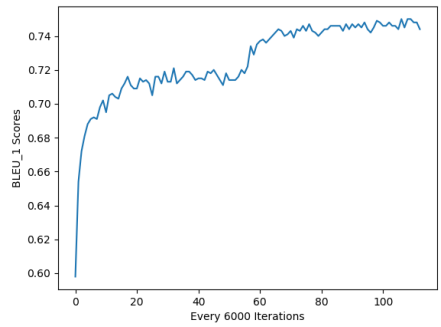

FIGURE 5.1: Evaluated by BLEU 1

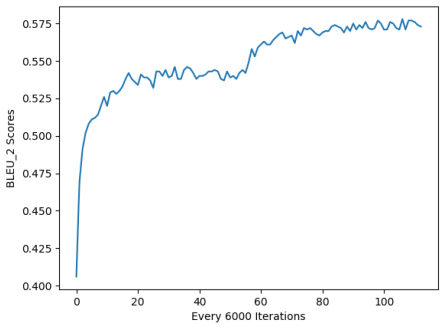

FIGURE 5.2: Evaluated by BLEU 2

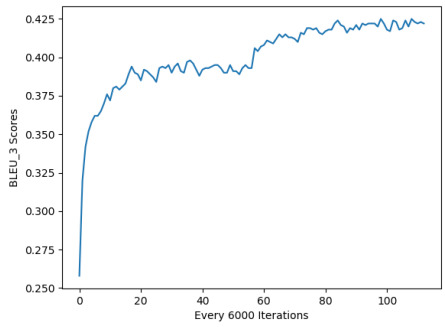

FIGURE 5.3: Evaluated by BLEU 3

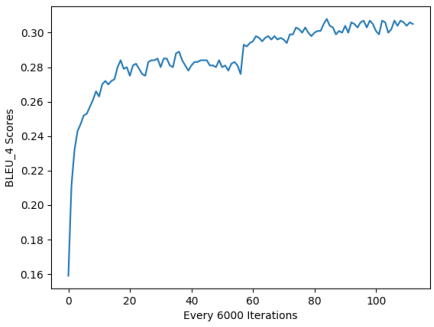

FIGURE 5.4: Evaluated by BLEU 4

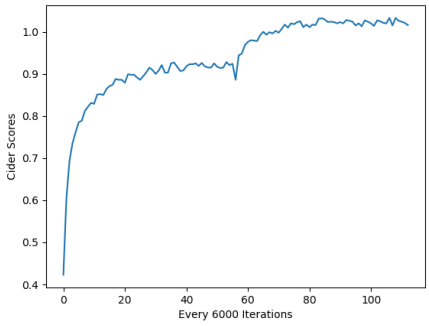

FIGURE 5.5: Evaluated by CIDEr

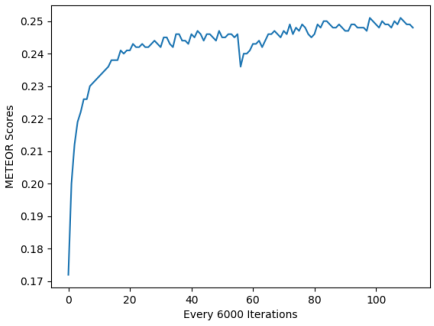

FIGURE 5.6: Evaluated by METEOR

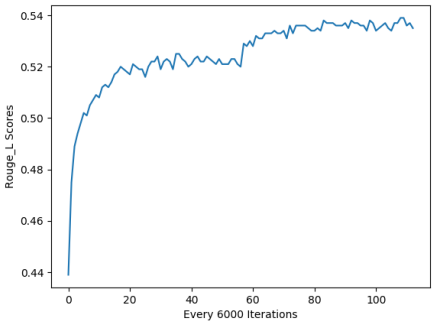

FIGURE 5.7: Evaluated by ROUGE-L

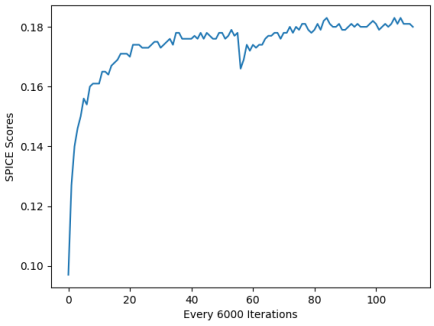

FIGURE 5.8: Evaluated by SPICE

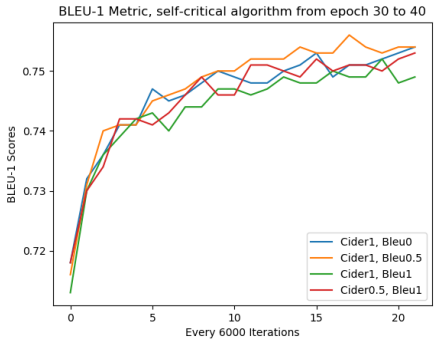

FIGURE 5.9: Evaluated by BLEU 1 (Trained for 10 Epochs)

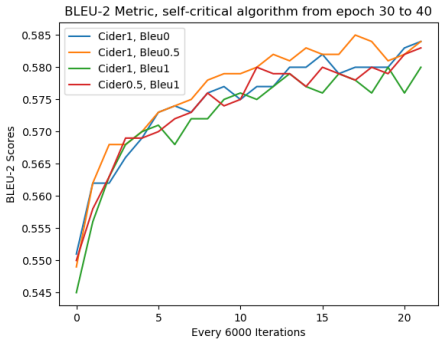

FIGURE 5.10: Evaluated by BLEU 2 (Trained for 10 Epochs)

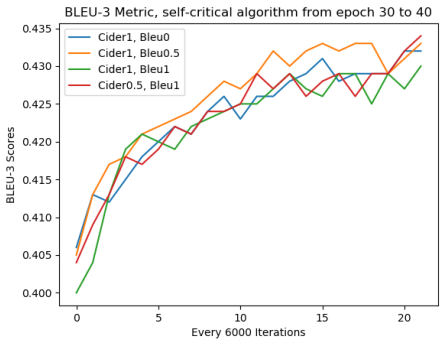

FIGURE 5.11: Evaluated by BLEU 3 (Trained for 10 Epochs)

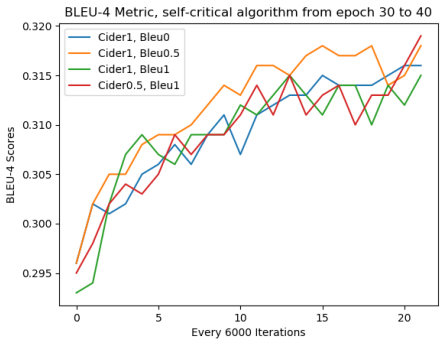

FIGURE 5.12: Evaluated by BLEU 4 (Trained for 10 Epochs)

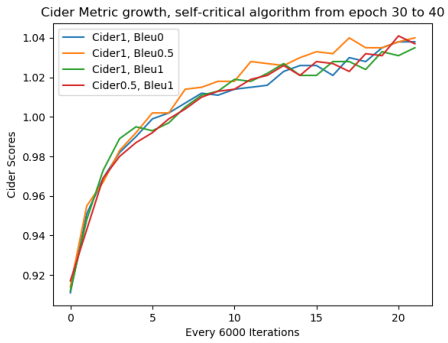

FIGURE 5.13: Evaluated by CIDEr (Trained for 10 Epochs)

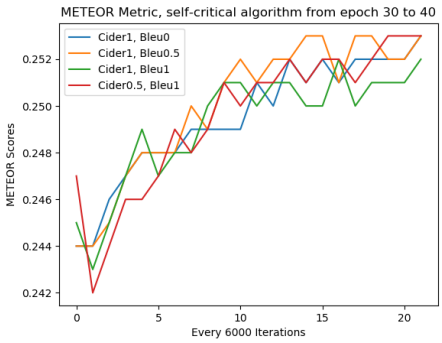

FIGURE 5.14: Evaluated by METEOR (Trained for 10 Epochs)

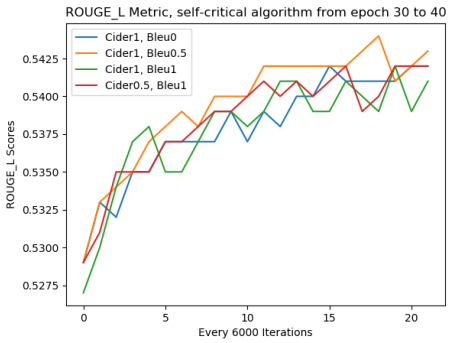

FIGURE 5.15: Evaluated by ROUGE-L (Trained for 10 Epochs)

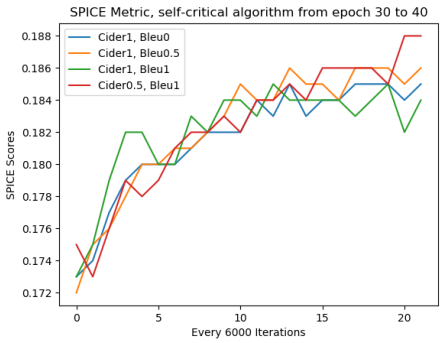

FIGURE 5.16: Evaluated by SPICE (Trained for 10 Epochs)

5.2 Clipped SCST

As explained in Equation 4.8, we proposed the Clipped-SC method, which uses the clipping mechanism of the proximal policy optimization to augment the vanilla policy gradient methods such as REINFORCE. We applied this method to the same experimental setting that is described in Appendix D. The result is shown in Figure 5.17.

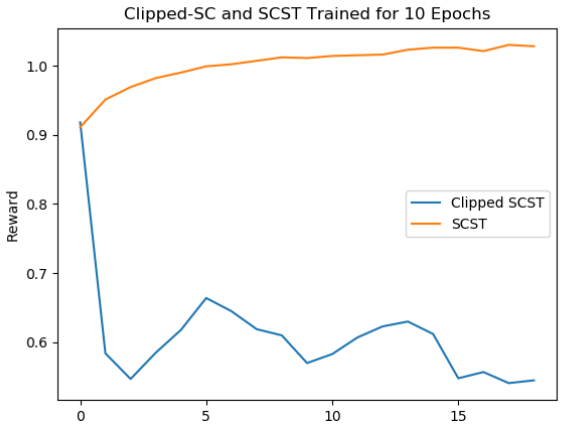

FIGURE 5.17: Clipped-SC vs. SCST (Trained for 10 epochs)

The figure shows a slightly higher reward (1.0 CIDEr + 0.5 BLEU) right at the beginning, but in later iterations, it dropped to a scale below the pre-training phase. One possibility for the Clipped-SC method's poor performance could be that the clipping mechanism introduces a high bias: the success of PPO hinged on bringing more bias to policy gradient methods [3]. While this was clearly useful in the lunar lander experiments of chapter 3, and in the paper that defined PPO and benchmarked it on many problems [3] (see also Appendix B), it may not be applicable here; further, [58] note that bias can have more destructive effects on the convergence of algorithms than the variance. The other factor the may have caused the discrepancy between the Lunar Lander environment, and the image captioning task is the differences in their state space size, which needs more investigation.

We also applied our proposed pre-training algorithm with Clipped-SC and SCST. We dedicated the first 6000 iterations to our Clipped method, and the rest is trained with the vanilla SCST. The result is shown in Figure 5.18. Although initially there was a better performance as we also

observed in our Clipped-SC, it did not lead to a significant improvement.

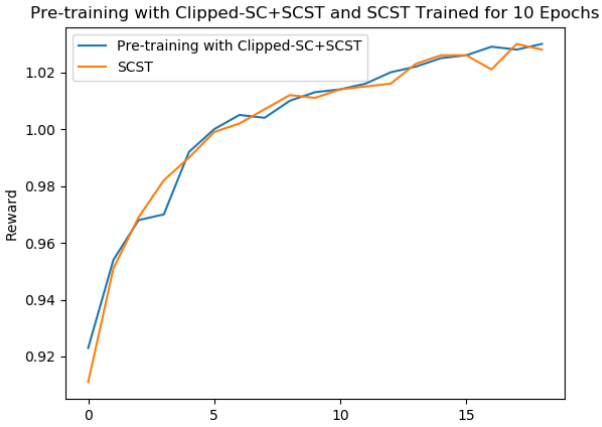

FIGURE 5.18: Pretraining Clipped-SC for 6000 iterations (half epoch)+SCST (Trained for 9.5 epochs) vs. SCST (Trained for 10 epochs)

To investigate this, we carry out experiments with the interpolated algorithm described by Equation 4.10 with the idea that there might be a productive balance between the higher variance of the original SCST and the higher bias of the Clipped-SC approach.

5.3 Interpolated Method

To test the effectiveness of the interpolated method, we examined four possibilities. The λs have this relation, $\lambda_{clipped-sc} + \lambda_{SCST} = 1$. In these experiments to observe the trend, we only trained our models for ten epochs. Figure 5.19 shows different possibilities and their output performances.

Our initial observation shows that there was no significant improvement in utilizing the interpolated algorithm. The method with the largest clipped component ($\lambda_{clipped\_sc} = 0.75$) performed dramatically worse than the others, while those with lower proportions performed similarly to the SCST method. The $\lambda_{SCST} = 0.75$ and $\lambda_{clipped\_sc} = 0.25$ combination shows slightly better performance compared to the original SCST, as shown in Figure 5.20.

To investigate this further, we arranged another experiment for 30 epochs comparing the SCST and the interpolated method with mentioned coefficients. The result is illustrated in Figure 5.21.

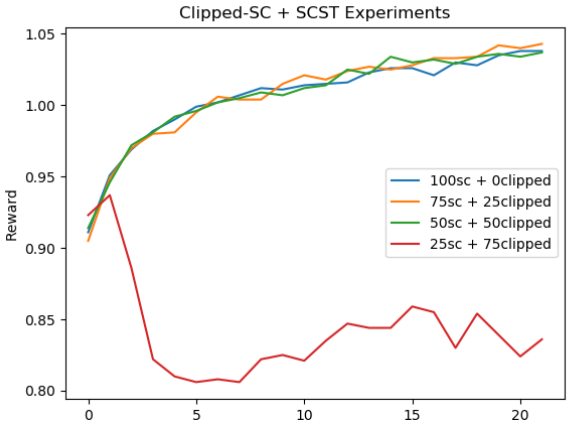

FIGURE 5.19: Interpolated algorithm with different coefficients (Trained for 10 epochs)

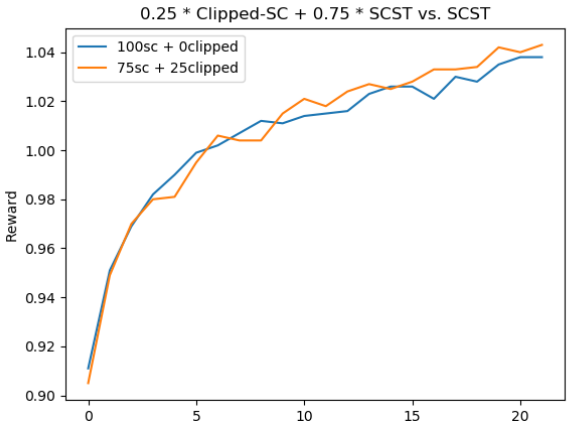

FIGURE 5.20: Interpolated algorithm; SCST vs. 0.25 Clipped-sc + 0.75 SCST (Trained for 10 epochs)

The final experiment showed no noticeable improvement using the interpolated self-critical algorithm with a clipping component. At this point, we considered two possibilities for this discrepancy with our positive results in the OpenAI environments. The first is that the vanilla self-critical method's incorporation of a dynamic baseline — which represents the expectation of the network of the future reward — in some way makes the variance reduction of an approach like PPO unnecessary. Secondly, the nature of image captioning may be different to the extent that it changes the dynamics of algorithms.

We took the first idea and arranged our next experiment, replacing the clipping term in the interpolated method with a constant in one execution and a random number in another one, as

FIGURE 5.21: Interpolated algorithm; SCST vs. 0.25 Clipped-sc + 0.75 SCST (Trained for 20 epochs)

shown in Equation 5.1, where C is representing the constant or random value.

$$L^{Interpolated}(\theta) \leftarrow \lambda_{CLP-SC} C + \lambda_{SCST} E[L^{SCST}(\theta)] \tag{5.1}$$

The constant was one, and the random number was chosen from a normal distribution with $\mu = 0$, $\sigma = 1$. The result is shown in Figure 5.22.

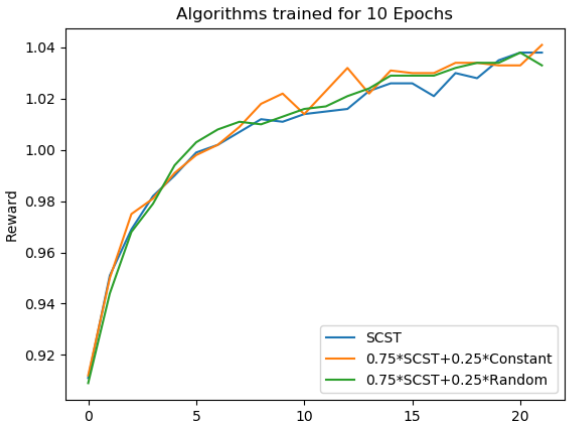

FIGURE 5.22: SCST vs. Constant and Random interpolated (Trained for 10 epochs)

These results show that interpolating noise and random constant values to the self-critical method in fact has similar effects to interpolating the PPO-style clipped component, and possibly better. While the sizes of improvements are small, the interpolated lines are fairly consistently above

the SCST baseline. The introduction of systematic noise may thus warrant further investigation. Overall, however, it seems the self-critical mechanism may be the primary factor in the improvement of the SCST method; and REINFORCE and our clipping method have a less significant role in producing the result. We keep exploring our second hypothesis as our future work.

6

Conclusion and Future Work

The differentiation problem has been one of the obstacles in training text generation models. Modern image captioning methods have been applying reinforcement learning algorithms to address this issue. Most of the proposed state-of-the-art algorithms were focused on earlier methods such as REINFORCE. In this thesis, we reviewed modern deep reinforcement learning methods and examined some possible improvements for the image captioning problem.

After reviewing the literature, it became apparent to us that the trust-region family of algorithms have achieved higher performance due to their controlled updates. We started our experiments with a physics simulation environment, the LunarLander which was used to test different possibilities of the vanilla REINFORCE and Proximal Policy Optimization. After observing the result of this phase, we formed decided on our approach to applying it to image captioning.

Our proposed methods were built on the self-critical sequence training algorithm that influenced later works in image captioning; and on PPO from the RL community, which can be considered a

descendant of the REINFORCE algorithm. Our approach comes in three configurations, the clipped self-critical (Clipped-SC), the pre-training with Clipped-SC, and the interpolated of clipped-SC and SCST.

After experimenting with our methods in the context of image captioning, the first two configurations did not achieve the observed positive results in the physics environment. The interpolated method, as our third learning algorithm, was tested with different coefficients. Although in the beginning, one of the modes of coefficients reached a slightly better reward than SCST, it did not sustain the initial growth and converged with the same rate as the original SCST.

These observations led us to hypothesize two possible reasons that might explain this discrepancy. The first one was that the self-critical mechanism is so dominant that it overwhelms the Clipped-SC and hence makes the method ineffective. The second possibility was other inherent differences in the nature of the two problems. The first hypothesis was tested by replacing the Clipped-SC with constant and random numbers. Our experimental results show that not only this amount of noise did not change the growth, but also it very slightly improved the outcome.

The bias-variance dichotomy can shed some light on this counterintuitive result. One of the reasons SCST was successful was its higher bias compared to the vanilla REINFORCE. Our Clipped-SC was introducing more bias which perhaps did not help an already saturated algorithm. The self-critical aspect of the SCST, which led to this high bias, did not allow the method or the noise to change the trend of growth.

6.1 Future Work

Although PPO would seem to be most obvious alternative to REINFORCE, there is a large repository of other techniques in the RL literature that can be explored and tested to achieve better performance in text generation. We would also like to investigate the differences between reinforcement learning in text generation context and the control problems such as simulated environments. Network architectures may be one of the determining factors in the success or failure of RL approaches.

In terms of continuing with PPO to understand its poor performance, and what it means for the use of RL in text-related tasks, a specific area of investigation is the effects of static clipping

on learning and its possible incompatibility with image captioning. The clipping ϵ can be more adaptive based on the learning progress: for instance, it may begin with a large ϵ resulting in little constraint and REINFORCE-like behaviour, and then increasing the constraint.A similar adaptive idea has been explored in the video games environment and succeeded in the development of PPO-λ [59], Trust Region-based PPO with Rollback (TR-PPO-RB) [60], and an evolutionary variant called PPO-CMA [61]. These methods can be appropriated for the image captioning task. We also leave the influences of network architecture as future work.

The other observation was the differences in sizes of the state space between the game environments such as Lunar Lander, and the image captioning may have caused the discrepancy. This hypothesis needs to be tested by developing and employing larger discrete state and action space games to see the effects of size on the algorithm performance.

A

State-of-the-art Deep RL Algorithms

Algorithm 2: Deep Q-learning algorithm [4]

Initialization;

Initialize replay memory;

Initialize target function $\hat{Q}$;

Initialize Q with arbitrarily values;

for *all the episodes* **do**

  Initialize states, s and preprocess them $\phi_1 = \phi(s_1)$

  Choose action a such that $a \in s$ using policy (i.e. ϵ-greedy);

  while *state s is not terminal* **do**

    Select action a, either random with ϵ probability or $argmax_a Q(\phi(s_t), a : \theta)$;

    Run action a and record reward,r, and the new state s';

    Update s and a: $s \leftarrow s'; a \leftarrow a'$ and preprocess the new state $\phi_{t+1} = \phi(s_{t+1})$;

    Save $(\phi_t, a_t, r_t, \phi_{t+1})$ in the experience replay;

    Sample random mini-batches of experience replay;

    if *episode terminates in the next step* **then**

      $Q_i = r_j$

    else

      $Q_i = r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta)$

    end

    Choose a' from s' using policy Q;

    Do a gradient descent step on $(y_i - Q(\phi_{j+1}, a'; \theta))^2$

    Every C steps reset $\hat{Q}; \hat{Q} = Q$;

  end

end

Algorithm 3: Deep Deterministic Policy Gradient algorithm (DDPG) [5]

Initialization

Randomly Initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with θs as its weight

Initialize target network Q' and μ' with weight $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer R

for *all the episodes* **do**

Initialize a random process N for action exploration

Receive initial observation state s_1

while *state s is not terminal* **do**

Select action $a_t = \mu(s_t|\theta^\mu) + N_t$ based on current policy and exploration noise

Run action a and record reward,r_t, and the new state s_{t+1}

Store transition (s_t, a_t, r_t, s_{t+1}) in R

Sample random mini-batches of experience replay

Set $y_i \leftarrow r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}))$

Update Critic by minimizing the loss: L $= \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

Update the critic policy using sampled policy gradient:

$$\nabla_{\theta^\mu} J \leftarrow \frac{1}{N}\sum \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

Update the target networks using τ:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\mu^Q + (1-\tau)\mu^{Q'}$$

end

end

Algorithm 4: Asynchronous Advantage Actor-Critic (A3C) [6]

Initialization

Initialize thread step $counter \leftarrow 1$

while $T < T_{max}$ **do**

  Reset gradients: $d\theta \leftarrow 0$ and $d\theta \leftarrow 0$

  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta_v^{'} = \theta_v$

  while *state* s_t *is not terminal and* $t - t_{start} < t_{max}$ **do**

    Do action a_t based on $\pi(a_t|s_t;\theta')$

    Get the reward r_t and the new state s_{t+1}

    $t \leftarrow t+1$

    $T \leftarrow T+1$

  end

  for $t-1 \leq i \leq t_{start}$ **do**

    $R \leftarrow r_i + \gamma R$

    Aggregate Gradients: $d\theta \leftarrow d\theta + \nabla\theta' log\,\pi(a_i|s_i;\theta')(R - V(s_i;\theta_v^{'}))$

    $d\theta_v \leftarrow d\theta_v + \frac{\partial (R - V(s_i;\theta_v^{'}))^2}{\partial \theta_v^{'}}$

  end

  Asynchronous update of θ and $\theta^{'}$

end

B

PPO Performance in Different Contexts

The performance is shown in two different contexts, the Atari video games and MuJoCo physics simulation environment. MuJoCo[1] is seen as a simulated robotic benchmark for RL algorithms.

[1] http://www.mujoco.org

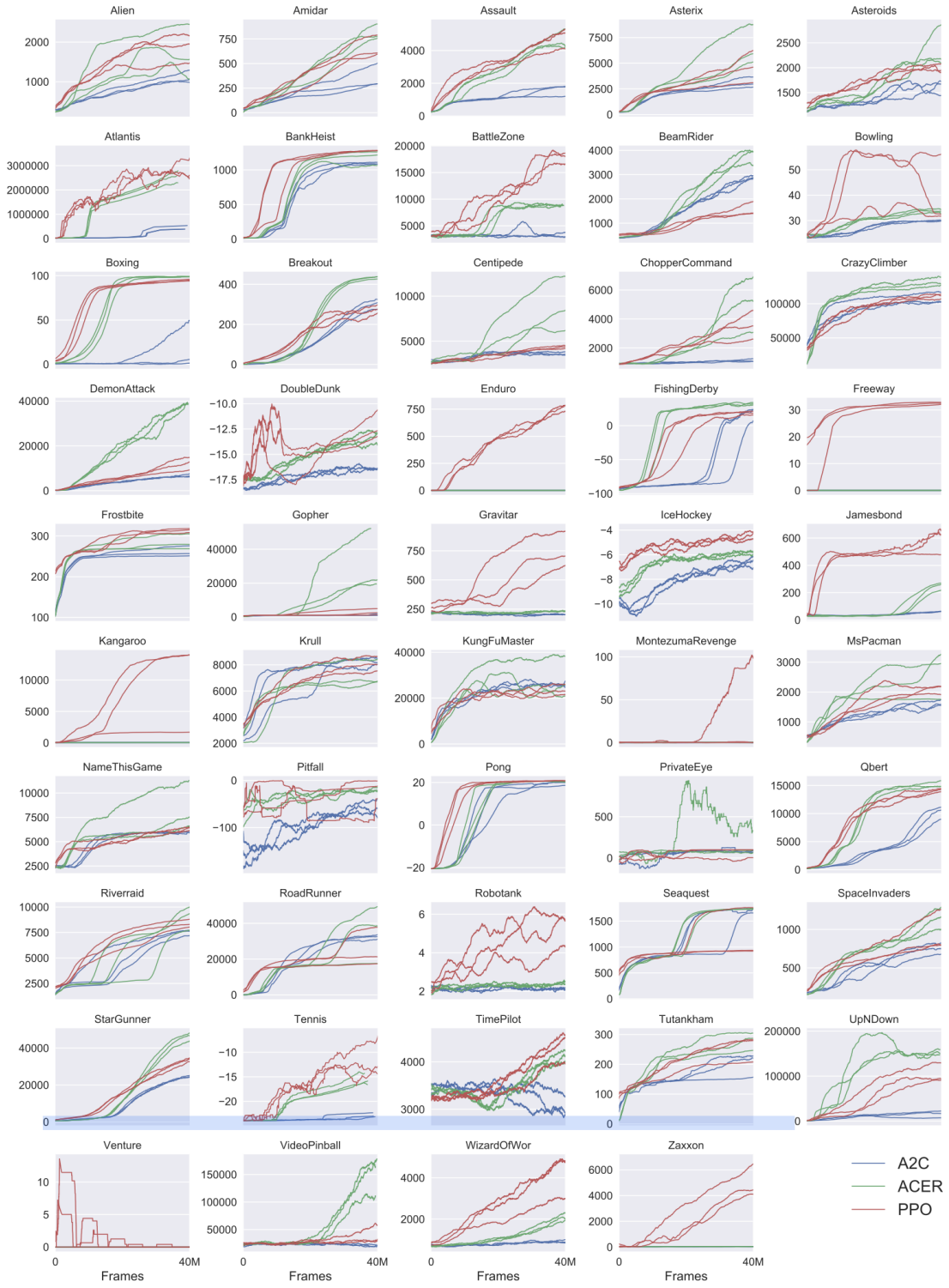

FIGURE B.1: A comparison of PPO and other state-of-the-art methods in the Atari context [3]

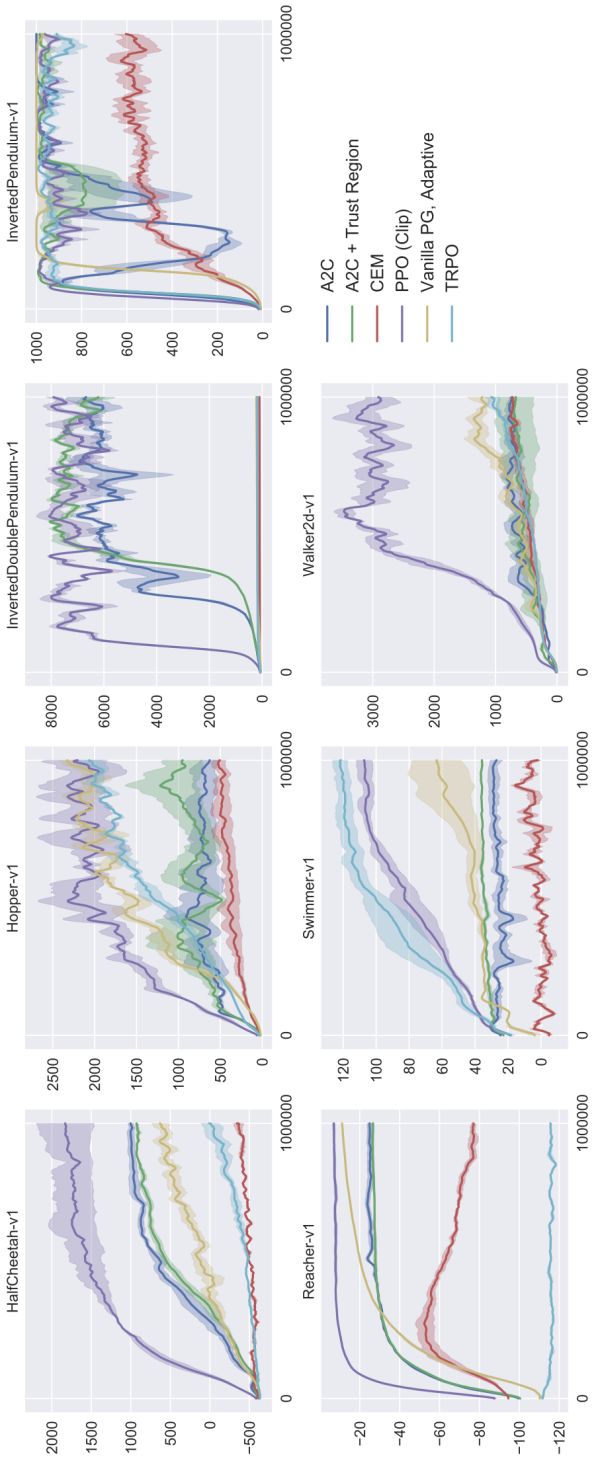

FIGURE B.2: A comparison of PPO and other state-of-the-art methods in the MuJoCo context [3]

C

The Lunar Lander Experimental Setting

The LunarLander implementation source code can be accessed via the link below:
https://github.com/FredAmouzgar/OpenAIGymExperiments.
The configurations are listed in the table below.

Hyperparameter	Value
Learning Rate	2×10^{-3}
Optimizer	Adam
Optimizer's βs	0.9 - 0.99
γ (Discount Factor)	0.99
ϵ for clipping	0.2
Number of Actions†	4
Number of Layers	3 Dense Layers
Hidden Layer Size	64 Neurons
Output Layer Size	Number of Actions†

D

SCST and Clipped-SC Experimental Setting

The Clipped-SC and SCST implementation source code can be accessed via the link below: https://github.com/FredAmouzgar/ImageCaptioningExperiments.

Hyperparameter	Value
Learning Rate	5×10^{-5}
Optimizer	Adam
Caption Model	FC
RNN size	512
Optimizer's βs	0.9 - 0.999
ϵ for clipping (for Clipped-SC)	0.2
Dropout probability	0.5
Mini batch size	10
Record Every	6000 iterations

List of Symbols

The symbols are explained in the sequence they appeared in the text.

RL	Reinforcement Learning
NLP	Natural Language Processing
PPO	Proximal Policy Optimization
DDPG	Deep Deterministic Policy Gradient
A3C	Asynchronous Advantage Actor-Critic
AI	Artificial Intelligence
ML	Machine Learning
OCR	Optical Character Recognition
DQN	Deep Q-Network
TRPO	Trust-Region Policy Optimization
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
BLEU	Bilingual Evaluation Understudy

ROUGE	Recall-Oriented Understudy
METEOR	Metric for Evaluation of Translation with Explicit ORdering
CIDEr	Consensus-based Image Description Evaluation
SPICE	Semantic Propositional Image Caption Evaluation
SCST	Self-Critical Sequence Training
MIXER	Mixed Incremental Cross-Entropy REINFORCE
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit activation function
ELU	Exponential Linear Unit activation function
Tanh	Hyperbolic Tangent activation function
w_i	The weight vector of an ANN
x_i	The input to an ANN
b	The bias value added to a perceptron
$f()$	The activation function performed on an ANN
$\mathbf{h_t}$	The hidden state of a recurrent neural network
a	An action available to a reinforcement learning agent
A	The set of actions available to the agent (action-space)
s	One state of the environment
S	The set of states that represent the world to the agent
r/R_a	Reward given for an action
P_a	Transition probability of an action

γ Dicount factor

y_t The output of an RNN

GAN Generative Adversarial Networks

VOC Visual Object Classes

PASCAL Pattern Analysis, Statistical Modeling, and Computational Learning

MS COCO Microsoft Common Objects in Context dataset

$Q(s, a)$ State-action value function

$V(s)$ State value function

TD Temporal Difference

π Agent's policy

DP Dynamic Programming

MC Monte-Carlo

ϵ The ϵ-greedy policy

α Learning step size

θ Model/Network parameters

$L(\theta)$ Loss function based on θ

G Return value from the Monte-Carlo rollout

KL Kullback–Leibler divergence

δ Trust-region constraint

$r(\theta)$ Ratio between the new and old policy

$\hat{A}$ Advantage function

$clip()$ The clipping function

ALE Arcade Learning Environment

References

[1] Charu C. Aggarwal. *Neural Networks and Deep Learning* (Springer, 2018). x, 6, 7, 8, 9, 10, 14

[2] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. *Sequence Level Training with Recurrent Neural Networks*. CoRR **abs/1511.06732** (2015). x, 4, 20, 22, 23, 24, 25, 36

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal Policy Optimization Algorithms*. CoRR **abs/1707.0**, 1 (2017). URL http://arxiv.org/abs/1707.06347. xi, 2, 3, 20, 21, 25, 26, 28, 29, 33, 36, 37, 40, 43, 56, 57

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. *Playing Atari with Deep Reinforcement Learning*. NIPS Deep Learning Workshop pp. 1–9 (2013). xiii, 2, 14, 17, 25, 52

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. *Continuous control with deep reinforcement learning*. CoRR **abs/1509.0** (2015). URL http://arxiv.org/abs/1509.02971. xiii, 2, 18, 53

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning*. In *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, pp. 1928–1937 (2016). URL http://arxiv.org/abs/1602.01783. xiii, 2, 19, 54

[7] A. L. Samuel. *Some studies in machine learning using the game of Checkers*. IBM JOURNAL OF RESEARCH AND DEVELOPMENT pp. 71–105 (1959). 1

[8] F. Amouzgar, A. Beheshti, S. Ghodratnama, B. Benatallah, J. Yang, and Q. Z. Sheng. *Isheets: A spreadsheet-based machine learning development platform for data-driven process analytics*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11434 LNCS, pp. 453–457 (Springer Verlag, 2019). 2

[9] A. Beheshti, F. Schiliro, S. Ghodratnama, F. Amouzgar, B. Benatallah, J. Yang, Q. Sheng, F. Casati, and H. Motahari-Nezhad. *Iprocess: Enabling iot platforms in data-driven knowledge-intensive processes*. In *Lecture Notes in Business Information Processing*, vol. 329 (2018). 2

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction* (The MIT Press, 2018), second ed. 2, 14, 15, 16, 20, 23

[11] H. van Hasselt, A. Guez, and D. Silver. *Deep Reinforcement Learning with Double Q-learning* (2015). URL http://arxiv.org/abs/1509.06461. 2, 17, 18

[12] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. *Dueling Network Architectures for Deep Reinforcement Learning* (9) (2015). URL http://arxiv.org/abs/1511.06581. 2, 18

[13] T. Note. *Q-Learning Technical Note* **292**, 279 (1992). 2

[14] R. J. Willia. *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*. Machine Learning **8**(3), 229 (1992). 3, 4, 20, 22, 25, 26, 36

[15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. *Trust Region Policy Optimization* (2015). URL http://arxiv.org/abs/1502.05477. 3, 20, 21, 25, 28, 37

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning* **48** (2016). URL http://arxiv.org/abs/1602.01783. 3

[17] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. *Baby talk: Understanding and generating image descriptions*. In *Proceedings of the 24th CVPR* (2011). 3, 10

[18] M. Hodosh, P. Young, and J. Hockenmaier. *Framing Image Description As a Ranking Task: Data, Models and Evaluation Metrics*. J. Artif. Int. Res. **47**(1), 853 (2013). 3, 10

[19] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. In F. Bach and D. Blei, eds., *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research*, pp. 2048–2057 (PMLR, Lille, France, 2015). URL http://proceedings.mlr.press/v37/xuc15.html. 3, 11

[20] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. *Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge*. IEEE Trans. Pattern Anal. Mach. Intell. **39**(4), 652 (2017). 11, 25

[21] A. Karpathy and L. Fei-Fei. *Deep Visual-Semantic Alignments for Generating Image Descriptions* (2014). URL http://arxiv.org/abs/1412.2306. 3

[22] H. Daumé Iii, J. Langford, and D. Marcu. *Search-based Structured Prediction*. Mach. Learn. **75**(3), 297 (2009). URL http://dx.doi.org/10.1007/s10994-009-5106-x. 3, 22

[23] L. Yu, W. Zhang, J. Wang, and Y. Yu. *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient* (2016). 4

[24] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. *Self-critical sequence training for image captioning*. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 **2017-Janua**, 1179 (2017). 4, 12, 20, 24, 25, 36, 40

[25] W. S. McCulloch and W. H. Pitts. *A logical calculus of the ideas immanent in nervous activity*. In *Systems Research for Behavioral Science: A Sourcebook*, pp. 93–96 (Taylor and Francis, 2017). 6

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning* (The MIT Press, 2016). 6, 7, 8, 9

[27] J. L. Elman. *Finding structure in time*. COGNITIVE SCIENCE **14**(2), 179 (1990). 9

[28] M. Schuster and K. K. Paliwal. *Bidirectional recurrent neural networks*. IEEE Transactions on Signal Processing **45**(11), 2673 (1997). 9

[29] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation* (2014). URL http://arxiv.org/abs/1406.1078. 9

[30] S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*. Neural Comput. **9**(8), 1735 (1997). URL http://dx.doi.org/10.1162/neco.1997.9.8.1735. 9

[31] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. *Generative adversarial text to image synthesis*. In *33rd International Conference on Machine Learning, ICML 2016*, vol. 3, pp. 1681–1690 (International Machine Learning Society (IMLS), 2016). 10

[32] P. Anderson. *Vision and Language Learning: From Image Captioning and Visual Question Answering towards Embodied Agents*. Tech. rep. (2018). 10, 25

[33] M. Boden. *Mind As Machine: A History of Cognitive Science* (Oxford University Press, Inc., New York, NY, USA, 2008). 10

[34] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. *Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN)*. CoRR **abs/1412.6632** (2014). 11

[35] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2015). 11

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. Int. J. Comput. Vision **115**(3), 211 (2015). URL http://dx.doi.org/10.1007/s11263-015-0816-y. 11

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin. *Attention is All you Need*. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., *Advances in Neural Information Processing Systems 30*, pp. 5998–6008 (Curran Associates, Inc., 2017). URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf. 11

[38] R. Luo, B. L. Price, S. Cohen, and G. Shakhnarovich. *Discriminability objective for training descriptive captions*. CoRR **abs/1803.04376** (2018). URL http://arxiv.org/abs/1803.04376. 12, 25, 40

[39] A. A. Liu, N. Xu, H. Zhang, W. Nie, Y. Su, and Y. Zhang. *Multi-level policy and reward reinforcement learning for image captioning*. In *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2018-July, pp. 821–827 (International Joint Conferences on Artificial Intelligence, 2018). 12

[40] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. *Collecting Image Annotations Using Amazon's Mechanical Turk*. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pp. 139–147 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2010). 12

[41] *The PASCAL Dataset Website*. URL http://host.robots.ox.ac.uk/pascal/VOC/. 12

[42] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. *From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions*. Transactions of the Association for Computational Linguistics **2**, 67 (2014). 12

[43] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. *Microsoft COCO: Common Objects in Context*. In *ECCV* (2014). 13

[44] *Common Objects in Context Dataset*. URL http://cocodataset.org/. 13

[45] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. *Microsoft COCO Captions: Data Collection and Evaluation Server*. ArXiv **abs/1504.00325** (2015). 13

[46] K. Papineni, S. Roukos, T. Ward, and W.-j. Zhu. *BLEU: a Method for Automatic Evaluation of Machine Translation*. pp. 311–318 (2002). 13

[47] P. Anderson, B. Fernando, M. Johnson, and S. Gould. *SPICE: Semantic Propositional Image Caption Evaluation*. CoRR **abs/1607.08822** (2016). URL http://arxiv.org/abs/1607.08822. 13

[48] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, Inc., New York, NY, USA, 1994), 1st ed. 15

[49] C. J. C. H. Watkins and P. Dayan. *Q-Learning*. Tech. rep. (1992). 16

[50] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. *Prioritized Experience Replay*. CoRR **abs/1511.0**, 1 (2015). URL http://arxiv.org/abs/1511.05952. 17, 18

[51] D. Silver, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. *Deterministic Policy Gradient Algorithms*. Tech. rep. 18

[52] B. Dai, S. Fidler, R. Urtasun, and D. Lin. *Towards Diverse and Natural Image Descriptions via a Conditional GAN*. Proceedings of the IEEE International Conference on Computer Vision **2017-Octob**, 2989 (2017). 25

[53] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva. *Deep Attention Recurrent Q-Network* pp. 1–7 (2015). URL http://arxiv.org/abs/1512.01693. 25

[54] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. *The Arcade Learning Environment: An Evaluation Platform for General Agents* (2012). URL http://arxiv.org/abs/1207.4708http://dx.doi.org/10.1613/jair.3912. 27

[55] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica. *RLlib: Abstractions for Distributed Reinforcement Learning* (2017). URL http://arxiv.org/abs/1712.09381. 27

[56] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. *OpenAI Gym* (2016). URL http://arxiv.org/abs/1606.01540. 27

[57] OpenAI. *OpenAI Gym, LunarLander-v2*. URL https://gym.openai.com/envs/LunarLander-v2/. 27, 28

[58] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. *High-Dimensional Continuous Control Using Generalized Advantage Estimation* (2015). URL http://arxiv.org/abs/1506.02438. 43

[59] G. Chen, Y. Peng, and M. Zhang. *An Adaptive Clipping Approach for Proximal Policy Optimization* (2018). URL http://arxiv.org/abs/1804.06461. 50

[60] Y. Wang, H. He, and X. Tan. *Truly Proximal Policy Optimization*. Tech. rep. URL https://github.com/. 50

[61] P. Hämäläinen, A. Babadi, X. Ma, and J. Lehtinen. *PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation* (2018). URL http://arxiv.org/abs/1810.02541. 50