Deep Reinforcement Learning Approach Based Grammatical Error Correction

Muhammad Hamza Akbar
COMSATS University Islamabad
Rabail Asghar
COMSATS University Islamabad
Muzammal Hussain
COMSATS University Islamabad
Muhammad Farhan (farhan@cuisahiwal.edu.pk)
COMSATS University Islamabad
Faiz Abdullah ALOTAIBI
King Saud University
Mrim M. Alnfiai
Taif University

Research Article

Keywords: Grammatical Error Correction, Deep Reinforcement Learning, Deep Neural Networks, Policy Gradient, Sequential Decision Making, Natural Language Processing

Posted Date: September 18th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3336364/v1

Abstract

Natural Language Processing (NLP) is a field of study that focuses on the interaction between computers and human language. It encompasses various techniques and methodologies aimed at enabling machines to understand, interpret, and generate human language text or speech. NLP plays a crucial role in many applications, including machine translation, sentiment analysis, information retrieval, and Grammatical Error Correction (GEC). Grammatical Error Correction (GEC) is an important task in natural language processing that aims to automatically correct errors in written text. It involves detecting and correcting errors related to grammar, syntax, spelling, punctuation, and other linguistic aspects. However, existing study is solely based on classical machine learning and deep learning methods for GEC. This study proposes a new approach using a DQN model and leverages the C4_200M dataset to automate the GEC process. The main goal of this study is to optimize the selection of action-value function (Q-function) and train a DRL model for automatic grammatical error correction and set baseline results using reinforcement learning (RL) techniques. Findings show that the proposed DQN model outperformed machine learning and rule-based techniques.

1 Introduction

Grammatical error correction (GEC) is a crucial task in the realm of Natural Language Processing (NLP) that aims to automatically identify and rectify grammatical errors in written text. GEC goes beyond traditional spelling and grammar checks, encompassing a broader range of linguistic aspects such as syntax, punctuation, and semantics. Automatic grammatical error correction in the English language provides several benefits that enhance the quality and effectiveness of the written English language. Firstly, it improves clarity and coherence, ensuring that the intended message is accurately conveyed. This is particularly valuable for communication in academic, professional, and informal settings. Secondly, Grammatical Error Correction reduces misunderstandings and misinterpretations that can arise from grammatical errors, promoting accurate and precise communication. Classical machine learning techniques for Grammatical Error Correction (GEC) have limitations in generalization to unseen errors, manual feature engineering, limited contextual understanding, difficulty handling ambiguity, and limited interpretability. They rely on annotated data and predefined rules, which may not cover all error types and variations in language usage. These techniques struggle with capturing complex contextual dependencies and may require regular updates to adapt to evolving language patterns. Additionally, classical machine learning models may have difficulty disambiguating between multiple error corrections and lack clear explanations for their decisions.

In this study we proposed the DRL technique to automate grammatical error correction which benefits from capturing complex dependencies in language, allowing for a more nuanced understanding and generation of corrections. They can learn directly from raw textual data without requiring explicit rule-based systems or extensive feature engineering. Moreover, DRL models have the potential to adapt and generalize well to unseen errors and language variations. For this purpose, we trained the DQN model on

the C4_200M dataset. The proposed model surpasses the classical machine learning and deep learning techniques in terms of better correction of errors with minimum loss and higher accuracy/reward.

The objective of this study is to answer the following questions.

1. Can the implementation of the DRL model make GEC more successful than previous ML and DL techniques?

2. Can the optimized Q function and POS tagging have any influence on the agent and environment of the DQN model for GEC?

Our study provided several significant contributions, including:

• By incorporating DRL, the study will contribute to pushing the boundaries of GEC techniques beyond traditional rule-based and machine-learning methods. It will demonstrate the potential of DRL in automating the correction process and improving the accuracy and contextuality of error corrections.

• This study aims to optimize the selection of the action-value function (Q-function) within the DQN model. This optimization will enhance the decision-making process of the model, leading to more effective and accurate error corrections.

• This study will establish baseline results for GEC using reinforcement learning (RL) techniques. By comparing the performance of the DQN model with traditional machine learning and rule-based techniques, the study will provide insights into the effectiveness and superiority of the DRL approach.

• The findings of this study can have practical implications for various domains involving written communication, such as education, language learning, and automated proofreading systems.

The remaining sections will be organized as follows: section II is the literature work that explains the previous approaches used for Grammatical Error Correction. Section III has the proposed methodology of this study. Section IV has the results and discussion. And lastly, Section V has a conclusion and future work of the study.

2 Literature Review

Over the last few years, Grammatical error detection and repair have received a lot of study attention. By providing a summary of the most recent study in the field, this chapter places the current study in perspective.

2.1 Early Approaches to Grammatical Error Correction

Hand-coded rules were used in early attempts at automatic error correction. The first commonly used grammar checkers relied on straightforward pattern matching and string replacement, such the Writer's Workbench [1], were based on simple pattern matching and string replacement. Other rule-based systems

used manually created grammar rules and syntactic analysis. For instance, IBM's Epistle [2] and Critique [3] performed complete syntactic analysis, in contrast to Aspen Software's Grammatik [4] which both relied on fundamental linguistic analysis. For some sorts of errors, rule-based methods are frequently simple to implement and can be quite effective. This is why current grammar checking algorithms continue to make extensive use of them. Rules, however, may eventually become unworkable for some complicated faults and unmanageable. It is impossible to create rules for every possible error due to language's great productivity. Rule-based methods are thus frequently shunned as a general fix.

When extensive annotated resources first became available in the 1990s, researchers switched to data-driven methodologies and used machine learning techniques to create classifiers for particular mistake types [5]–[11]. Two mistake types—articles and prepositions—have received the majority of attention in research utilizing machine learning classifiers on a large diverse corpus and achieved an accuracy of 88%. This is because these errors are some of the most prevalent and difficult ones for ESL learners, and they are also simpler to correct using machine learning techniques than by manually writing rules. [9] used maximum entropy models to correct errors for 34 common English prepositions in learner text utilizing machine learning classifiers. This is because these errors are some of the most prevalent and difficult ones for ESL learners, and they are also simpler to correct using machine learning techniques than by manually writing rules [12]. A finite confusion set or candidate set, such as a list of English articles or prepositions, is defined for these closed-class errors and includes all potential repair possibilities. During the training process, examples used to train grammatical error detection systems, whether they are native or learner data, are represented as vectors of linguistic properties deemed relevant to the specific error type. These linguistic properties can include neighboring words, part-of-speech tags, grammatical relations (GRs), and dependency trees, among others. Using various machine learning methods, classifiers are trained based on these features. Once a system has been trained, it compares the most likely choice predicted by the classifier with the original term used in the text. This comparison allows the system to identify and rectify new errors. It is vital to create distinct classifiers for each type of error since the most beneficial characteristics frequently rely on the word class. [6] used a large, diverse dataset to train a maximum entropy classifier to identify article errors, and the accuracy was 88%. Maximum entropy models were employed by [9] to fix 34 typical English prepositional errors in learner material.

ESL learners' L1s frequently have a role in their mistakes [13]. Systems perform significantly better when their L1s are taken into account. For the purpose of fixing prepositional errors, [10] compared four linear machine learning classifiers. Results demonstrated that discriminative classifiers perform best, and that performance can be further enhanced by L1 adaptation. Rather than training individual classifiers for each specific native language (L1), the authors proposed the inclusion of language-specific priors using Naive Bayes (NB) models during the decision-making process. This approach involves incorporating prior knowledge specific to each language when making predictions. By utilizing NB models, which assume independence between features, the system can factor in language-specific characteristics to improve the accuracy of error detection and correction.

Techniques based on "classification by error type" have a drawback in that they primarily focus on local context and treat errors in isolation, assuming that there is only one error present and that all surrounding information is accurate. However, the errors made by language learners can often interact and combine in complex ways. This limitation restricts the practical applications of an error correction system that exclusively targets one specific type of error. For effective language learning, it is crucial to address the interplay and intricate relationships between different types of errors, rather than considering them in isolation.

Building several classifiers and then cascading them into a pipeline system is a frequent technique. Systems that repair multiple errors are frequently built using a combination of classifier-based and rule-based procedures [11], [14]. The order of the classifiers is important, and this type of solution is labor-intensive and requires numerous pre- and post-processing stages. Furthermore, it does not address the issue of interacting errors, and predictions made by different classifiers might not agree. The following is a typical illustration from [15]:

Example

electric cars is still regarded as a great trial innovation ...

Predictions made by a system that combines independently-trained classifiers: cars is → car are.

The issue of interacting errors has been addressed using a variety of strategies. [16] created a beam-search decoder to iteratively produce candidate sentences at the sentence level and score them using individual classifiers and a general LM rather than making decisions independently. The following five types of alterations were made to the five proposers to create new candidates: orthography, articles, prepositions, punctuation insertion, and noun number. The decoder showed superior performance compared to a pipeline system consisting of separate classifiers and rule-based processes, which indicated promising results. However, it should be noted that the decoder only addresses five specific types of errors. To cover a wider range of grammatical problems, additional modules or components need to be incorporated into the system. Designing these additional components can be challenging, as they must effectively handle various types of errors that may pose difficulties in correction. Nevertheless, expanding the system to include more proposers is necessary to enhance its capabilities and address a broader spectrum of grammatical issues. Additionally, the number of candidates increases exponentially as the type of errors (i.e., the quantity of proposers) and the length of the sentence increase. It is impossible to count all candidates, so creating a reliable decoder becomes challenging. [17] suggested a combined inference model to address contradictions brought about by different classifiers. Integer Linear Programming (ILP) was used to integrate the output of different classifiers with a set of linguistic constraints. These limitations were personally established and immediately written into the system. In order to take into account brand-new types of interacting errors, any additional constraints must be manually defined. Language uses the subject-verb and article-NPhead structures. [15] addressed by developing two joint classifiers. Instead of utilising two classifiers independently for each of the structures, a joint classifier simultaneously predicts two words that are a component of the same

structure. The joint classifier, in contrast to the ILP model suggested by [17], does not require human-defined limitations because it may learn directly from the training data. On the other hand, it is more difficult to compile enough pairs of candidates to represent the necessary structures and use them as training data. One joint classifier can only target one form of interacting mistake, therefore new classifiers must be built for every new type of interaction. These classifier-based systems still rely on the individual classifier scores, making it time-consuming to train each classifier for all potential types of (interacting) errors.

Using n-gram LMs is a more comprehensive method for fixing many faults in ESL text [18], [19]. After being trained on a large number of precise phrases, a single model is used to assign probabilities to word sequences based on counts from the training data. In this approach, the target word sequence is replaced with substitutes drawn from a precompiled candidate set, and the LM scores for both the original text and the replacements are calculated. Whichever has the highest likelihood is chosen as the proper order. Correct word combinations should theoretically have high probabilities whereas incorrect or undetected ones should have low possibilities. Parts of a sentence with low scores are presumed to contain errors. No matter how large a training corpus is, it is impossible to cover every conceivable accurate word sequence in practice. Another problem is how to distinguish inappropriate word combinations from low-frequency ones. The LM technique is widely used in conjunction with other approaches to prioritize modification suggestions offered by other models. [18] coupled machine learning classifiers with an LM in addition to LM classifiers. [16] scored correction candidates in a beam-search decoder using an LM and classifiers.

Additionally, several initiatives have been made to address learner mistakes that are particularly challenging to find and fix. A linguistically motivated method to verb mistake correction was put out by [20]. In order to initially identify verb candidates in noisy learner text, their methodology coupled a rule-based system with a machine learning approach. From there, verb finiteness information was used to detect errors and identify the specific sort of error. The development of a computational approach for spotting plagiarism in ESL essays. [21] They proposed a technique to award high ratings to words and phrases that are likely to be repetitive inside a given sentence by comparing an ESL sentence with the output from commercial MT systems. Using compositional distributional semantic models, [22] accomplished mistake detection for adjective-noun and verb-object combinations in learner data for content word combinations.

2.2 Machine Learning-Based Approaches and Error Correction

It is possible to distinguish two basic types of grammatical error correction methods regarding English compositions: grammatical error correction methods based on rules and grammatical error correction methods based on statistics, when it comes to English compositions. The former is done by handwriting grammar rules on a piece of paper. A statistical model, such as *n*-grams, is used in English composition in order to correct grammatical errors that have been committed by the writer. There have been several

early grammar correction tools that use rule-based approaches [23]. In spite of the fact that the Japanese language lacks articles and singular and plural nouns, English expressions tend to use these in their place. The issue of translation from Japanese to English must always be considered when translating the two languages. A literature study [24] proposes guidelines to determine whether a singular or plural noun should be added to the translated sentence based on its context. A recent study found that the accuracy rate of this method was 89 percent, which is in line with the results of the test. An approach based on rules offers many advantages over a rules-based approach. The addition, modification, or deletion of grammar rules is remarkably simple. Users can receive more specific and targeted feedback by adding grammatical explanations to each rule. A key feature of the system is its ability to debug a problem directly because the system provides prompt information in the rules. Rule bases are easier to write for linguists without programming skills or limited programming knowledge. It is difficult, however, to use corpus statistics in handwritten rules due to exceptions, as noted in the literature [25] .The use of statistics can also be used to correct grammatical errors if they are caused by a faulty sentence structure. Statistics-based grammar checking treats grammar checking primarily as a classification task, focusing mainly on article and preposition error checking [26]–[28] .Literature [29] typically uses vocabulary and parts-of-speech features for classification. Language model scores, neighboring words, and part-of-speech marks are included. Literature [30] also adds analytical features, resulting in a higher accuracy rate and recall rate for preposition error correction. Accordingly, the classification algorithm makes use of the maximum entropy algorithm, the voting perceptron algorithm [30], and the naive Bayes algorithm [31]. As well as ensuring that no errors occur in prepositions and articles, some attention has also been paid to verb form error checking [32], [33]. A classification algorithm's main advantage is its capability of correcting certain types of errors. Recently, some work has been done to correct various types of errors comprehensively. Various errors are detected using a high-order sequential labeling model in the literature [34]. An approach based on rules and syntactic *n*-grams is used in the literature [35]. In this document, syntactic ngrams contain syntactic information, unlike traditional *n*-grams. Historically, most grammatical errors were corrected by correcting articles and prepositions, since these are common errors made by non-native speakers. Grammar points such as clauses are also difficult to master, as they play an important role in English writing expression. Studying the Chinese learners corpus, we found that related word errors are the most common type of clause grammatical errors, and they are difficult to correct. Nevertheless, not much research has been conducted on automatic error correction for English clauses. An algorithm based on statistical machine translation is used in the literature [36]. The language model can then be used to correct all of the errors of CoNLL2014, including false errors. Machine translation also involves automatic grammar correction. For this reason, we need to find a solution, an automatic editing system is required to enhance the output of the machine translation system, since Japanese lacks articles, for instance. In order to choose the right article, the output sentence needs to be corrected when translating Japanese into English. Using these types of systems for grammatical correction, literature [26] improves machine translation by solving the article selection problem.

The different faults that ESL students make should be able to be corrected by a viable error correction system. In more recent studies, MT approaches have been effectively applied to fix a wider range of

problems.

Text is automatically translated into a target language by MT algorithms from a source language. Error correction can therefore be seen as a special translation problem from grammatically incorrect statements into proper ones. Unlike traditional MT jobs, the source and target sentences are both in the same language, even when the source sentences could be grammatically incorrect. The correction mappings that MT-based GEC systems have learnt from parallel examples are used to generate a corrected version of the original (incorrect) sentence that fixes as many errors as possible.

2.3 Machine Translation-Based Methods

As a sequence-to-sequence (seq2seq) method, the machine translation-based approach can convert faulty sentences into the proper ones. Several strategies have been put out in recent years to enhance the effectiveness of grammatical error correction models based on machine translation. Reference[37] applies a seq2seq model to a pre-trained masked language model, such as BERT. A copy-augmented architecture for grammatical error correction is suggested in reference [38]. Using translation models to create more synthetic data for pre-training is the topic of reference [39]. Reference[40] explicitly introduces noise to back-translate standard sentences. Create a seq2seq model with a multi-layer convolution and attention mechanism for Chinese text according to [41]. A BiLSTM-based machine translation model is suggested in reference[42] to capture long-distance interdependency. The machine translation-based models continue to suffer from creating results from scratch despite the aforementioned improvements, which inevitably result in over-correction and generation errors.

2.4 Sequence Tagging-Based Methods

By defining the objective of correcting grammatical errors as a sequence tagging task, another line of study adopts a different perspective. These models often perform an edit to incorrect tokens and forecast a specified set of tags depending on the original phrase. According to reference [43], editions occur when a token or term is kept, dropped, or added to an existing lexicon. For a fixed number of iterations, reference [44] predicts token-level editions sequentially in a non-autoregressive manner. To produce more compact editions, reference [45] generates span-level tags. Reference [46] further develops the strategy by creating more precise editions based on Arabic lexical norms. while accomplishing grammatical error correction assignments in English reasonably effectively. In addition, rather than fixing grammatical problems, the majority of the models listed above concentrate on identifying them. Using BERT for text prediction, reference [47] adds a mask to the text places designated as missing mistakes for Chinese grammatical error repair. This method still needs a different correction model for grammatical error correction, though.

We employed the English grammatical error correction model GECToR [46] with a unique dynamic word embedding upgrade and a residual connection network to further increase the English grammatical error correction capacity. By training the model on an enhanced dataset, we increase the algorithm's capacity to handle complicated tags in the interim.

2.5 Machine Learning and Deep Learning Approaches

These algorithms can comprehend human language without being explicitly coded thanks to statistical techniques, and they are now commonly utilized in GEC tasks. Systems that use machine learning and deep learning begin by examining the training set in order to gain information and create their classifiers and rules. Deep learning algorithms are based on probabilistic outcomes, and their primary benefit is their learnability. They also don't require manual rules or grammar coding. Furthermore, viewing error correction as a translation process is a desirable and easier approach. The basic premise is that a statistical machine translation (SMT) system should be able to convert material written in "poor" (incorrect) English into "good" (correct) English. Studies like reference [48], [49], have used this technique. There have been numerous attempts to create a hybrid system that combines rule-based and deep-learning algorithms. These studies include [50], [51], which used a grammar-based parser for text-to-SQL translation and deep learning to supplement rule-based grammar by fixing the syntax and removing typos. Additionally, the earlier end-to-end GEC methods relied on recurrent neural networks, were manually created, and were frequently derived from short words (RNNs). Additionally, there are constraints on training models based on constrained error correction sentence pairings that prevent models from precisely correcting sentences. Additionally, it is typically inappropriate to use single-round grammar corrections on sentences that include many faults. Table.1 provides a summary of the literature review.

Table 1
A Glimpse of Previous Work

Reference	Model	Dataset	Accuracy
[52]	Sequence to Sequence model	CoNLL-2014 Shared Task, JFLEG	F0.5: 61.15 (CoNLL-2014 Shared Task), GLEU: 61.0 (JFLEG) 11†source
[53]	Multilayer Convolutional Encoder-Decoder Neural Network	CoNLL-2014 Shared Task, JFLEG	F0.5: 54.79 (CoNLL-2014 Shared Task), GLEU: 57.47 (JFLEG) 17†source
[54]	Neural Network Translation Models	CoNLL 2014 test set	F0.5: 41.75* 40†source
[55]	Neural Network models	CoNLL-2014 Shared Task	81.6 (CoNLL-2014 Shared Task)
[56]	Contextualized word embeddings	CoNLL-2014 Shared Task	F0.5: 52.2 (CoNLL-2014 Shared Task), GLEU: 52.3 (CoNLL-2014 Shared Task)
[57]	Sequence to Sequence model	JFLEG	F0.5: 46.17 (JFLEG), GLEU: 42.83 (JFLEG)
[58]	BERT-based models	CoNLL-2014 Shared Task	F0.5: 56.4 (CoNLL-2014 Shared Task)

2.6 Shared Tasks on Grammatical Error Correction

Over the previous few years, four GEC shared tasks have provided participating teams with a place to compare results on shared training and test data. Participants are instructed to build their GEC systems in a few months utilizing any publicly available information and resources after receiving a fully annotated training set. The performance of the systems is then assessed for the participating teams using fresh data from a blind test. Within a few days after the test results being published, systems should be able to identify grammatical problems in material written by non-native speakers and deliver repaired versions. The organizers then assess the output of each system and offer the final rankings.

2.6.1 HOO 2011 & 2012

The Helping Our Own (HOO) shared tasks from 2011 and 2012 were the first in the NLP community to promote the use of NLP tools and methodologies for the development of automated systems that may help non-native authors in their writing [59], [60]. A collection of texts authored by non-native authors and culled from the ACL Anthology were sent to participants in the HOO-2011 shared task. All textual errors had to be automatically found and corrected. Based on the CLC coding scheme, errors were divided into 13 different error kinds. There were six teams involved in the work, and some of them excelled by concentrating solely on a small number of fault kinds.

Due to HOO-2011's difficulty, the HOO-2012 shared task only examined article and prepositional errors. The designated training set was the FCE dataset. There were 14 teams who participated, and the majority of them created machine learning classifiers. For evaluation in both HOO shared tasks, the P, R, and F-score between a system's edit set and a manually created gold-standard edit set were calculated.

2.6.2 CoNLL 2013 & 2014

The following two joint projects were completed in conjunction with the Conference on Computational Natural Language Learning (CoNLL). Three new error types—noun number (Nn), verb form (Vform), and subject-verb agreement (SVA)—were added to the HOO-2012 scope by the CoNLL-2013 shared effort [61]. This revised error list is more thorough and includes interaction faults in addition to article (ArtOrDet) and preposition (Prep) issues. Used as in-domain training data was NUCLE v2.313. 50 brand-new essays that were prepared in response to two prompts make up the test data. The training set likewise had one prompt, but the other was brand-new.

The M2 scorer was used to evaluate systems, and the organizers suggested restricting the maximum number of unmodified words per edit to three by setting the max unchanged words parameter to three. P and R were given equal weight when determining rankings using F1. As there was only one set of gold annotations at first, participating teams were later given the chance to offer different solutions (gold-standard modifications), as there are typically multiple appropriate fixes for numerous faults. This procedure was used for the shared HOO duties in 2011 and 2012. There were so two evaluation rounds, the second of which permitted alternate responses. Ng [61] pointed out that these new ratings tended to favor the teams that provided alternate replies. Therefore, they advised against using alternative replies in future evaluation in order to lessen bias. Ultimately, 17 teams took part in CoNLL-2013. A frequent

strategy used by these teams was to create classifiers for each sort of mistake. Heuristic rules, MT, and LM were among more methodologies.

The CoNLL-2014 shared task [62] attempted to stretch GEC's limits once more by switching back to an all-errors correction task. Three significant changes in particular were made in comparison to CoNLL-2013: Two human annotators independently annotated the test writings, participating systems were required to fix all forms of grammatical errors, and the evaluation measure was altered from F1 to F0.5, prioritizing P over R. As official training data, NUCLE v3.0, a more recent version of the NUCLE corpus, was utilized. A new collection of 50 essays written by non-native English speakers was used as supplementary blind test data. The CoNLL-2013 test set is available for unrestricted training and/or improvement. The official scorer was once more the M2 scorer.

13 teams in all submitted work to CoNLL-2014. The majority of them created hybrid systems that integrated various techniques. LM and MT approaches were utilized for non-specific error type correction, whilst rule-based methods and machine learning classifiers were favored for correction of single error types.

2.7 Deep Learning-Based Approach

Two broad methods correct spelling errors based on context. As a first step, words that may be corrected are generated, and as a second step, the candidate word is related to the context and is used for determining the final correction. A typical method of generating candidate words determines the edit distance between the target word and its dictionary equivalent [63]. As a result, distance from the keyboard corresponding to the edit distance were taken into consideration in the candidate generation method [64], which is based on the environment for keyboard input and restricts candidate generation accordingly. An approach that uses contextual information [65] has recently been developed to eliminate the need for word comparisons. This study generated and searched error words using this method. 3-gram is candidate word generation method based on contextual information. A corpus of ten quadrillion English words was used to generate a variety of high-quality candidate words. Developing or selecting an appropriate correction language model is the next research objective. For the purpose of context-sensitive spelling error correction, both statistical and deep learning methods are used. As of now, statistical corrections have usually been used in conjunction with the noise channel model [63] or the n-gram-based language model [66]. Research on Korean statistical methods include smoothing, interpolation, and improving n-gram search structures based on the noisy channel model [64], [65], [67]. Deep learning has recently been used to develop a method of correcting words using recurrent neural networks and convolutional neural networks [68], [69], as well as word embedding [70]. Context-sensitive spelling errors have been neglected in recent years; however, the correction of documents can be of substantial benefit to researchers and writers of texts. This paper discusses context-sensitive spelling error correction for a variety of words in a variety of documents. We chose an unsupervised deep learning model to solve context-sensitive spelling errors because it is difficult to obtain correct answers to all spelling errors. Our method uses a variety of deep learning language models, the purpose of this is to correct context-sensitive spelling errors.

3 Proposed Methodology

In this section, we will discuss the properties of the dataset and implementation details of the proposed approach. The detailed architecture of the proposed methodology is shown in Figure.1.

3.1 Environmental Setup

For this study, we utilized the Google Cloud Platform (GCP) with NVIDIA T4 GPUs and with a substantial 256 GB RAM. The E2 processor family was employed, offering a balance between computational performance and cost efficiency for the study requirements. The combination of the GCP, NVIDIA T4 GPUs, ample RAM, and the E2 processor family formed a robust environment setup.

3.2 C4_200M Corpus

The C_200M dataset utilized in this study is a large-scale corpus designed specifically for grammatical error correction (GEC) tasks, serving as a valuable resource for training and evaluating deep reinforcement learning (DRL) models. It encompasses approximately 200 million sentences sourced from a wide array of materials, including books, articles, and web documents. The distribution of sentences is shown in Figure.2. Each sentence within the dataset deliberately contains a diverse range of grammatical errors, spanning grammar, syntax, spelling, punctuation, and other linguistic aspects. These errors represent common mistakes encountered in written language, presenting a realistic and comprehensive sample for GEC investigations. With its extensive scale and diverse error patterns, the C4_200M dataset facilitates the effective training and evaluation of DRL models, thereby driving advancements in automated GEC systems. This data includes the following statistics shown in Table.2.

Table 2
Statistics of C4_200M Corpus

Description	Values
Maximum Length of Sentence	7092 Words
Minimum Length of Sentence	1 Word
Total Number of Tokens in Incorrect Sentences	8861928
Total Number of Tokens in Correct Sentences	8774798

3.3 Preprocessing

Preprocessing plays a vital role in preparing the dataset for grammatical error correction (GEC) tasks. The detailed architecture is shown in Figure.3.

3.3.1 Normalization

A crucial preprocessing step known as normalization was performed on the dataset. Normalization is a fundamental technique used to bring data into a standardized and consistent format, ensuring fairness

and comparability across different data points. By applying normalization techniques, the dataset's values were rescaled or transformed to adhere to a specific range or distribution, eliminating any potential biases or variations that could impact subsequent analyses or modeling. By performing normalization, the dataset was prepared for subsequent data analysis, modeling, and machine-learning tasks. Normalization plays a vital role in enhancing the reliability and accuracy of these processes by providing a consistent and equitable basis for comparison and evaluation.

The normalization is applied by below mentioned tasks.

• **Special Character Removal**: In this step, we remove all irrelevant special characters from the dataset to achieve maximum reliability. The special charcters that we removed are: @ \# \& \$ \% _ = + [ ] { } | < > ? / ~ ! * () {}

• **Numerical Digits**: We removed numerical integers [0–9]

3.3.2 Tokenization

In order to facilitate a better understanding of the data by the model, the dataset underwent a crucial preprocessing step known as tokenization. Tokenization is the process of breaking down a text into individual units, known as tokens, which can be words, subwords, or characters. By tokenizing the dataset, the text was segmented into meaningful units, allowing the model to process and analyze the information more effectively. Tokens serve as the fundamental building blocks for various natural language processing tasks, enabling the model to comprehend the structure and meaning of the text.

We used the NLTK Python library for this purpose. In the context of this study on GEC using DRL with the DQN model, tokenization was a critical step in preparing the dataset for training the model. It enabled the model to interpret the text at a granular level, effectively capturing the grammatical errors and providing the necessary context for error correction.

3.3.3 POS Tagging

During this particular step of the processing, an essential task known as parts of speech (POS) tagging was executed. POS tagging involves assigning a grammatical label or tag to each word in a given sentence, indicating its syntactic role and category within the sentence structure. This linguistic analysis aids in understanding the grammatical structure and meaning of the text. By employing the POS tagging, each word in the text corpus was assigned a corresponding POS tag. These tags typically include categories such as nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and more. The resulting tagged corpus provides valuable insights into the syntactic structure of the sentences and allows for more advanced linguistic analysis, such as dependency parsing, semantic role labeling, and information extraction.

3.3.4 Unique Word Extraction

We extracted the unique words present in the dataset. This process plays a crucial role in understanding the vocabulary and linguistic characteristics of the text data. By identifying and analyzing the unique words, we gain insights into the distinct linguistic patterns and lexical richness exhibited by the dataset. To accomplish this task, we implemented a robust methodology for unique word extraction. We leveraged the power of Python programming language and utilized various libraries and techniques to ensure accurate and efficient extraction of unique words.

3.4 Feature Selection

We aimed to assign a unique identifier, or ID, to each of the extracted unique words. This assignment was carried out to facilitate a better understanding and representation of the textual data for the reinforcement learning (RL) agent, a key component of our analysis. By assigning a unique ID to each word, we created a consistent and standardized representation of the words within the dataset. This unique identifier serves as a compact numerical representation, allowing the RL agent to efficiently process and comprehend the textual information during its learning and decision-making processes. To accomplish this, we developed a systematic approach within our Python implementation. As we encountered each unique word in the dataset, we assigned it a unique identifier, ensuring that no other word received the same ID. This process involved maintaining a mapping or lookup table that associated each word with its respective ID. By assigning unique IDs to the words, we effectively transformed the text data into a numerical format that the RL agent can comprehend and utilize for its learning algorithms. This numerical representation enables the RL agent to operate with a reduced-dimensional input space, facilitating more efficient and effective learning and decision-making processes. The assignment of unique IDs to each word also enables the RL agent to establish associations between different words based on their respective IDs. This allows the agent to capture semantic relationships and contextual information within the dataset, aiding in its ability to understand and make informed decisions based on the textual data.

3.5 Proposed Model

In the problem of grammatical error correction, the goal is to automatically detect and correct grammatical errors in text, such as spelling mistakes, punctuation errors, and incorrect word usage. This task is crucial in natural language processing and text editing applications to enhance the clarity and correctness of written content.

Traditional approaches to grammatical error correction often rely on rule-based or statistical methods, which require extensive manual feature engineering or large annotated datasets. However, these approaches may struggle with handling the complexity and variability of grammatical errors in real-world text.

Deep reinforcement learning, specifically the DQN algorithm, offers a promising alternative for addressing grammatical error correction problems. Here are some motivations for using DQN:

Ability to Learn from Raw Text

Deep reinforcement learning, in combination with deep neural networks, has shown remarkable success in learning from raw sensory data, such as images or text. By directly processing the text as input, DQN can capture complex patterns and contextual information relevant to grammatical error correction without relying heavily on handcrafted features.

Sequential Decision-Making Framework

Grammatical error correction involves a sequence of actions that need to be taken in context to improve the correctness of the text. Reinforcement learning, with its emphasis on sequential decision-making, provides a natural framework for learning to correct errors in a step-by-step manner.

Exploration and Exploitation

DQN incorporates the exploration-exploitation trade-off, allowing the agent to explore different correction actions while gradually exploiting the learned knowledge. This is particularly useful in grammatical error correction, as the agent can learn from both correct and incorrect corrections, exploring different options to refine its correction strategy.

Learning from Reward Signals

Reinforcement learning enables learning from reward signals that indicate the quality or correctness of the agent's actions. In grammatical error correction, rewards can be designed to reflect the improvement in grammatical accuracy, coherence, or other linguistic properties. By optimizing for these rewards, the DQN algorithm can learn to make better corrections over time.

Adaptability and Generalization

DQN has the potential to adapt to different writing styles, genres, and error patterns by learning from diverse training data. This adaptability allows the model to generalize well and perform effectively on various types of text, making it a valuable tool for grammatical error correction across different domains.

By leveraging the advantages of deep reinforcement learning and the DQN algorithm, researchers and practitioners can explore new avenues for automating grammatical error correction, potentially improving the accuracy and efficiency of language editing tasks.

3.6 Environment Setup

GECToR{ GECToR – Grammatical Error Correction: Tag, Not Rewrite} [46] is an approach implemented in Pytorch for the purpose of grammatical error correction. Instead of completely rewriting sentences, GECToR works by identifying and tagging specific words or phrases that contain errors. The aim is to simplify the correction process by treating it as a sequence labeling problem and using a neural network model to automatically assign error tags to the corresponding words or phrases.

This dataset preparation step ensures that the data is organized and suitable for training and evaluating the GEC models. Further implementation steps are explained below.

This step involves setting up the environment to ensure the smooth execution of the project. The Conda package manager is utilized to create a dedicated environment for the project.

During the environment setup, all the necessary packages and submodules are installed to guarantee that the environment is equipped with the correct dependencies. This ensures compatibility among the installed packages and minimizes any potential compatibility issues during subsequent steps.

3.7 Supervised Learning (SL) Pre-Training

The methodology includes a pre-training phase using the supervised learning (SL) approach. A Sequence-to-Label model derived from GECToR is employed and pre-trained to initialize its parameters. The training script specific to the pre-training phase is used, and a configuration file guides the process. During this phase, the model is exposed to labeled data and learns the grammar correction patterns from the provided corrections. The SL pre-training establishes a solid foundation for subsequent fine-tuning using DRL techniques.

3.8 Supervised Learning (SL) Fine-tunning

After the pre-training phase, the next step is to fine-tune the GEC model using the supervised learning (SL) method. The fine-tuning process involves the utilization of specific scripts and configurations. In this phase, the model is trained on a larger amount of labeled data. The labeled data consists of original sentences with their corresponding corrected versions. The model learns from the corrections and adjusts its parameters to improve its grammar error correction capabilities. The SL fine-tuning stage aims to enhance the model's accuracy and effectiveness in correcting grammar errors by leveraging the knowledge acquired from the labeled data.

3.9 Deep Reinforcement Learning (DRL) Fine-tunning

The core of the methodology lies in applying Deep Reinforcement Learning (DRL) techniques to further refine and optimize the GEC model. In this phase, the GEC model is trained using a combination of reinforcement learning algorithms. The batched version of the REINFORCE algorithm is employed to optimize the model's parameters based on rewards and penalties received during training. Additionally, a simple action-search algorithm is used to facilitate safe exploration during RL training. This DRL fine-tuning process allows the model to iteratively improve its grammar error correction abilities, making it more proficient in identifying and rectifying grammar errors.

3.10 Evaluation

The final step of the methodology involves evaluating the performance of the trained GEC models. The models are evaluated using three established GEC benchmarks: CONLL-2014, BEA-2019, and JFLEG. The evaluation results provide valuable insights into the models' performance across different datasets and allow for a comprehensive analysis and comparison of their capabilities.

4 Results

4.1 Evaluation

The dataset was processed to create multiple chunks for analysis. The raw dataset was organized into chunks ranging from 10k sentences to 50k sentences. This process was applied to the entire dataset, resulting in separate CSV files for each chunk. These chunks were then used as input data for training and evaluating deep reinforcement learning models. Multiple models were trained using these chunks, and the results were compared for performance analysis. This approach enabled a comprehensive analysis of the impact of different chunks on the model's performance. The generated results were compared and analyzed to identify the optimal results for the specific task, providing valuable insights into the effect of temporal resolution on the performance of the deep reinforcement learning models.

4.2 Training on 10k Dataset

The DQN algorithm's reward value for 7000 episodes on a 10k sentences dataset. The reward graph shows the reward values in one graph to illustrate episode performance. The x-axis shows the episode number and the y-axis reward values. The graph illustrates reward value trends throughout training, revealing the model's reward performance. The graph shows data from 7000 episodes, offering a full view of the model's performance and learning trends in rewards, showing potential limitations or opportunities for optimization.

DQN's cumulative reward value on the 7000 episodes of training on a 10k sentences dataset. Episode number and cumulative reward values are on the axes. The graph shows the model's reward performance. Episode rewards show optimization and limitations.

The DQN algorithm's average reward value on 7000 episodes of training on a 10k sentences dataset. The average reward graph illustrates episode performance. Episode number and average reward values are on the axes. The graph shows the model's training reward performance. The graph illustrates the model's average reward performance and learning patterns.

Assuming that i is between 1 and 7000, let L_i stand for the loss value of the DQN algorithm on the i-th episode during training. The loss graph can be adjusted to show the minimum loss value by setting the lower limit of the y-axis to L_min, where L_min is the minimum value of L_i overall i. This allows for better visibility of the trends and patterns of loss values during training, highlighting episodes or time points with high loss values that may indicate potential issues or areas for improvement in the model. By setting the lower limit to L_min, the graph provides insights into the model's performance in terms of minimizing the loss during training, allowing for detailed monitoring and analysis of the model's convergence and optimization.

Table 4.1 provides a comprehensive analysis of the performance metrics achieved by the Deep Q-Network (DQN) model. The performance metrics evaluated in this analysis include the DQN Average Reward, DQN Cumulative Reward, and DQN Loss.

Table 4.1
Performance Metrics of the DQN Model

Metrics	Results
DQN Average Reward	2.469
DQN Cumulative Reward	8960.431
DQN Loss	2.493

4.3 Training on 20k Dataset

The DQN algorithm's reward value for 7000 episodes on a 20k sentences dataset. The reward graph shows the reward values in one graph to illustrate episode performance. The x-axis shows the episode number and the y-axis reward values. The graph illustrates reward value trends throughout training, revealing the model's reward performance. The graph shows data from 7000 episodes, offering a full view of the model's performance and learning trends in rewards, showing potential limitations or opportunities for optimization.

DQN's cumulative reward value on the 7000 episodes of training on a 20k sentences dataset. Episode number and cumulative reward values are on the axes. The graph shows the model's reward performance. Episode rewards show optimization and limitations.

The DQN algorithm's average reward value on 7000 episodes of training on 20k sentences dataset. The average reward graph illustrates episode performance. Episode number and average reward values are on the axes. The graph shows the model's training reward performance. The graph illustrates the model's average reward performance and learning patterns.

Assuming that i is between 1 and 7000, let L_i stand for the loss value of the DQN algorithm on the i-th episode during training. The loss graph can be adjusted to show the minimum loss value by setting the lower limit of the y-axis to L_min, where L_min is the minimum value of L_i overall i. This allows for better visibility of the trends and patterns of loss values during training, highlighting episodes or time points with high loss values that may indicate potential issues or areas for improvement in the model. By setting the lower limit to L_min, the graph provides insights into the model's performance in terms of minimizing the loss during training, allowing for detailed monitoring and analysis of the model's convergence and optimization.

Table 4.2 provides a comprehensive analysis of the performance metrics achieved by the Deep Q-Network (DQN) model. The performance metrics evaluated in this analysis include the DQN Average Reward, DQN Cumulative Reward, and DQN Loss.

Table 4.2
Performance Metrics of the DQN Model

Metrics	Results
DQN Average Reward	2.742
DQN Cumulative Reward	10686.418
DQN Loss	3.203

4.4 Training on 30k Dataset

The DQN algorithm's reward value for 7000 episodes on a 30k sentences dataset. The reward graph shows the reward values in one graph to illustrate episode performance. The x-axis shows the episode number and the y-axis reward values. The graph illustrates reward value trends throughout training, revealing the model's reward performance. The graph shows data from 7000 episodes, offering a full view of the model's performance and learning trends in rewards, showing potential limitations or opportunities for optimization.

DQN's cumulative reward value on the 7000 episodes of training on a 30k sentences dataset. Episode number and cumulative reward values are on the axes. The graph shows the model's reward performance. Episode rewards show optimization and limitations.

The DQN algorithm's average reward value on 7000 episodes of training on a 30k sentences dataset. The average reward graph illustrates episode performance. Episode number and average reward values are on the axes. The graph shows the model's training reward performance. The graph illustrates the model's average reward performance and learning patterns.

Assuming that i is between 1 and 7000, let L_i stand for the loss value of the DQN algorithm on the i-th episode during training. The loss graph can be adjusted to show the minimum loss value by setting the lower limit of the y-axis to L_min, where L_min is the minimum value of L_i overall i. This allows for better visibility of the trends and patterns of loss values during training, highlighting episodes or time points with high loss values that may indicate potential issues or areas for improvement in the model. By setting the lower limit to L_min, the graph provides insights into the model's performance in terms of minimizing the loss during training, allowing for detailed monitoring and analysis of the model's convergence and optimization.

Table 4.3 provides a comprehensive analysis of the performance metrics achieved by the Deep Q-Network (DQN) model. The performance metrics evaluated in this analysis include the DQN Average Reward, DQN Cumulative Reward, and DQN Loss.

Table 4.3
Performance Metrics of the DQN Model

Metrics	Results
DQN Average Reward	4.784
DQN Cumulative Reward	14639.138
DQN Loss	5.310

4.5 Training on 40k Dataset

The DQN algorithm's reward value for 7000 episodes on a 40k sentences dataset. The reward graph shows the reward values in one graph to illustrate episode performance. The x-axis shows the episode number and the y-axis reward values. The graph illustrates reward value trends throughout training, revealing the model's reward performance. The graph shows data from 7000 episodes, offering a full view of the model's performance and learning trends in rewards, showing potential limitations or opportunities for optimization.

DQN's cumulative reward value on the 7000 episodes of training on a 40k sentences dataset. Episode number and cumulative reward values are on the axes. The graph shows the model's reward performance. Episode rewards show optimization and limitations.

The DQN algorithm's average reward value on 7000 episodes of training on a 40k sentences dataset. The average reward graph illustrates episode performance. Episode number and average reward values are on the axes. The graph shows the model's training reward performance. The graph illustrates the model's average reward performance and learning patterns.

Assuming that i is between 1 and 7000, let L_i stand for the loss value of the DQN algorithm on the i-th episode during training. The loss graph can be adjusted to show the minimum loss value by setting the lower limit of the y-axis to L_min, where L_min is the minimum value of L_i overall i. This allows for better visibility of the trends and patterns of loss values during training, highlighting episodes or time points with high loss values that may indicate potential issues or areas for improvement in the model. By setting the lower limit to L_min, the graph provides insights into the model's performance in terms of minimizing the loss during training, allowing for detailed monitoring and analysis of the model's convergence and optimization.

Table 4.4 provides a comprehensive analysis of the performance metrics achieved by the Deep Q-Network (DQN) model. The performance metrics evaluated in this analysis include the DQN Average Reward, DQN Cumulative Reward, and DQN Loss.

Table 4.4
Performance Metrics of the DQN Model

Metrics	Results
DQN Average Reward	3.831
DQN Cumulative Reward	17969.184
DQN Loss	4.310

4.6 Training on 50k Dataset

The DQN algorithm's reward value for 7000 episodes on a 50k sentences dataset. The reward graph shows the reward values in one graph to illustrate episode performance. The x-axis shows the episode number and the y-axis reward values. The graph illustrates reward value trends throughout training, revealing the model's reward performance. The graph shows data from 7000 episodes, offering a full view of the model's performance and learning trends in rewards, showing potential limitations or opportunities for optimization.

DQN's cumulative reward value on the 7000 episodes of training on a 50k sentences dataset. Episode number and cumulative reward values are on the axes. The graph shows the model's reward performance. Episode rewards show optimization and limitations.

The DQN algorithm's average reward value on 7000 episodes of training on a 50k sentences dataset. The average reward graph illustrates episode performance. Episode number and average reward values are on the axes. The graph shows the model's training reward performance. The graph illustrates the model's average reward performance and learning patterns.

Assuming that i is between 1 and 7000, let L_i stand for the loss value of the DQN algorithm on the i-th episode during training. The loss graph can be adjusted to show the minimum loss value by setting the lower limit of the y-axis to L_min, where L_min is the minimum value of L_i overall i. This allows for better visibility of the trends and patterns of loss values during training, highlighting episodes or time points with high loss values that may indicate potential issues or areas for improvement in the model. By setting the lower limit to L_min, the graph provides insights into the model's performance in terms of minimizing the loss during training, allowing for detailed monitoring and analysis of the model's convergence and optimization.

Table 4.5 provides a comprehensive analysis of the performance metrics achieved by the Deep Q-Network (DQN) model. The performance metrics evaluated in this analysis include the DQN Average Reward, DQN Cumulative Reward, and DQN Loss.

Table 4.5
Performance Metrics of the DQN Model

Metrics	Results
DQN Average Reward	4.937
DQN Cumulative Reward	14923.533
DQN Loss	4.653

Table 4.6 showcases the statistics of rewards and losses generated by the DQN model during its training process on various dataset chunks. The table specifically highlights the DQN Reward for different dataset sizes, namely 10k, 20k, 30k, 40k, and 50k.

Table 4.6
Result Statistics

	DQN Reward of 10k Dataset	DQN Reward of 20k Dataset	DQN Reward of 30k Dataset	DQN Reward of 40k Dataset	DQN Reward of 50k Dataset
Reward	2.469	2.742	4.784	3.831	**5.937**
Loss	2.493	3.203	5.310	4.310	4.653

5 Conclusion and Future Work

5.1 Conclusion

Expanding on this observation, we can infer that a larger dataset provides the DQN model with more diverse and representative examples, allowing it to learn more effectively and make better predictions. The increased dataset size likely exposes the model to a broader range of scenarios and variations, enabling it to generalize better to unseen situations.

Therefore, based on the findings it is reasonable to conclude that further increasing the dataset beyond 50k would likely result in continued performance improvement for the DQN model. This suggests that collecting additional data and augmenting the training set could be a beneficial strategy for enhancing the model's capabilities.

By training a DRL model for automatic grammatical error correction, the proposed DQN model demonstrated superior performance compared to traditional machine learning and rule-based techniques. The utilization of RL techniques in GEC showcased its potential to enhance the accuracy and effectiveness of error correction in written text. Through rigorous experimentation and evaluation, the effectiveness and superiority of the proposed DQN model were established, indicating its potential for real-world applications in automated grammatical error correction.

These findings contribute to the field of NLP and GEC by showcasing the feasibility and benefits of employing reinforcement learning techniques, particularly DQN models, in automating the error correction process. This research opens up new avenues for further exploration, such as incorporating additional datasets, exploring advanced RL algorithms, and conducting in-depth error analysis to address specific challenges in GEC.

5.2 Future Work

Future work in the field of grammatical error correction using deep reinforcement learning encompasses several key areas for further exploration and advancement. One avenue is the incorporation of additional datasets or domain-specific data to augment the RL agent's grammar error correction capabilities. By exposing the agent to a wider range of linguistic contexts and domain-specific language patterns, its ability to accurately identify and correct grammar errors can be further improved.

Domain-specific fine-tuning is another promising area to pursue. By tailoring the RL agent's training on specific domains or genres of text, such as academic writing or technical documents, its accuracy and effectiveness in specialized contexts can be significantly improved. The exploration of online learning methods is also worth considering. By enabling the RL agent to continuously learn and adapt from real-time user interactions and feedback, it can dynamically enhance its error correction capabilities and better align with user preferences and needs. To assess the quality and efficacy of the grammar error correction system, thorough human evaluations should be conducted. These evaluations can provide valuable insights by comparing the performance of the RL agent against human-level correction, thereby guiding further refinements and improvements.

Furthermore, integration into existing writing tools or the development of standalone applications can extend the practical utility of the RL agent. By seamlessly integrating grammar error correction capabilities into the writing process, users can benefit from real-time assistance and refinement. Lastly, expanding the scope to include multilingual support is a compelling avenue for future research. Adapting the approach to handle grammar error correction in multiple languages, while considering language-specific challenges and variations, would broaden the reach and applicability of the system.

In summary, by incorporating additional data, exploring advanced algorithms, conducting error analysis, fine-tuning for specific domains, enabling online learning, performing human evaluations, integrating with writing tools, and expanding to multiple languages, future research in grammatical error correction using deep reinforcement learning can contribute to the development of more robust, effective, and versatile systems for improving grammatical accuracy in written communication.

Declarations

Ethical Approval

No ethical approval needed because no human or animals were involved during the research.

Competing interests

No competing interests of a financial or personal nature

Authors' contributions

M.H.A. contributed to the initial draft of the paper and conducted the experiments. R.A. contributed to the Python code implementation and drafting of the paper. M.H. contributed to developing the overall concept and methodology. M.F. contributed to the methodology design, supervised the work, and reviewed the paper. F.A.A. reviewed the paper and provided resources. M.M.A. collaborated on the work, reviewed the paper, and provided feedback. All authors reviewed the final manuscript.

Funding

Researchers Supporting Project number (RSPD2023R838), King Saud University, Riyadh, Saudi Arabia.

Availability of data and materials

Data and materials are available upon reasonable request to corresponding author.

References

1. J. P. Kincaid and S. Schalow, “The Computer Readability Editing System,” *http://dx.doi.org/10.1177/154193128502900519*, vol. 29, no. 5, pp. 489–493, Oct. 1985, doi: 10.1177/154193128502900519.
2. G. E. Heidorn, K. Jensen, L. A. Miller, R. J. Byrd, and M. S. Chodorow, “The EPISTLE text-critiquing system,” *IBM Systems Journal*, vol. 21, no. 3, pp. 305–327, Sep. 1982, doi: 10.1147/SJ.213.0305.
3. S. D. Richardson and L. C. Braden-Harder, “The Experience of Developing a Large-Scale Natural Language Text Processing System: Critique,” pp. 195–202, 1988, doi: 10.3115/974235.974271.
4. F. R. Bustamante and F. S&nchez, “GramCheck: A Grammar and Style Checker.”
5. G. M. Correia and A. F. T. Martins, “A Simple and Effective Approach to Automatic Post-Editing with Transfer Learning,” *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pp. 3050–3056, 2019, doi: 10.18653/V1/P19-1292.
6. N.-R. Han, M. Chodorow, and C. Leacock, “Detecting Errors in English Article Usage with a Maximum Entropy Classifier Trained on a Large, Diverse Corpus.” 2004. Accessed: May 15, 2023. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2004/pdf/695.pdf
7. M. Chodorow, J. R. Tetreault, and N.-R. Han, “Detection of Grammatical Errors Involving Prepositions.” Association for Computational Linguistics, pp. 25–30, 2007. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/W07-1604
8. R. De Felice and S. G. Pulman, “Automatically Acquiring Models of Preposition Use.” pp. 45–50, 2007. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/W07-1607

9. J. R. Tetreault and M. Chodorow, “The Ups and Downs of Preposition Error Detection in ESL Writing.” Manchester, pp. 865–872, 2008. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/C08-1109
10. A. Rozovskaya and D. Roth, “Algorithm Selection and Model Adaptation for ESL Correction Tasks.” pp. 924–933, 2011. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/P11-1093
11. D. Dahlmeier, H. T. Ng, E. Jun, and F. Ng, “NUS at the HOO 2012 Shared Task.” pp. 216–224, 2012. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/W12-2025
12. M. Felice, “Number 895 Artificial error generation for translation-based grammatical error correction,” 2016, [Online]. Available: http://www.cl.cam.ac.uk/
13. J. Lee and S. Seneff, “An analysis of grammatical errors in non-native speech in English,” *2008 IEEE Workshop on Spoken Language Technology, SLT 2008 - Proceedings*, pp. 89–92, 2008, doi: 10.1109/SLT.2008.4777847.
14. A. Rozovskaya, K.-W. Chang, M. Sammons, and D. Roth, “The University of Illinois System in the CoNLL-2013 Shared Task.” pp. 13–19, 2013. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/W13-3602
15. A. Rozovskaya and D. Roth, “Joint Learning and Inference for Grammatical Error Correction.” pp. 791–802, 2013. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/D13-1074
16. D. Dahlmeier and H. T. Ng, “A Beam-Search Decoder for Grammatical Error Correction.” Association for Computational Linguistics, pp. 568–578, 2012. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/D12-1052
17. Y. Wu and H. T. Ng, “Grammatical Error Correction Using Integer Linear Programming.” pp. 1456–1465, 2013. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/P13-1143
18. M. Gamon *et al.*, “Using Contextual Speller Techniques and Language Modeling for ESL Error Correction.” 2008. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/I08-1059
19. M. Gamon, “High-Order Sequence Modeling for Language Learner Error Detection.” pp. 180–189, 2011. Accessed: May 15, 2023. [Online]. Available: https://aclanthology.org/W11-1422
20. A. Rozovskaya, D. Roth, and V. Srikumar, “Correcting grammatical verb errors,” *14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014*, pp. 358–367, 2014, doi: 10.3115/V1/E14-1038.
21. H. Xue and R. Hwa, “Redundancy Detection in ESL Writings,” *14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014*, pp. 683–691, 2014, doi: 10.3115/V1/E14-1072.
22. A. Herbelot and E. Kochmar, “‘Calling on the classical phone’: a distributional model of adjective-noun errors in learners’ English.” [Online]. Available: http://ilexir.co.uk/applications/adjective-noun-dataset/
23. C. Universit and S. Germany, “Definiteness Predictions for Japanese N o u n Phrases *,” *Technology (Singap World Sci)*, pp. 519–525, 1993.

24. M. Murata, M. N.- arXiv preprint cmp-lg/9405019, and undefined 1994, “Determination of referential property and number of nouns in Japanese sentences for machine translation into English,” *arxiv.org*, [Online]. Available: https://arxiv.org/abs/cmp-lg/9405019
25. N. R. Han, M. Chodorow, and C. Leacock, “Detecting errors in English article usage by non-native speakers,” *Nat Lang Eng*, vol. 12, no. 2, pp. 115–129, May 2006, doi: 10.1017/S1351324906004190.
26. K. Knight, I. C.- AAAI, and undefined 1994, “Automated postediting of documents,” *aaai.org*, 1994, [Online]. Available: https://www.aaai.org/Papers/AAAI/1994/AAAI94-119.pdf
27. J. Tetreault, M. C.-P. of the 22nd International, and undefined 2008, “The ups and downs of preposition error detection in ESL writing,” *aclanthology.org*, [Online]. Available: https://aclanthology.org/C08-1109.pdf
28. D. Dahlmeier, H. T. N.-P. of the 49th annual meeting of, and undefined 2011, “Grammatical error correction with alternating structure optimization,” *aclanthology.org*, [Online]. Available: https://aclanthology.org/P11-1092.pdf
29. M. G.-H. L. T. T. 2010 Annual and undefined 2010, “Using mostly native data to correct errors in learners’ writing,” *aclanthology.org*, pp. 163–171, [Online]. Available: https://aclanthology.org/N10-1019.pdf
30. J. Tetreault, J. Foster, and M. Chodorow, “Using parse features for preposition selection and error detection,” 2010, [Online]. Available: https://doras.dcu.ie/15989/
31. A. Rozovskaya, D. R.-P. of the 49th Annual Meeting, and undefined 2011, “Algorithm selection and model adaptation for ESL correction tasks,” *aclanthology.org*, pp. 924–933, [Online]. Available: https://aclanthology.org/P11-1093.pdf
32. X. Liu, B. Han, K. Li, S. H. S.-P. of the 2010 …, and undefined 2010, “SRL-based verb selection for ESL,” *aclanthology.org*, [Online]. Available: https://aclanthology.org/D10-1104.pdf
33. T. Tajiri, M. Komachi, Y. M.- of the 50th Annual Meeting of …, and undefined 2012, “Tense and aspect error correction for ESL learners using global context,” *aclanthology.org*, pp. 8–14, 2012, [Online]. Available: https://aclanthology.org/P12-2039.pdf
34. M. G.-P. of the Sixth Workshop on Innovative Use and undefined 2011, “High-order sequence modeling for language learner error detection,” *aclanthology.org*, pp. 180–189, [Online]. Available: https://aclanthology.org/W11-1422.pdf
35. G. Sidorov, A. Gupta, M. Tozer, D. Català, A. Catena, and S. Fuentes, “Rule-based System for Automatic Grammar Correction Using Syntactic N-grams for English Language Learning (L2).” pp. 96–101, 2013. [Online]. Available: https://aclanthology.org/W13-3613
36. M. Felice, Z. Yuan, Ø. E. Andersen, H. Yannakoudakis, and E. Kochmar, “Grammatical error correction using hybrid systems and type filtering,” *CoNLL 2014 - 18th Conference on Computational Natural Language Learning, Proceedings of the Shared Task*, pp. 15–24, 2014, doi: 10.3115/V1/W14-1702.
37. M. Kaneko, M. Mita, S. Kiyono, J. Suzuki, and K. Inui, “Encoder-Decoder Models Can Benefit from Pre-trained Masked Language Models in Grammatical Error Correction,” *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pp. 4248–4254, 2020, doi: 10.18653/V1/2020.ACL-MAIN.391.

38. W. Zhao, L. Wang, K. Shen, R. Jia, and J. Liu, "Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 156–165, 2019, doi: 10.18653/V1/N19-1014.
39. Z. Xie, G. Genthial, S. Xie, A. Y. Ng, and D. Jurafsky, "Noising and denoising natural language: Diverse backtranslation for grammar correction," *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 619–628, 2018, doi: 10.18653/V1/N18-1057.
40. W. Zhou, T. Ge, C. Mu, K. Xu, F. Wei, and M. Zhou, "Improving grammatical error correction with machine translation pairs," *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, pp. 318–328, 2020, doi: 10.18653/V1/2020.FINDINGS-EMNLP.30.
41. H. Ren, L. Yang, and E. Xun, "A Sequence to Sequence Learning for Chinese Grammatical Error Correction," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11109 LNAI, pp. 401–410, 2018, doi: 10.1007/978-3-319-99501-4_36.
42. J. Zhou, C. Li, H. Liu, Z. Bao, G. Xu, and L. Li, "Chinese Grammatical Error Correction Using Statistical and Neural Models," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11109 LNAI, pp. 117–128, 2018, doi: 10.1007/978-3-319-99501-4_10.
43. E. Malmi, S. Krause, S. Rothe, D. Mirylenka, and A. Severyn, "Encode, tag, realize: High-precision text editing," *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 5054–5065, 2019, doi: 10.18653/V1/D19-1510.
44. A. Awasthi, S. Sarawagi, R. Goyal, S. Ghosh, and V. Piratla, "Parallel iterative edit models for local sequence transduction," *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 4260–4270, 2019, doi: 10.18653/V1/D19-1435.
45. F. Stahlberg and S. Kumar, "Seq2Edits: Sequence transduction using span-level edit operations," *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 5147–5159, 2020, doi: 10.18653/V1/2020.EMNLP-MAIN.418.
46. K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi, "GECToR - Grammatical error correction: Tag, not rewrite," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 163–170, 2020, doi: 10.18653/V1/2020.BEA-1.16.
47. S. Wang *et al.*, "Combining ResNet and Transformer for Chinese Grammatical Error Diagnosis." pp. 36–43, 2020. Accessed: Jun. 20, 2023. [Online]. Available: https://aclanthology.org/2020.nlptea-1.5

48. D. Watson, N. Zalmout, and N. Habash, “Utilizing Character and Word Embeddings for Text Normalization with Sequence-to-Sequence Models,” *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pp. 837–843, 2018, doi: 10.18653/V1/D18-1097.
49. S. Ahmadi, J. Le Roux, and N. Tomeh, “Attention-based Encoder-Decoder Networks for Spelling and Grammatical Error Correction,” Sep. 2018, Accessed: Jun. 20, 2023. [Online]. Available: https://arxiv.org/abs/1810.00660v1
50. N. Zalmout and N. Habash, “Don’t Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic,” *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 704–713, 2017, doi: 10.18653/V1/D17-1073.
51. M. N. Ibrahim and M. M. Ragheb, “CUFE@QALB-2015 Shared Task: Arabic Error Correction System,” *2nd Workshop on Arabic Natural Language Processing, ANLP 2015 - held at 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015 - Proceedings*, pp. 133–137, 2015, doi: 10.18653/V1/W15-3215.
52. W. Zhao, L. Wang, K. Shen, R. Jia, and J. Liu, “Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data,” pp. 156–165, 2019, doi: 10.18653/V1/N19-1014.
53. S. Chollampatt and H. T. Ng, “A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction,” *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 5755–5762, Jan. 2018, doi: 10.1609/aaai.v32i1.12069.
54. S. Chollampatt, K. Taghipour, and H. T. Ng, “Neural Network Translation Models for Grammatical Error Correction,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2016-January, pp. 2768–2774, Jun. 2016, Accessed: Jun. 20, 2023. [Online]. Available: https://arxiv.org/abs/1606.00189v1
55. R. Grundkiewicz, M. Junczys-Dowmunt, and K. Heafield, “Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data,” *ACL 2019 - Innovative Use of NLP for Building Educational Applications, BEA 2019 - Proceedings of the 14th Workshop*, pp. 252–263, 2019, doi: 10.18653/V1/W19-4427.
56. M. Kaneko, Y. Sakaizawa, and M. Komachi, “Grammatical Error Detection Using Error- and Grammaticality-Specific Word Embeddings.” pp. 40–48, 2017. Accessed: Jun. 20, 2023. [Online]. Available: https://aclanthology.org/I17-1005
57. A. Schmaltz, Y. Kim, A. M. Rush, and S. M. Shieber, “Adapting Sequence Models for Sentence Correction,” *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 2807–2813, 2017, doi: 10.18653/V1/D17-1298.
58. S. Katsumata and M. Komachi, “Stronger Baselines for Grammatical Error Correction Using Pretrained Encoder-Decoder Model,” *Journal of Natural Language Processing*, vol. 28, no. 1, pp. 276–280, May 2020, doi: 10.5715/jnlp.28.276.

59. R. Dale, A. K.-P. of the 13th European Workshop, and undefined 2011, “Helping our own: The HOO 2011 pilot shared task,” *aclanthology.org*, pp. 242–249, [Online]. Available: https://aclanthology.org/W11-2838.pdf
60. R. Dale, I. Anisimoff, and G. Narroway, “HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task.” pp. 54–62, 2012. Accessed: May 16, 2023. [Online]. Available: https://aclanthology.org/W12-2006
61. H. T. Ng, S. M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault, “The CoNLL-2013 Shared Task on Grammatical Error Correction.” pp. 1–12, 2013. Accessed: May 16, 2023. [Online]. Available: https://aclanthology.org/W13-3601
62. H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, “The CoNLL-2014 Shared Task on Grammatical Error Correction,” *CoNLL 2014 - 18th Conference on Computational Natural Language Learning, Proceedings of the Shared Task*, pp. 1–14, 2014, doi: 10.3115/V1/W14-1701.
63. K. W. Church and W. A. Gale, “Probability scoring for spelling correction,” *Stat Comput*, vol. 1, no. 2, pp. 93–103, May 1991, doi: 10.1007/BF01889984.
64. M. Kim, S. K. Choi, J. Jin, H. C. K.-2015 I. International, and undefined 2015, “Adaptive context-sensitive spelling error correction techniques for the extremely unpredictable error generating language environments,” *ieeexplore.ieee.org*, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7363179/
65. J.-H. Lee, M. Kim, and H.-C. Kwon, “Improved Statistical Language Model for Context-sensitive Spelling Error Candidates,” *Journal of Korea Multimedia Society*, vol. 20, no. 2, pp. 371–381, May 2017, doi: 10.9717/KMMS.2017.20.2.371.
66. E. Mays, F. J. Damerau, and R. L. Mercer, “Context based spelling correction,” *Inf Process Manag*, vol. 27, no. 5, pp. 517–522, 1991, doi: 10.1016/0306-4573(91)90066-U.
67. M. Kim, H.-C. Kwon, and S. Choi, “Context-sensitive Spelling Error Correction using Eojeol N-gram,” *Journal of KIISE*, vol. 41, no. 12, pp. 1081–1089, May 2014, doi: 10.5626/JOK.2014.41.12.1081.
68. H. Li, Y. Wang, X. Liu, Z. Sheng, and S. Wei, “Spelling Error Correction Using a Nested RNN Model and Pseudo Training Data,” May 2018, [Online]. Available: http://arxiv.org/abs/1811.00238
69. H. Gong, Y. Li, S. Bhat, and P. Viswanath, “Context-Sensitive Malicious Spelling Error Correction,” May 2019, [Online]. Available: http://arxiv.org/abs/1901.07688
70. “Context-sensitive spelling error correction techniques... - Google Scholar.” https://scholar.google.com/scholar?as_q=Context-sensitive+spelling+error+correction+techniques+using+contextual+embeddings&a (accessed May 14, 2023).

Figures

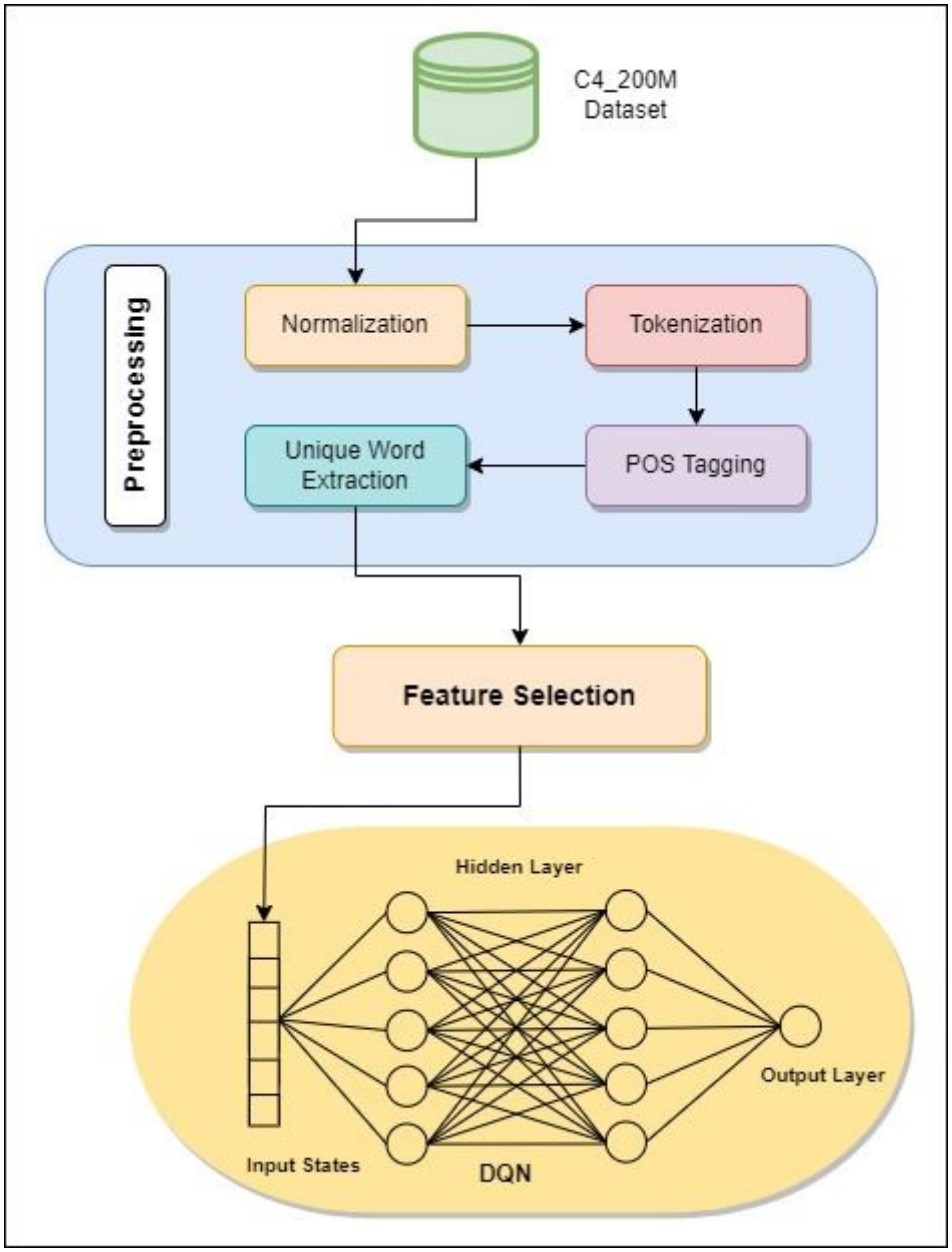

Figure 1

Detailed architecture of proposed methodology

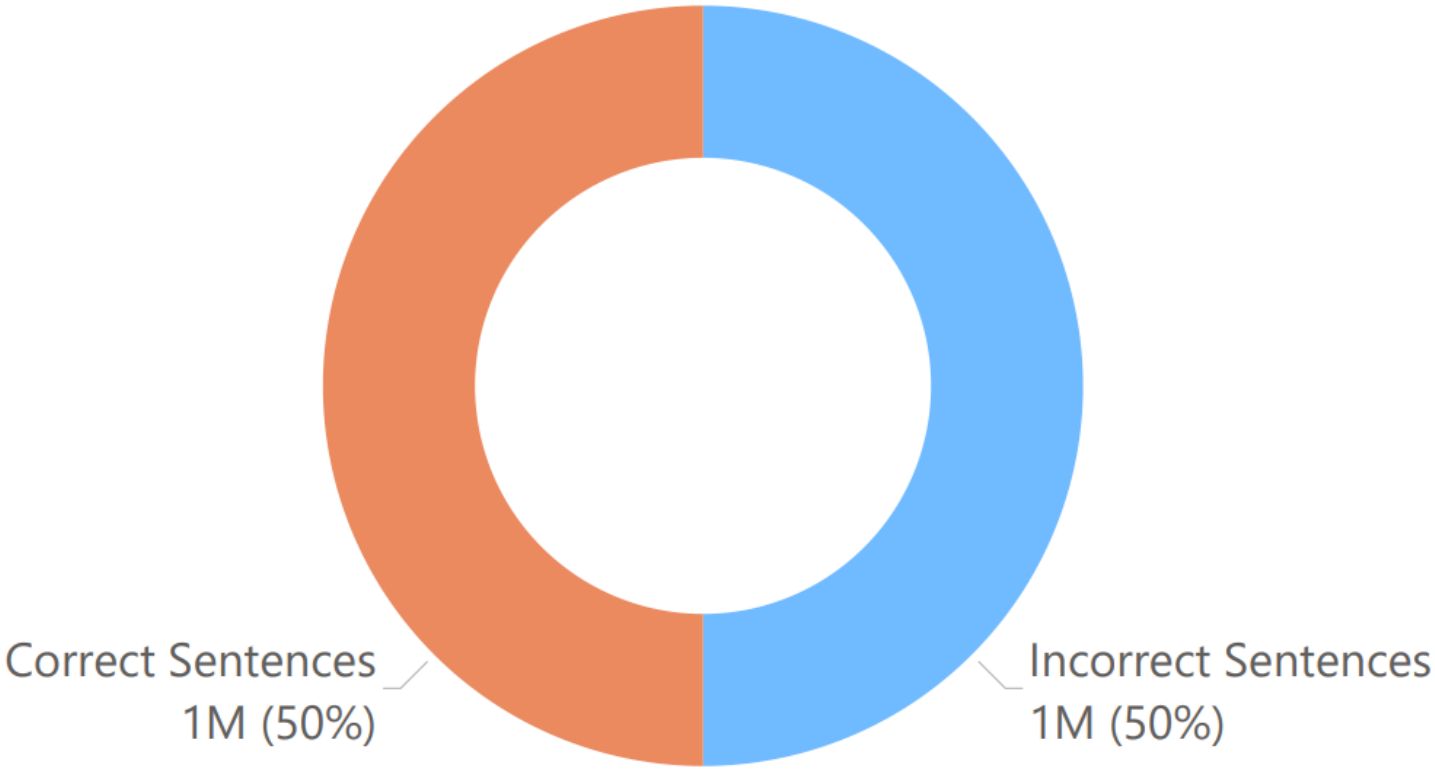

Figure 2

Distribution of Correct and Incorrect Sentences

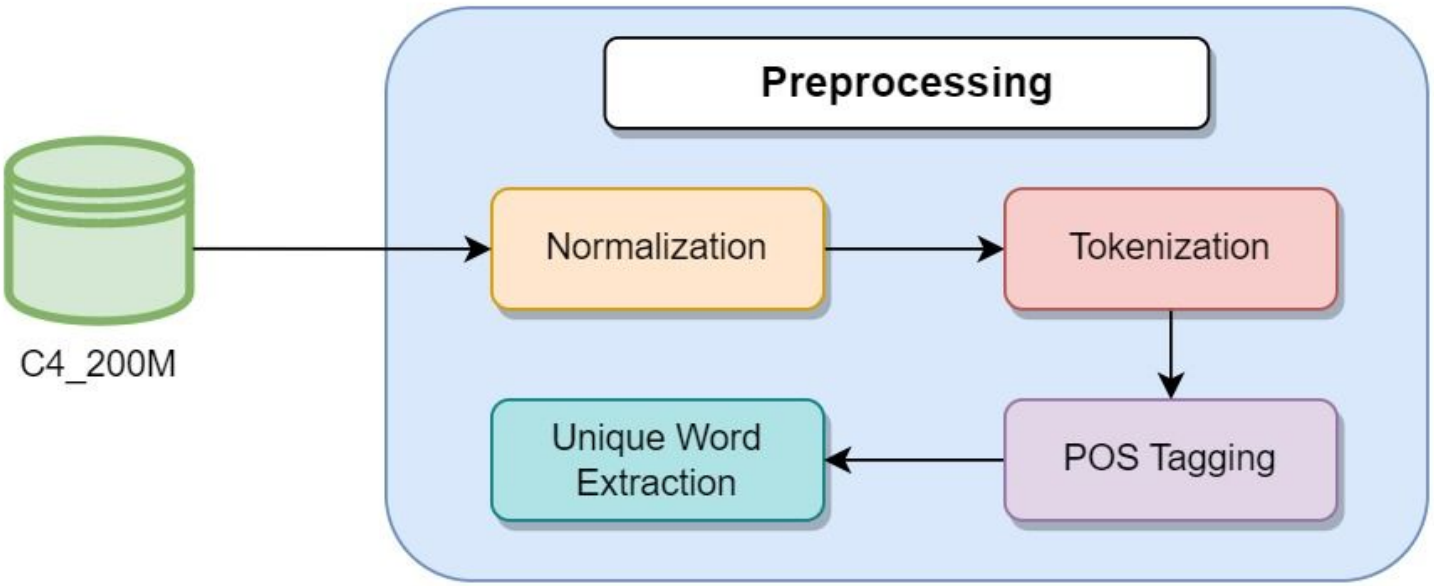

Figure 3

Detailed Architecture of Preprocessing

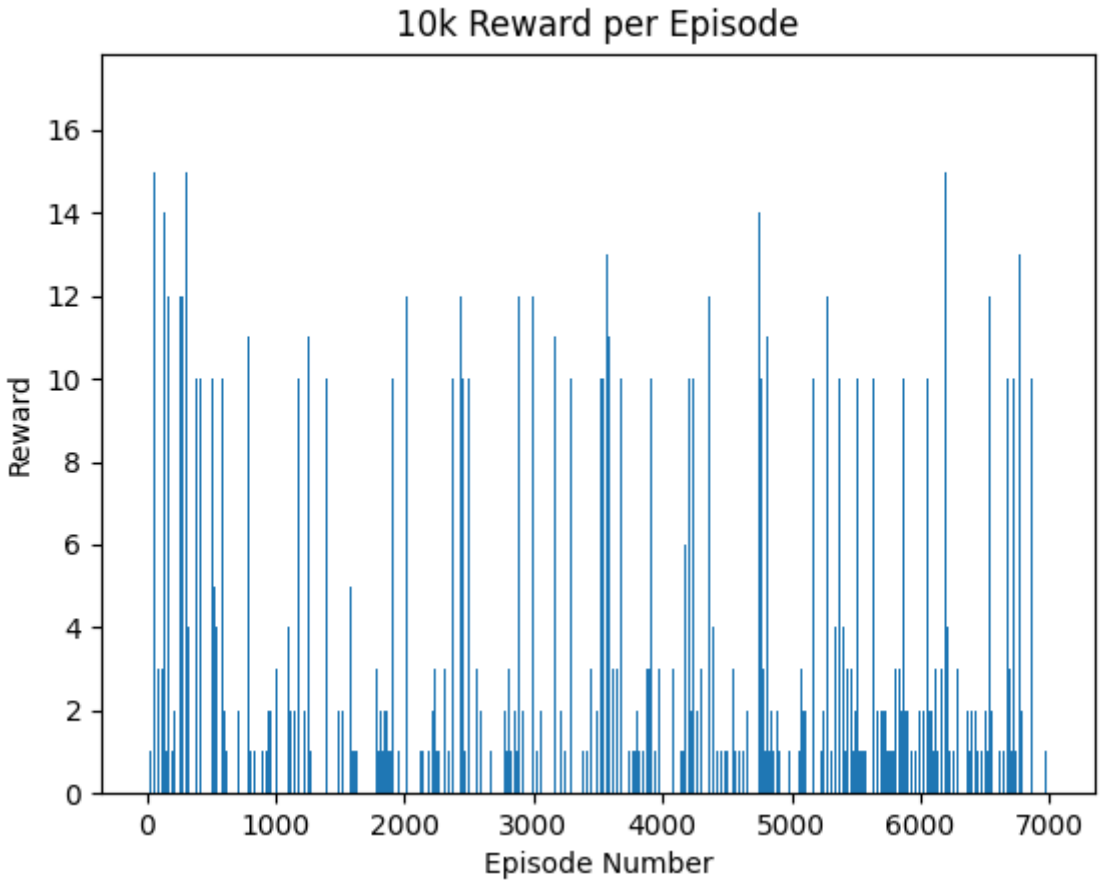

Figure 4

Figure 4.1 10k Reward Values Fluctuations during DQN Algorithm Training

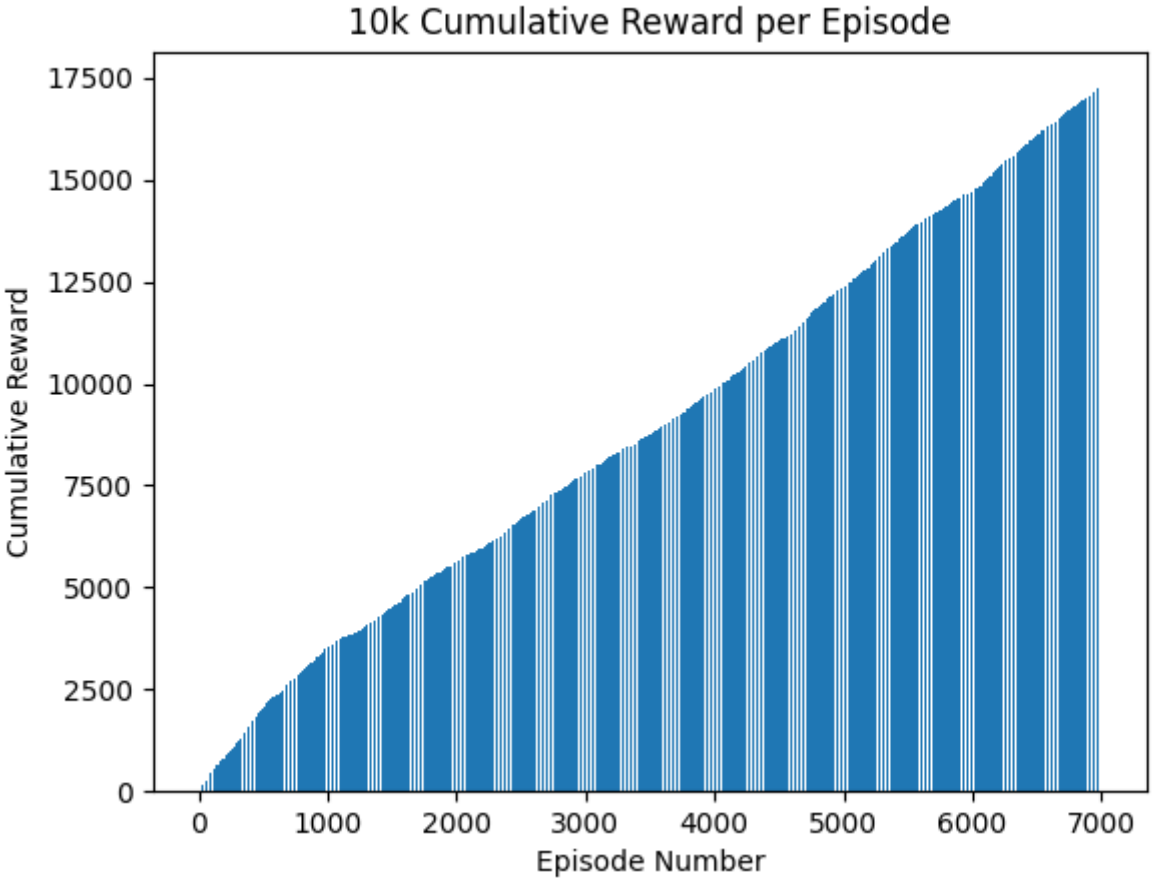

Figure 5

Figure 4.2 10k Cumulative Reward

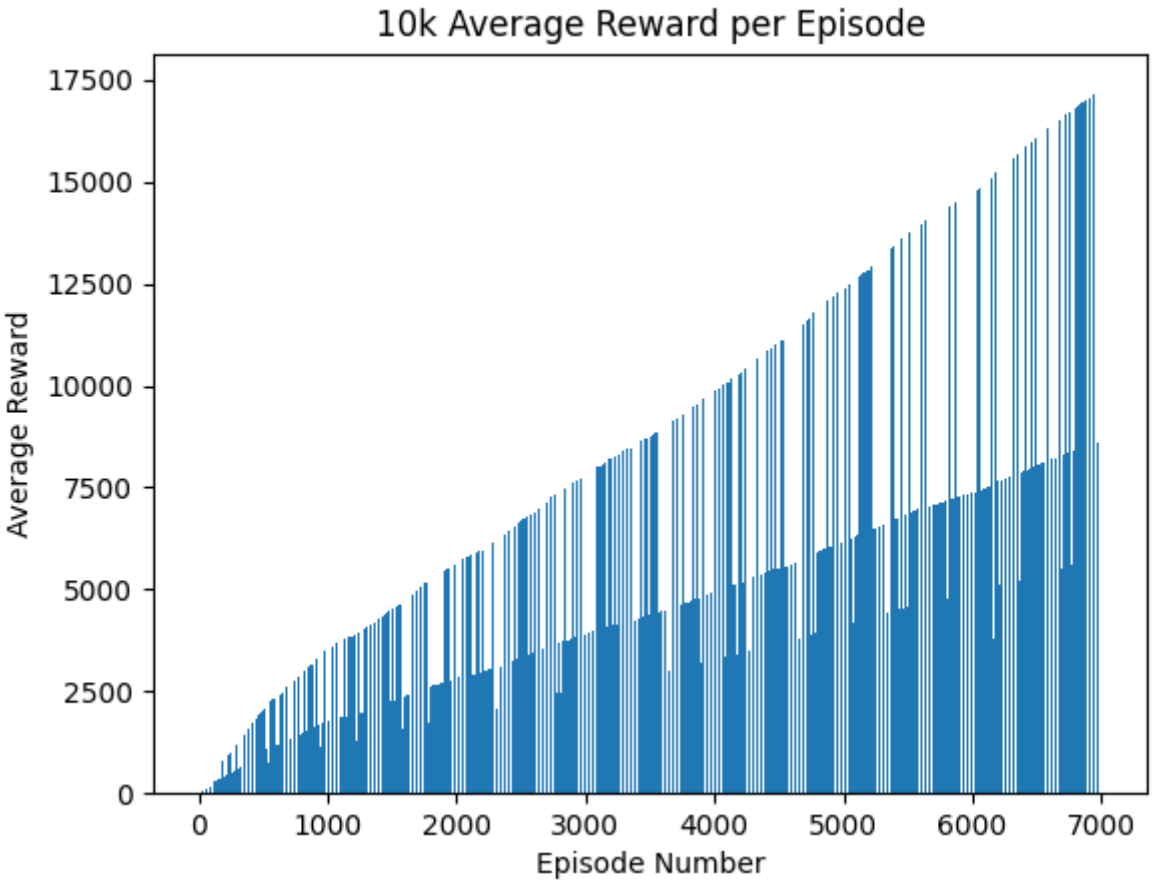

Figure 6

Figure 4.3 10k Average Reward

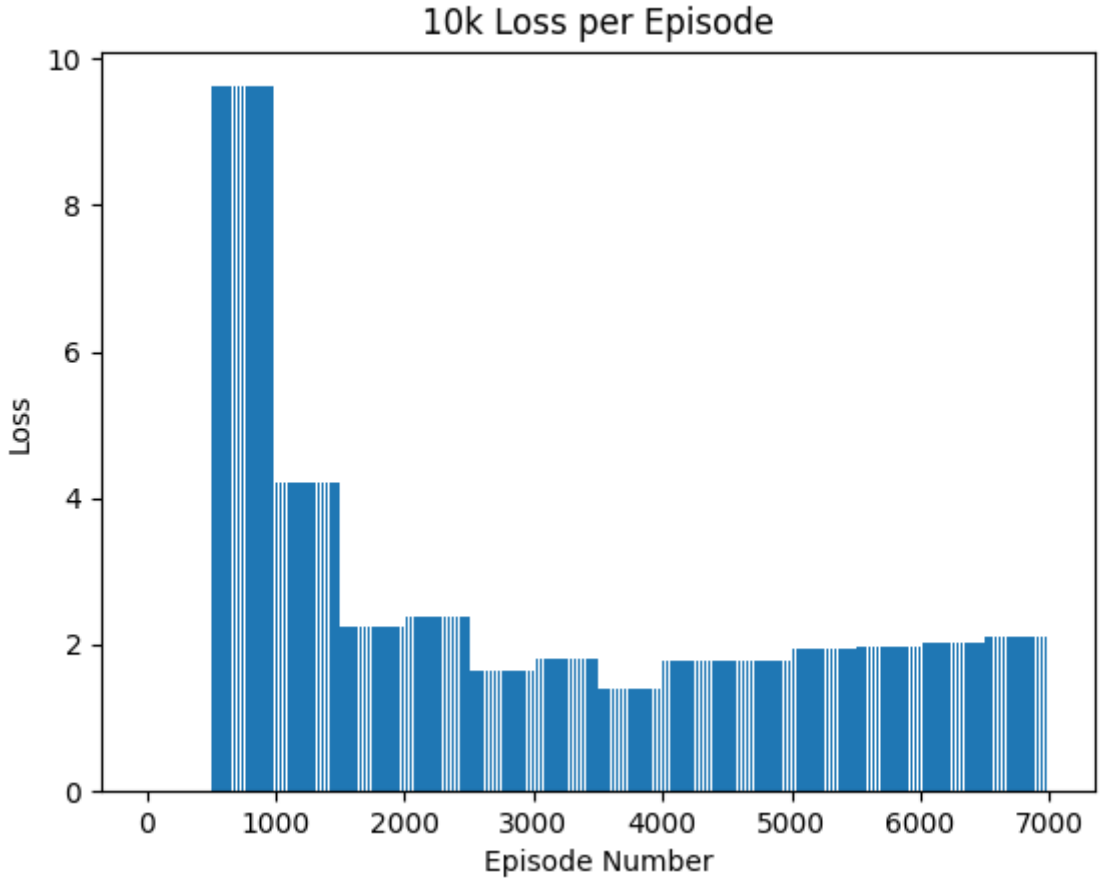

Figure 7

Figure 4.4 10k Loss

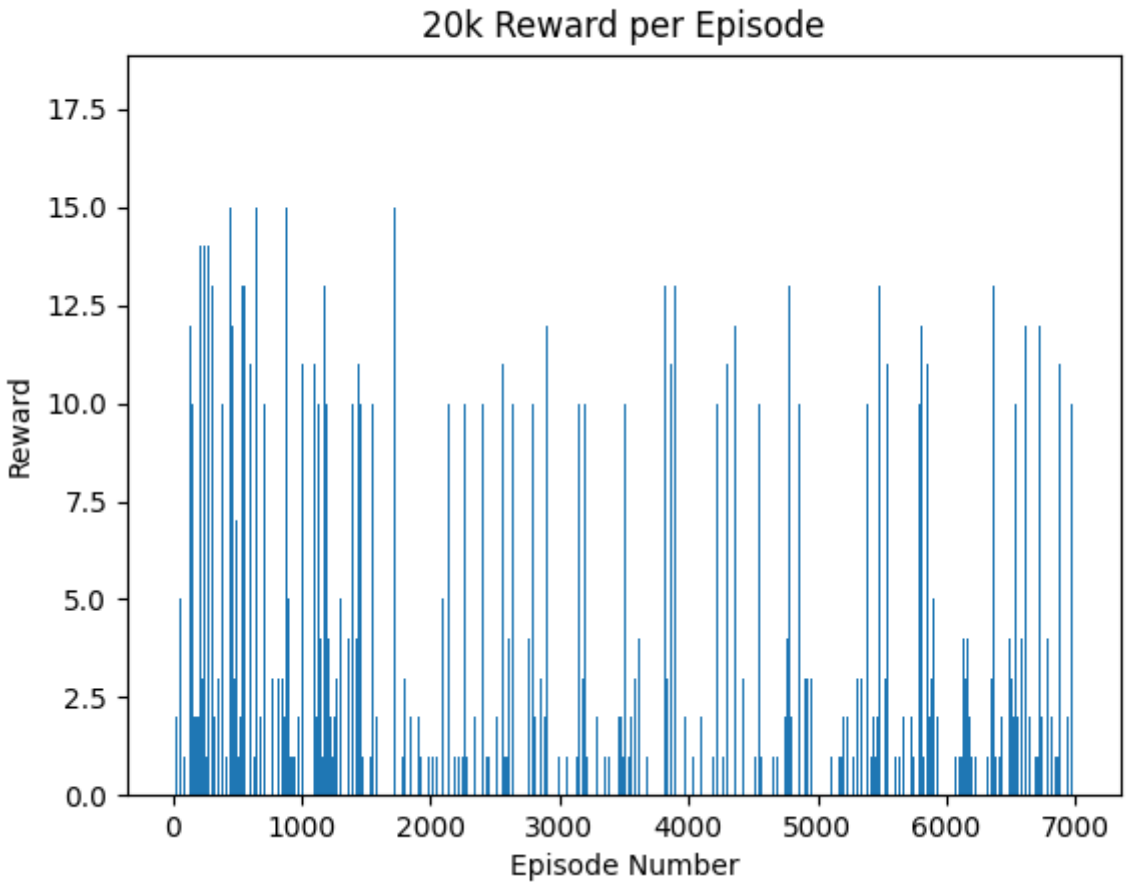

Figure 8

Figure 4.5 20k Reward Values Fluctuations during DQN Algorithm Training

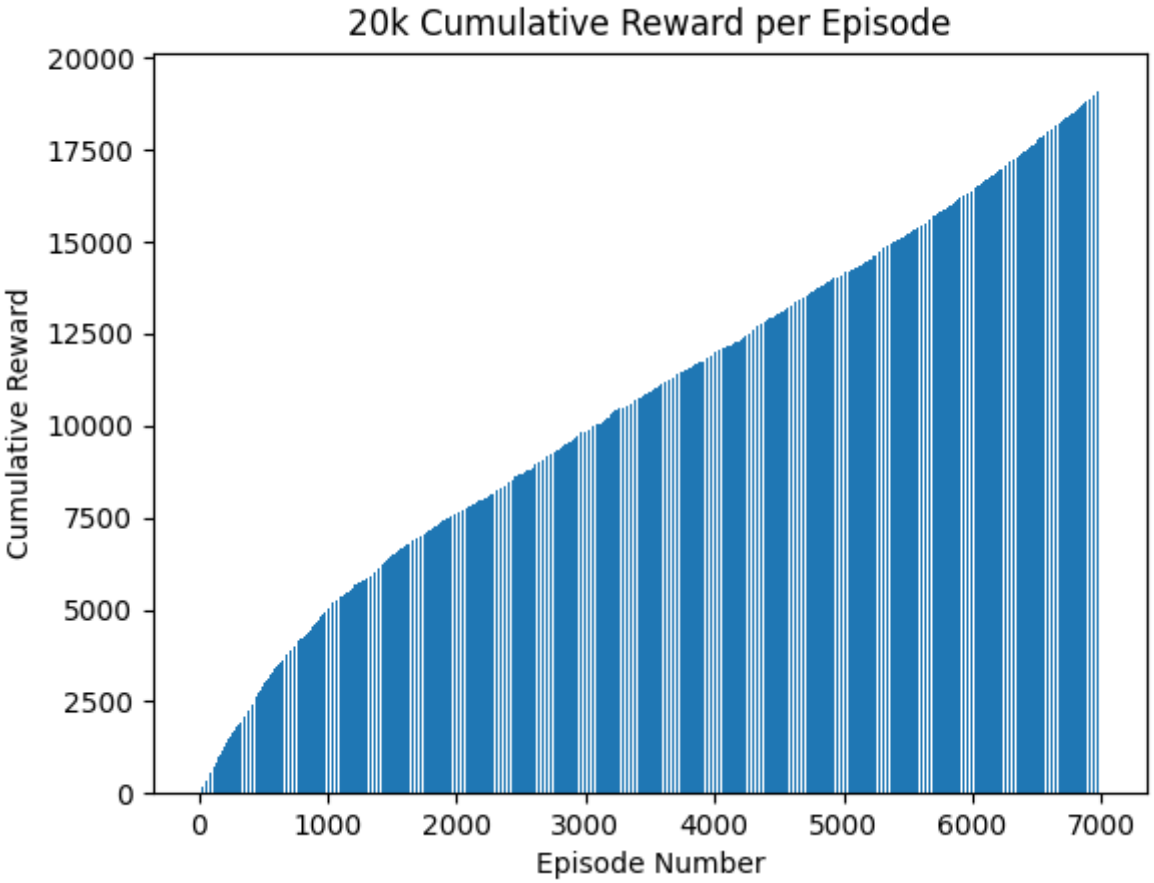

Figure 9

Figure 4.6 20k Cumulative Reward

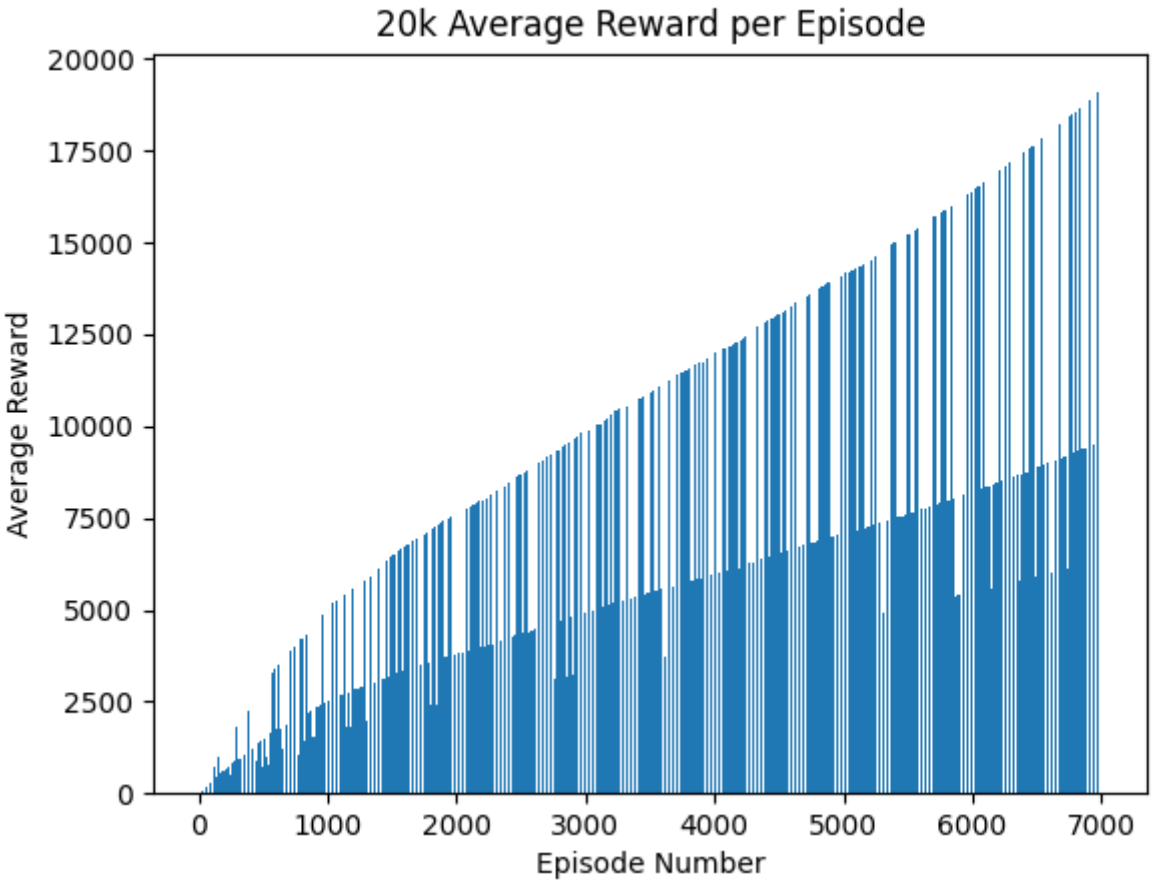

Figure 10

Figure 4.7 20k Average Reward

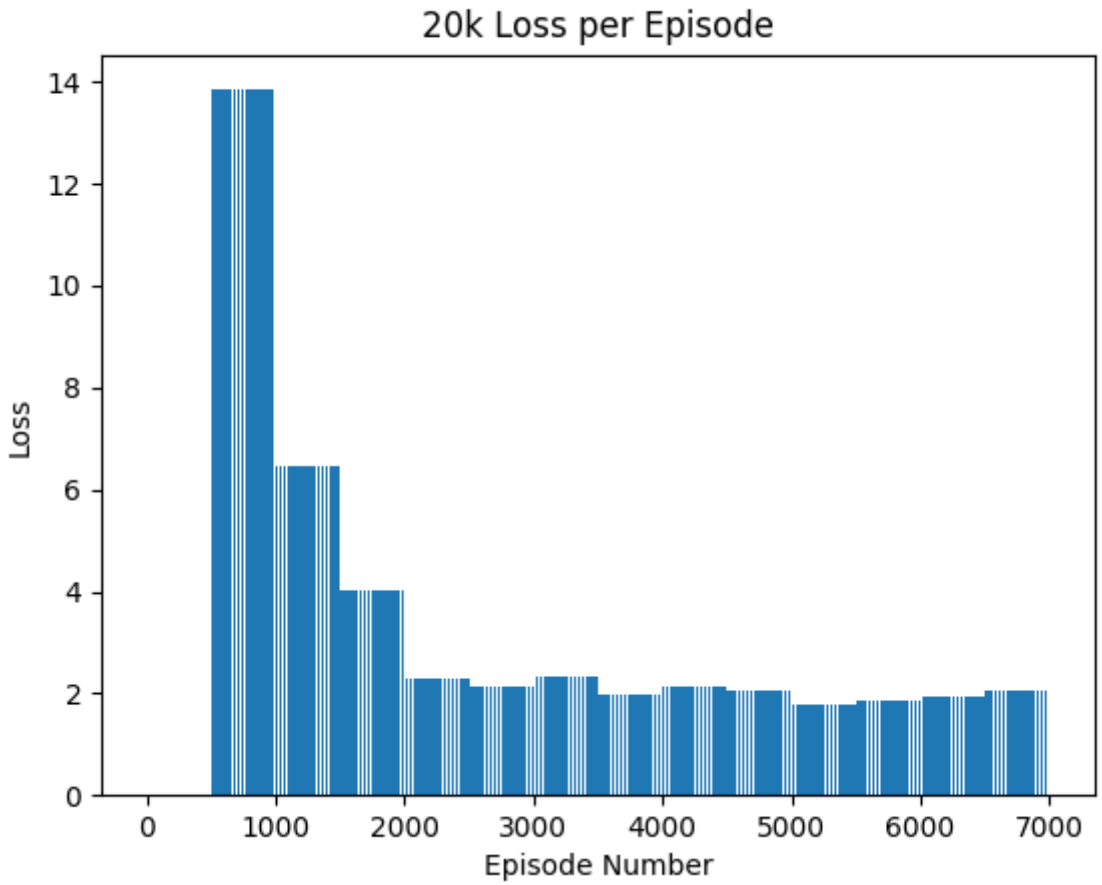

Figure 11

Figure 4.8 20k Loss

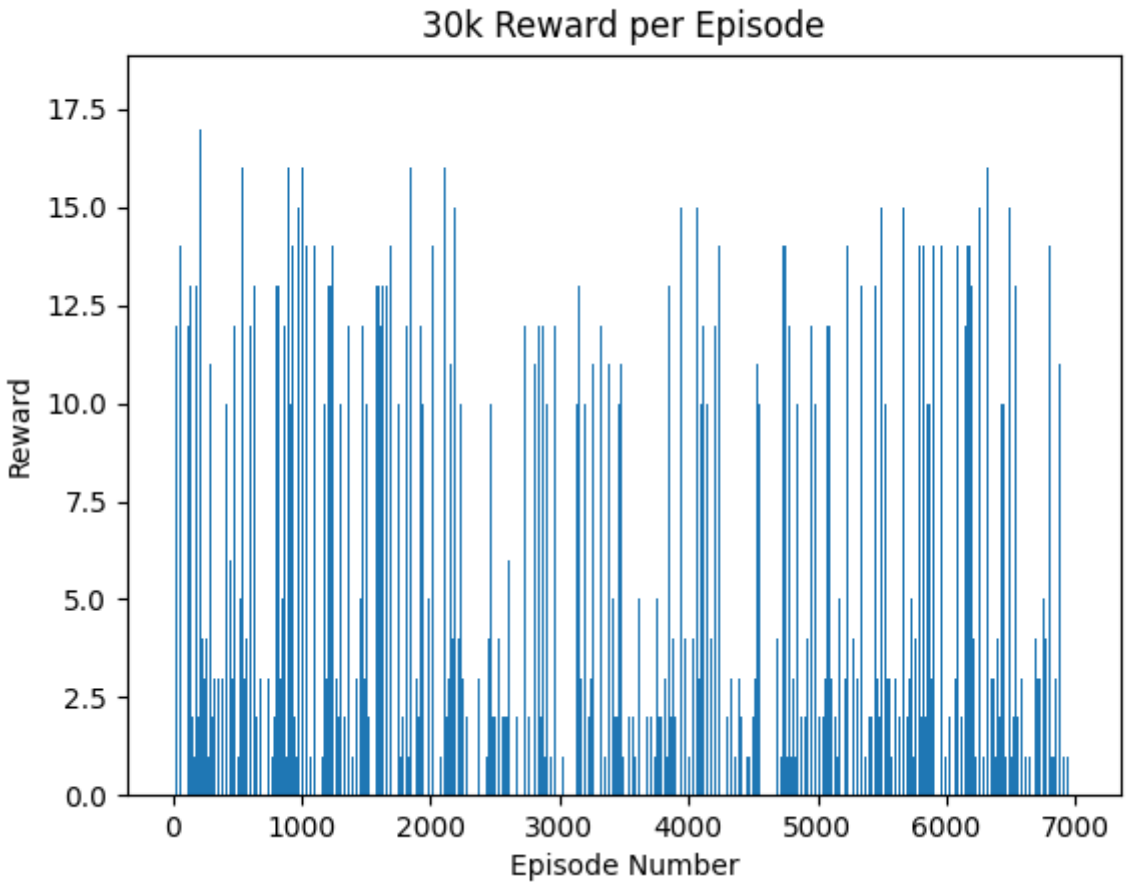

Figure 12

Figure 4.9 30k Reward Values Fluctuations during DQN Algorithm Training

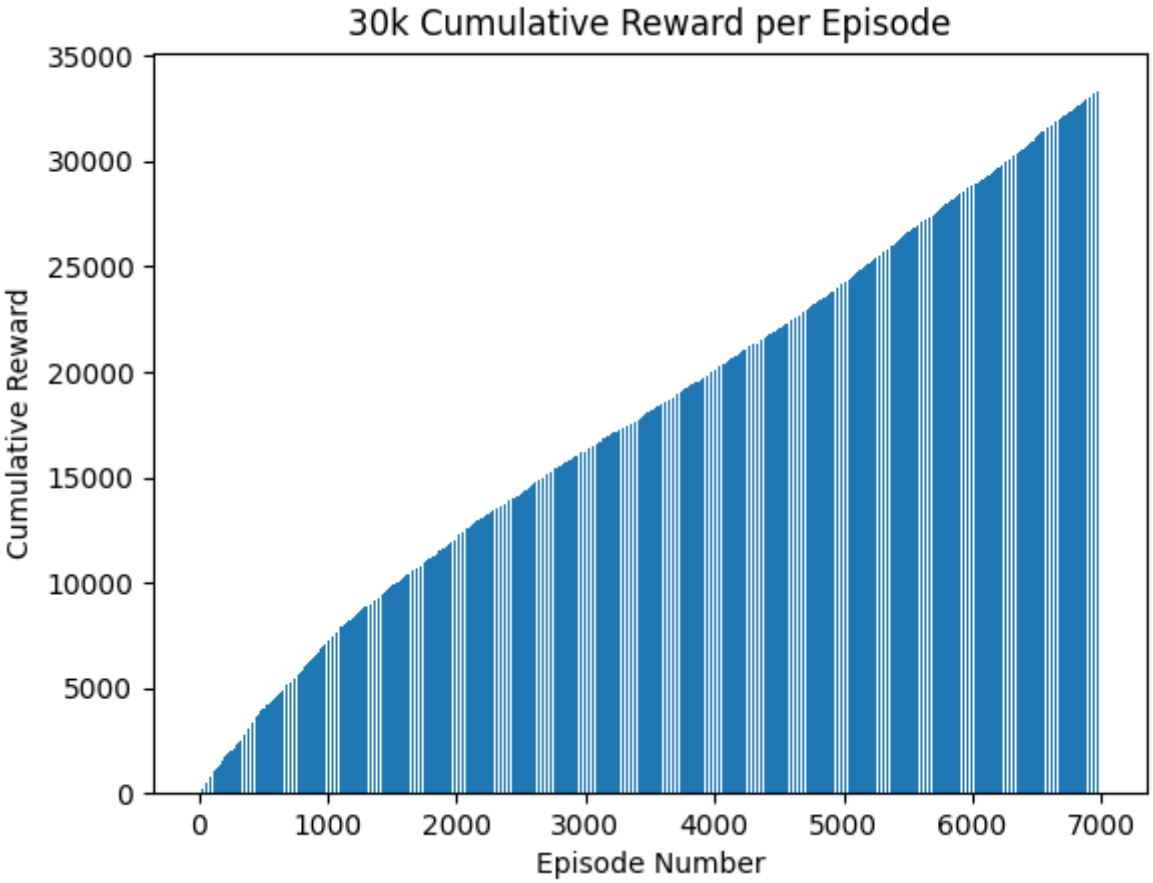

Figure 13

Figure 4.10 30k Cumulative Reward

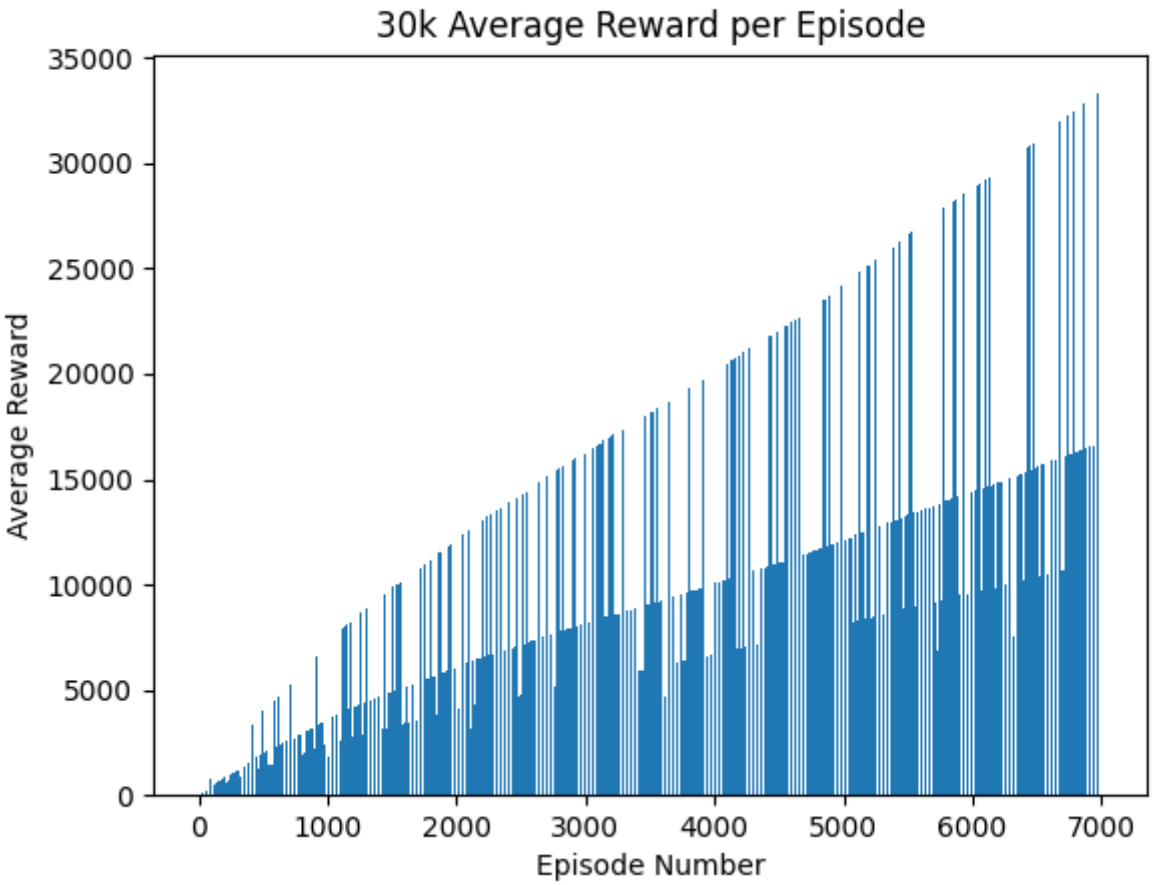

Figure 14

Figure 4.11 30k Average Reward

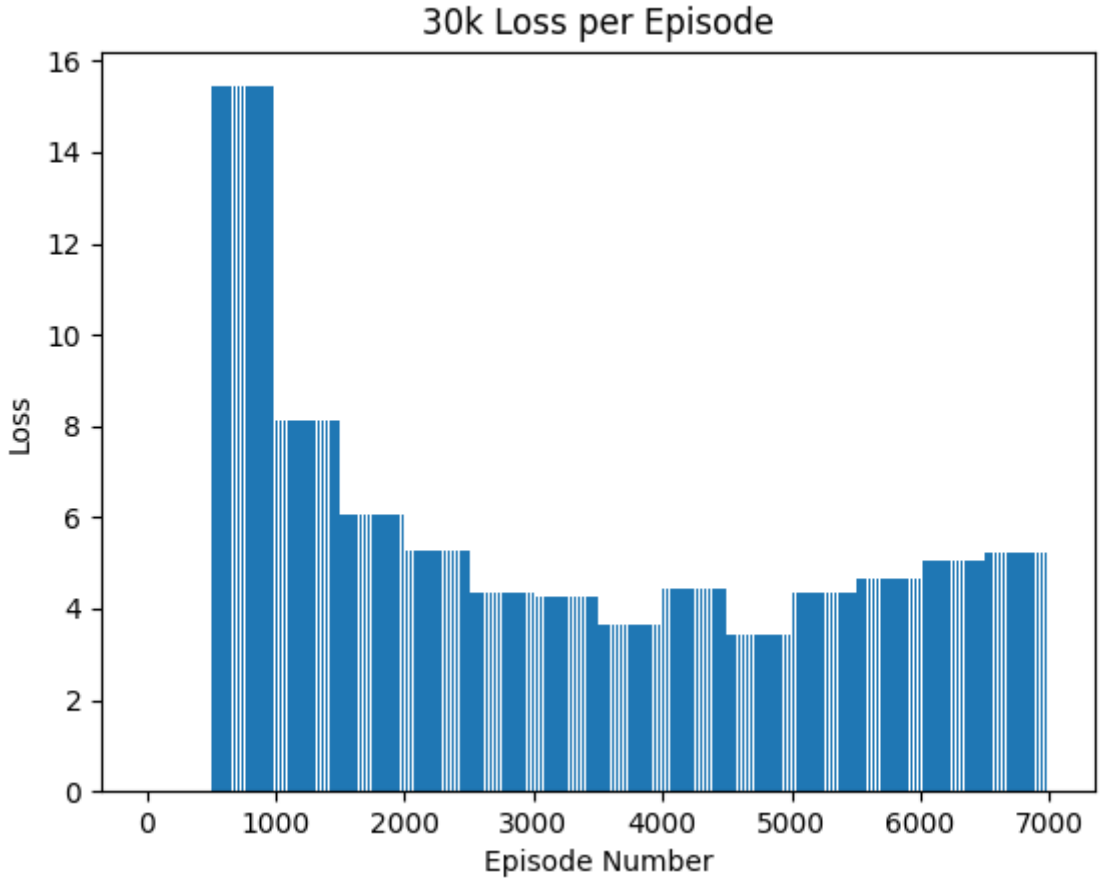

Figure 15

Figure 4.12 30k Loss

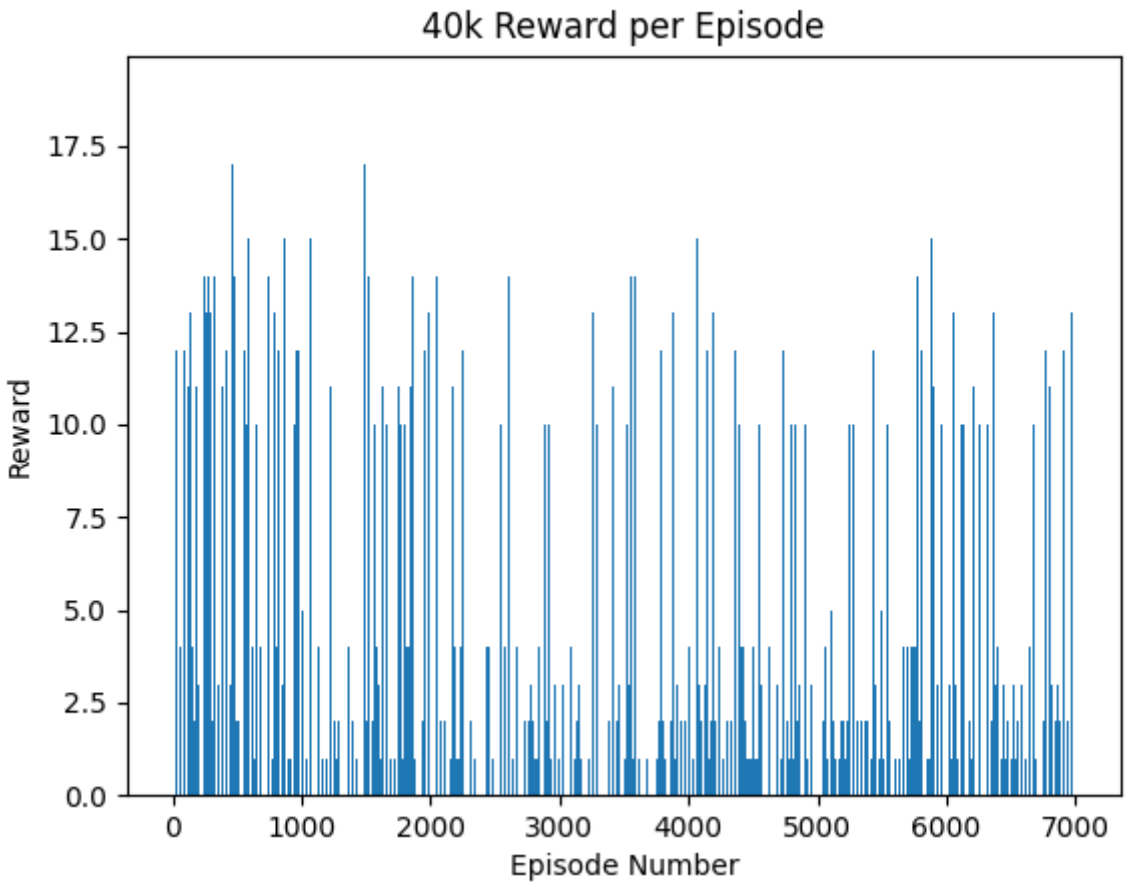

Figure 16

Figure 4.13 40k Reward Values Fluctuations during DQN Algorithm Training

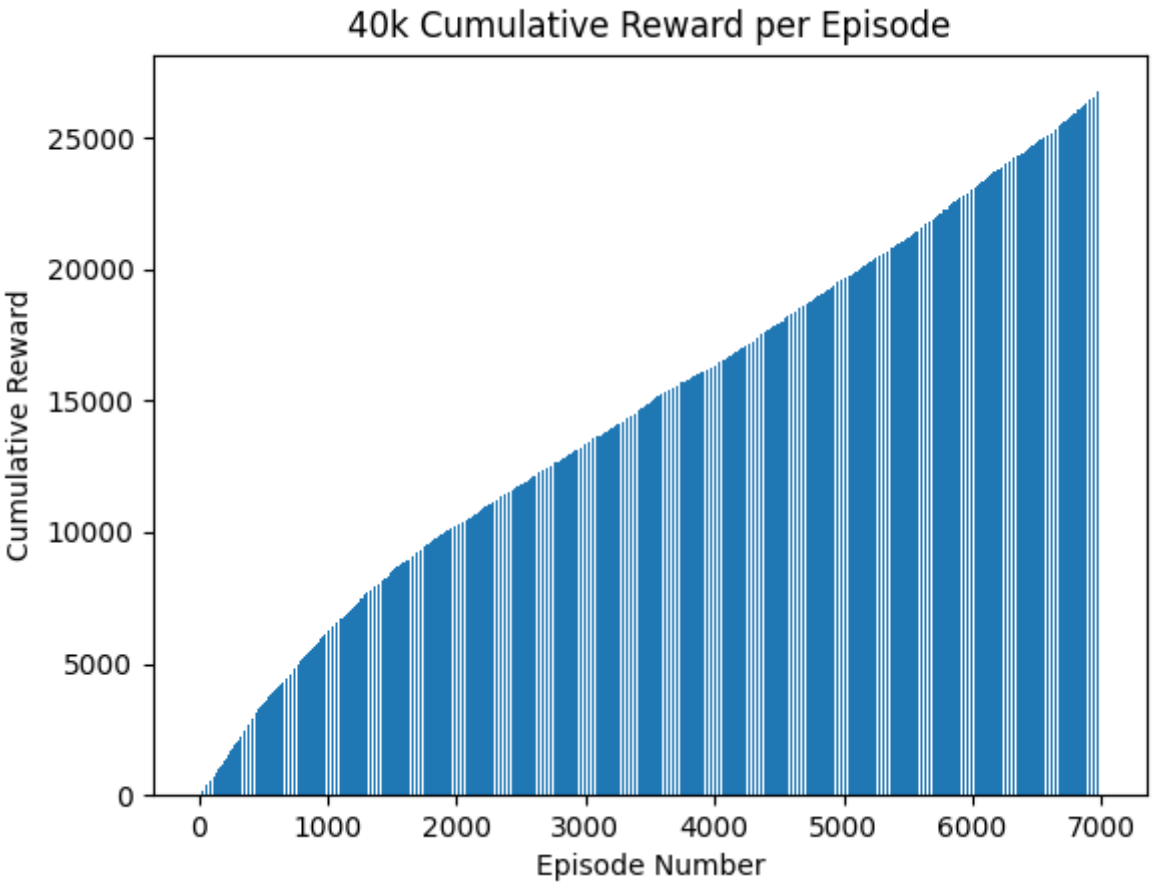

Figure 17

Figure 4.14 40k Cumulative Reward

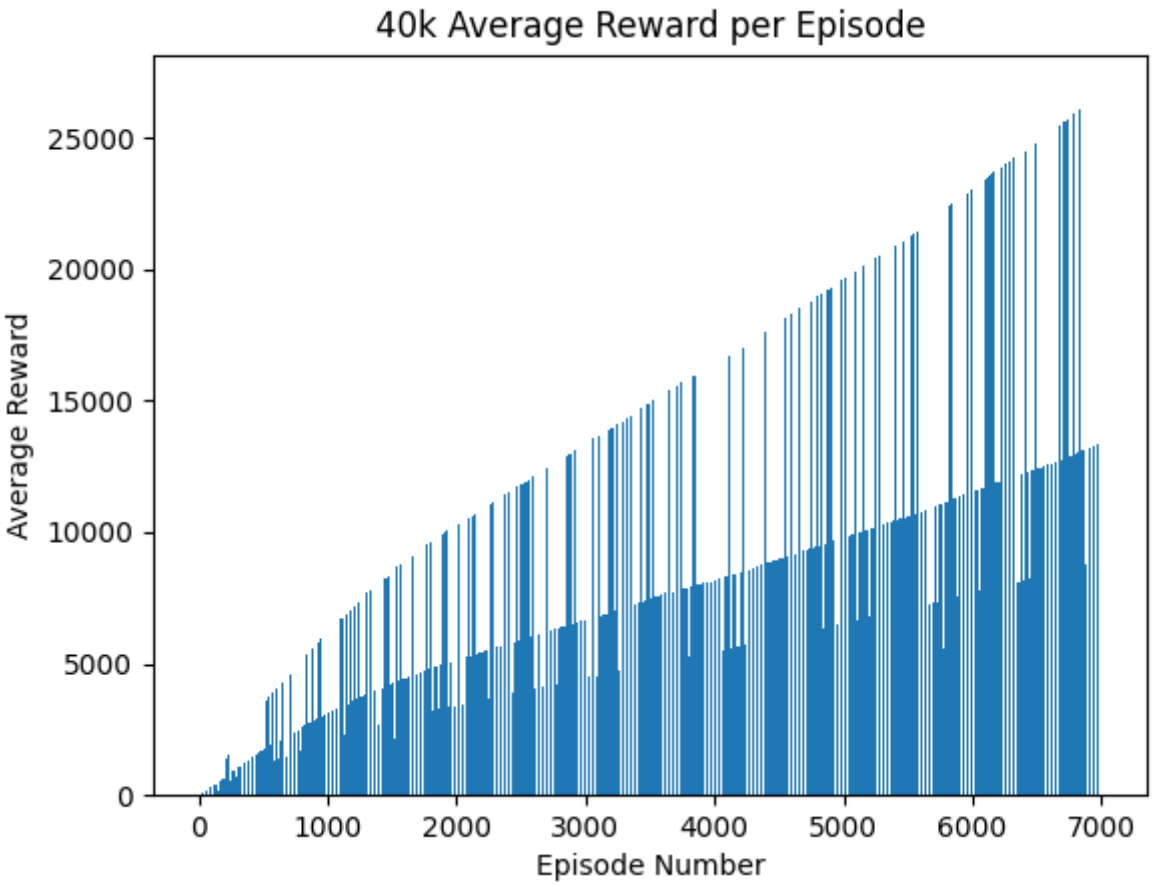

Figure 18

Figure 4.15 40k Average Reward

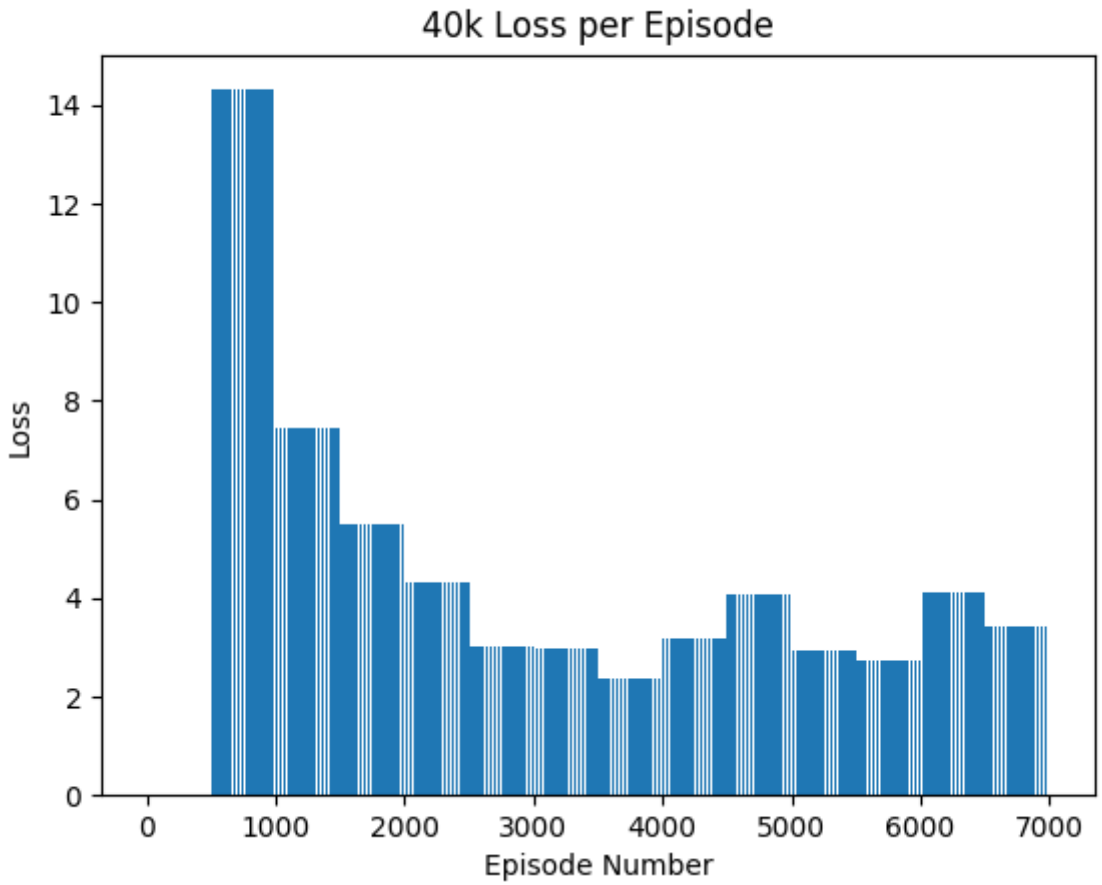

Figure 19

Figure 4.16 40k Loss

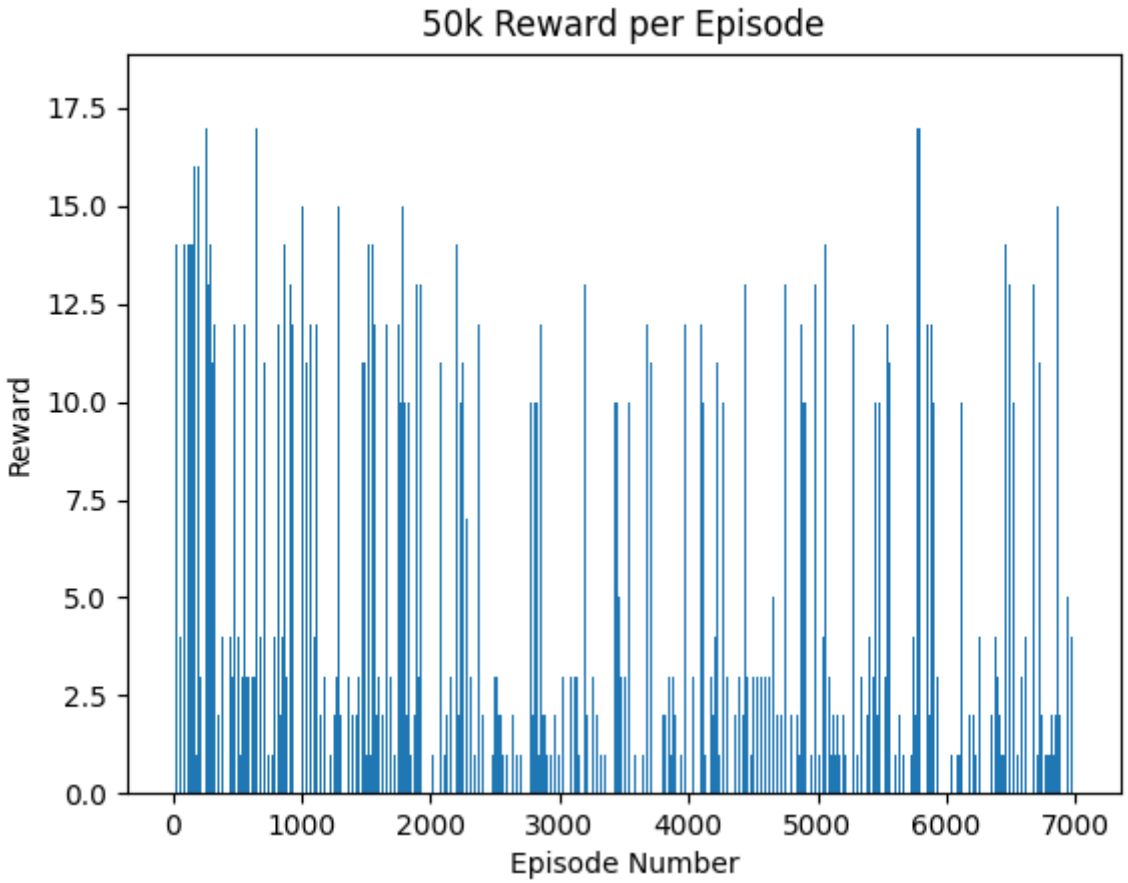

Figure 20

Figure 4.17 50k Reward Values Fluctuations during DQN Algorithm Training

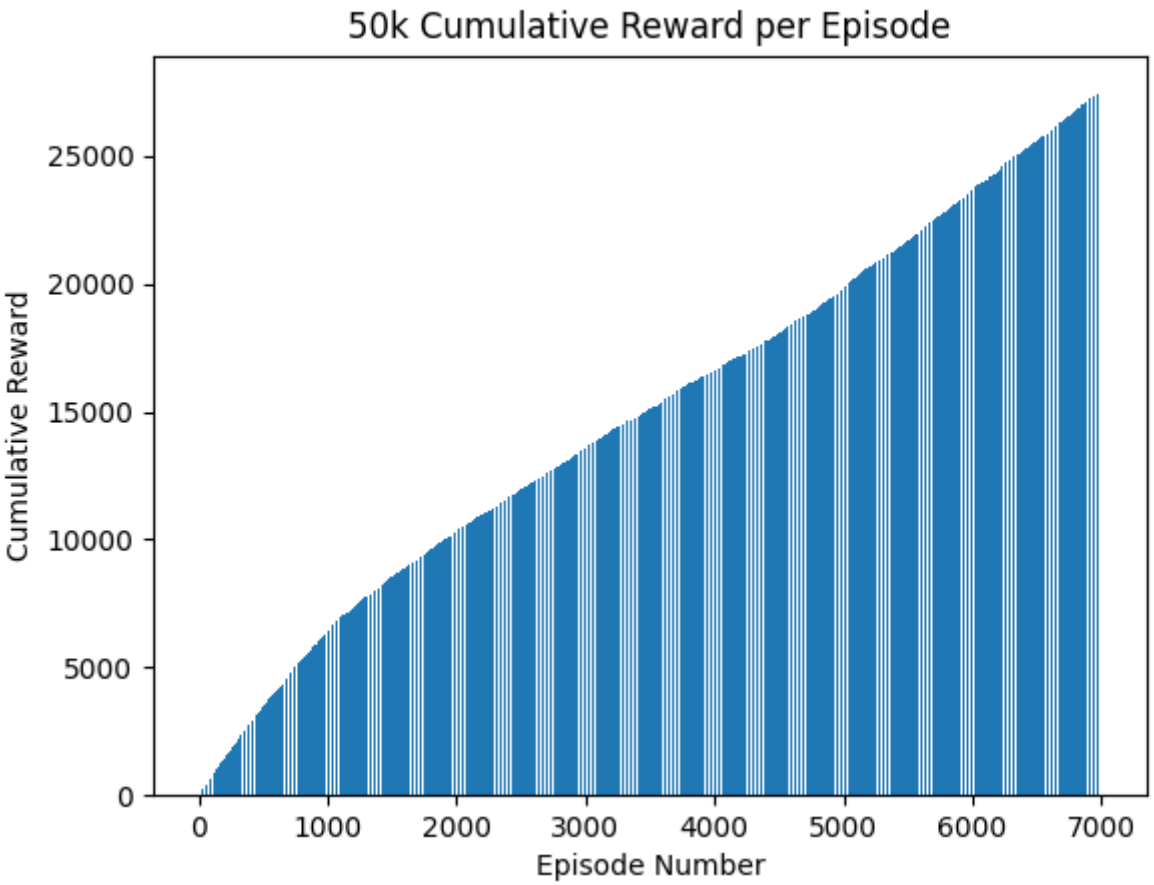

Figure 21

Figure 4.18 50k Cumulative Reward

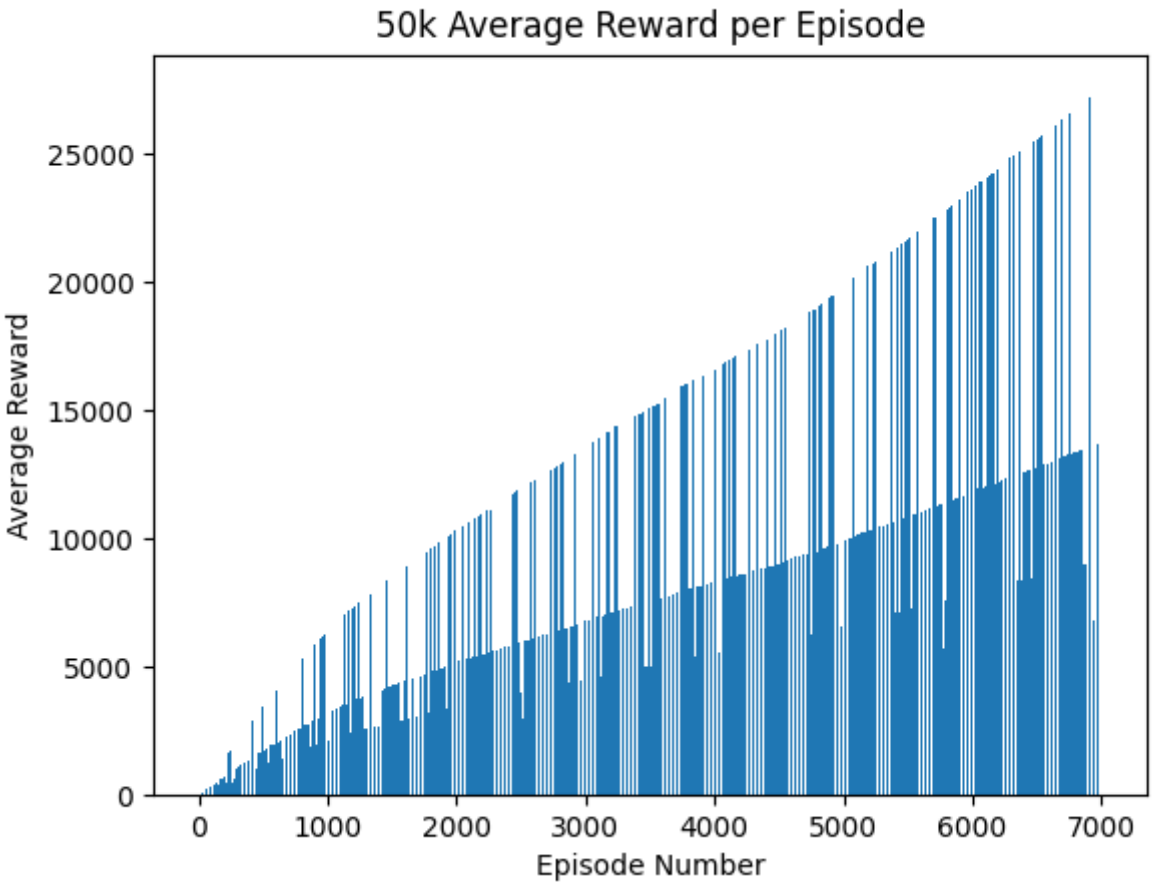

Figure 22

Figure 4.19 50k Average Reward

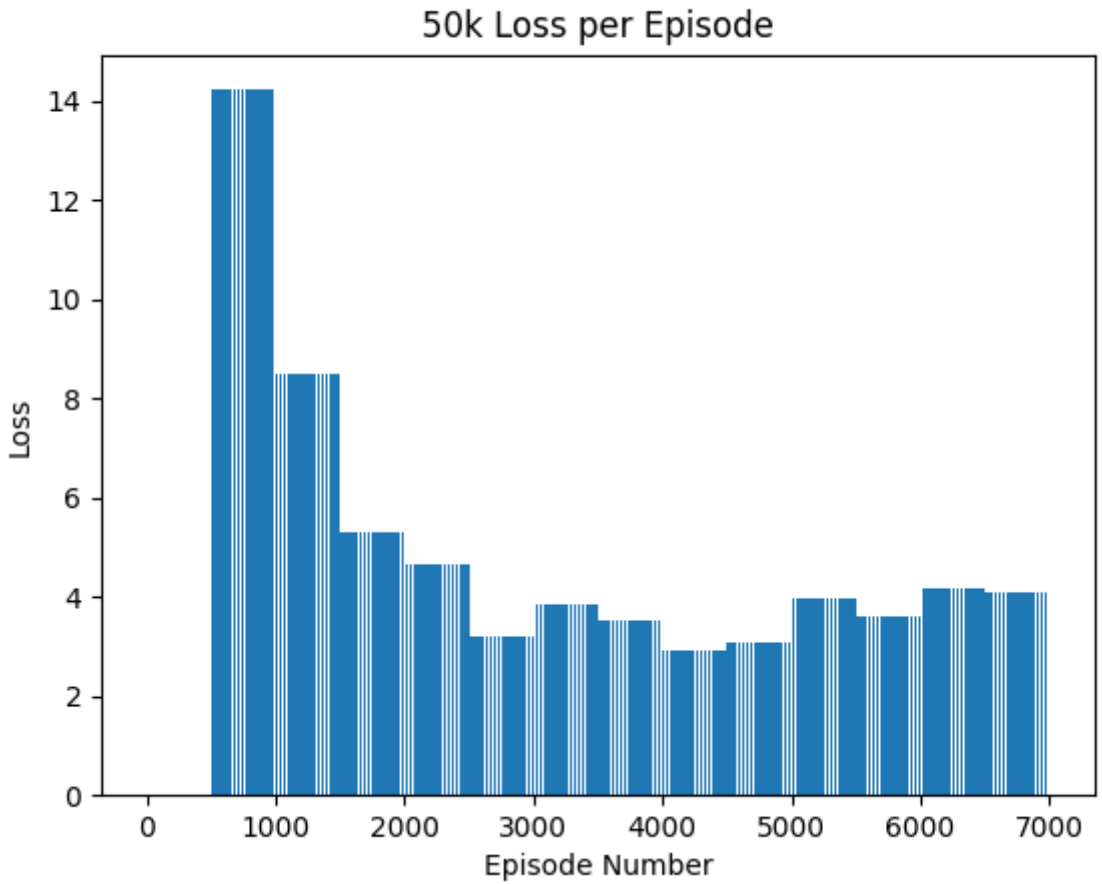

Figure 23

Figure 4.20 50k Loss