Value-Based Reinforcement Learning for Sequence-to-Sequence Models

Fabian Retkowski
fabian.retkowski@kit.edu
Karlsruhe Institute of Technology (KIT)

Alex Waibel
waibel@kit.edu
Karlsruhe Institute of Technology (KIT)

ABSTRACT

This paper demonstrates the theoretical possibility of applying advanced value-based reinforcement learning methods on sequence-to-sequence models for the first time. This approach avoids major issues that have emerged with supervised sequence-to-sequence models such as loss-evaluation mismatch, exposure bias and search error. At the same time, when compared to policy gradient methods, it does not rely on well-trained fully supervised models and is not restricted to fine-tuning. Specifically, a sequence-to-sequence model is introduced, which is trained in a Rainbow-like setup. While such a model is practically still limited by its scalability, the work contributes towards a more generally applicable approach to reinforcement learning in natural language processing which is beyond the scope of fine-tuning. For this, the paper provides a theoretical and practical framework, a first baseline, and valuable insights by studying ablated models and different approaches for utilizing demonstration data.

KEYWORDS

Reinforcement Learning, Natural Language Processing, Sequence-to-Sequence

1 INTRODUCTION

Seq2seq models offer great promise for sequence generation problems such as machine translation, text summarization, or dialogue generation. Nevertheless, fully supervised trained sequence-to-sequence (seq2seq) models have several methodological weaknesses which have not been completely solved. First, such models suffer from *exposure bias* as they are usually trained with teacher forcing [1, 2]. In this method, the model is conditioned on ground-truth data as input instead of its own outputs. Secondly, word predictions by these models do not consider the whole sequence, which introduces *search error.* Thirdly and most importantly, the maximum likelihood estimation (MLE) objective is often used in approximating the probability distribution $P(y|x)$, such that the likelihood of outputs given input is maximized. In many problems such as machine translation utilizing this distribution may be sufficient. However, for other problems, it differs substantially from the test objectives and the real-world goal.

It has been shown by Li et al. [3] that when applied to the task of dialogue generation, the aforementioned models tend to generate highly generic, repetitive, and short-sighted responses, with "I don't know" among them. This outcome can be ascribed to the high frequency of such phrases in dialogue corpora, which are then favoured by the MLE objective.

Reinorcement learning (RL) addresses all of these issues. It allows using any function as a reward, which may include non-differentiable test metrics, human feedback, or other functions that are closer to the real-world goal. Furthermore, RL relies on its own outputs instead of ground-truth data, and it naturally incorporates future rewards, thus avoiding exposure bias and search error.

However, RL is considered sample inefficient, especially in the case of reward sparsity and large action spaces, and this is particularly true for most NLP tasks, which require dealing with huge vocabulary sizes. Thus, most research in this area has focused on fine-tuning supervised models. It has, therefore, also been limited to methods that output softmax probabilities, which makes them easy to pretrain with supervised approaches. This category includes PG methods and actor-critic setups. For instance, REINFORCE is still widely used in this area, although this algorithm is known to have severe issues, such as a time-consuming training and high variance. In contrast, value-based learning methods such as Q-learning have received little attention, since they need to predict future rewards for every single action and cannot be easily pretrained. However, value-based methods have made significant progress in recent years. Advanced methods such as Rainbow and FQFs are state-of-the-art approaches in many fields (e.g., game-playing) and might overcome some caveats of Q-learning making it a reasonable choice for the area of NLP. Thus, this work investigates whether seq2seq models can be trained with Rainbow [4], which is a Q-learning-based approach that seeks to combine several improvements made to Deep Q-Networks (DQNs) in recent years, including prioritized experience replay (PER) and multi-step learning.

2 RELATED WORK

Sequence generation using RL. Motivated by the issues of fully supervised sequence-to-sequence models, Ranzato et al. (2015) [1] applied RL on different sequence generation problems to align training and test measurements. The researchers utilized supervised training to initialize the policy, and they then introduced the MIXER algorithm which provides an annealing schedule between supervised training (i.e., cross-entropy loss and teacher forcing) and reinforcement learning, using REINFORCE [5] and the model's own predictions.

Due to the disadvantages of REINFORCE, which include the time-consuming training and its high variance, Bahdanau et al. (2017) [6] took this idea one step further by employing an actor-critic setup to train encoder-decoder models. To speed up the training process and deal with the ample action space, the authors used several techniques such as penalizing the variance of the value predictions and reward shaping to provide rewards for the whole sequence.

Rennie et al. [7] (2017) introduced another variant, a Self-Critic (SC) baseline for policy gradient (PG) methods employed in sequence generation. The algorithm avoids estimating a normalization (cf. REINFORCE) or training a value function (cf. actor-critics) for the baseline by utilizing the inference outputs.

However, all above mentioned approaches rely on well-trained supervised models, and the application of RL is limited to fine-tune such models. Consequently, this line of research focuses on PG methods as the provided outputs are alike (i.e., token probabilities based on softmax activations). Practical and more advanced applications of the presented approaches can be found in [3], [8], and [9] which embed them in a more complex multi-stage training procedure while also using multi-component reward functions.

Recurrence and memories in RL. Another related research direction seeks to equip DQN agents with a memory. The first notable work in this area was conducted by Hausknecht et al. in 2017, with their proposed Deep Recurrent Q-Network (DRQN) [10]. The authors argue that most real-world applications fail to meet the Markov property; that is, their true states are only partially observable. Consequently, they propose replacing the first fully-connected layer with an LSTM layer. With this goal in mind, Hausknecht et al. consider two possibilities: sequential updates (replaying whole episodes while violating DQN's random sampling policy) and random updates (with zeroing out hidden states). The paper concludes that both updates work similarly well.

A more recent approach is Recurrent Replay Distributed DQN (R2D2), developed by Kapturowski et al. (2019) [11]. The authors store and replay fixed-length sequences ($m = 80$) using a mix of mean and max for prioritization of the samples: $p = \eta \max_i \delta_i + (1-\eta)\overline{\delta}$ with $\eta = 0.9$. They hypothesize two strategies for the hidden state: storing and replaying such states or applying a burn-in period, which involves using a part of the replay sequence to produce a start state.

3 SEQUENCE-TO-SEQUENCE MODELS

The classic seq2seq architecture, also called encoder-decoder architecture was first proposed by Sutskever et al. (2014) [12]. The motivation behind this architecture is to map an input sequence (source) to an output sequence (target), both of which can be of arbitrary lengths. The architecture is composed of an encoder and a decoder.

- The encoder RNN compresses the input sequence $x = (x_1, x_2, \ldots, x_{n_x})$ to a fixed-length vector C (thought vector or context vector), which is the final hidden state vector h_{n_x} of the RNN.
- The decoder's hidden state is initialized with the fixed-length vector C. The decoder RNN then generates the output sequence $y = (y_1, y_2, \ldots, y_{n_y})$.

RNN encoders and decoders are typically implemented as LSTM [13] or GRU [14], either unidirectional or bidirectional. The decoder usually adds another linear layer with softmax activation (the so-called *generator*) to output a probability distribution over the vocabulary.

The outputs of the decoder network are fed into the next sequential unit as input. Consequently, mistakes at the beginning of the sequence can lead to increasing erroneousness, which ultimately results in slow convergence. Thus, the most common algorithm for training the model is *teacher forcing*, which feeds the actual correct sequence (the targets) into the model. This algorithm allows parallelization as it removes the necessity to wait for the sequential outputs to be used as inputs. The teacher forcing algorithm can also be derived from the (conditional) maximum likelihood objective, which is the CE loss:

$$\begin{aligned} \mathcal{L}_{CE} &= -\log p(y_1, \ldots, y_{n_y}) = -\log \prod_{t=1}^{n_y} p(y_t | y_1, \ldots, y_{t-1}) \\ &= -\sum_{t=1}^{n_y} \log p(y_t | y_1, \ldots, y_{t-1}) \end{aligned} \tag{1}$$

During inference, the next output word is chosen by a greedy left-to-right process, $y_{t+1} = \operatorname{argmax}_y p(y|y_t, h_t)$, without considering the complete sequence. This approach, however, might not produce the most likely sequence according to the abovementioned objective, an outcome known as *search error.* One way to reduce the search error is beam search, tracking k word candidates. In addition, this setup suffers from *exposure bias* because of the distribution mismatch of ground-truth data and the model's predictions.

4 METHODOLOGY

This research aims to apply the Rainbow DQN setup to the seq2seq architecture. Therefore, this section addresses the question of how the classic DQN approach can be transferred to this architecture. With the fulfilment of this requirement, most of the DQN extensions that have been used in Rainbow are also straightforwardly transferable. Specifically, the following extensions are included:

- Double Q-learning [15].
- Prioritized experience replay [16], which is the only extension that is not transferable without significant methodological changes.
- Dueling networks [17].
- Multi-step learning [18].
- Distributional RL; for this paper, the more recent QR-DQN [19] is chosen over categorical DQNs [20] because it is easier to implement and has also been shown to yield better results, though categorical DQNs have been used for Rainbow.
- Noisy nets [21].

4.1 Reinforcement Learning Setting

In this section, the RL environment is described as it varies from one task to another. In general, the setting is similar to those in other sequence prediction tasks such as [1], [3], and [6], although the perspective on the state differs substantially as these works use policy gradient approaches.

Action space. In the context of this work, the action space $\mathcal{A}$ is the vocabulary space. At each time step t, the decoder of the seq2seq model chooses the next action A_t, which is a token in a sequence.

State space. The state S_t at a specific time step t includes all the input data that is required to produce the next action A_t. Since the decoder generates its output depending on the previous hidden state h_{t-1} and the previously chosen action y_{t-1}, $[h_{t-1}, y_{t-1}]$ may be used as the state. Alternatively, the input sentence and all previous actions $[x, y_{1:t-1}]$ can be viewed as the state, as it is possible to reproduce the hidden states with this information.

Reward. The reward function r can be any user-defined function. For example, advanced dialogue generation models like [3] utilize rather complex reward functions such as combinations of information flow, semantic coherence, and ease of answering. A unique characteristic in NLP settings compared to other RL tasks is that the reward is always and only collected at the end of the sequence. However, to keep the work as simple, comparable and interpretable as possible, and to focus on the feasibility of transferring Q-learning to seq2seq models, BLEU [22] and ROUGE-W [23] are selected as exemplary reward functions. In addition, with such rewards, it will be possible to provide a strong baseline for the model. The cross-entropy (CE) objective is known to approximate these metrics quite well. Both metrics evaluate a generated sentence against a reference sentence. Thus, a dataset with sources and targets is required. More details in regard to BLEU and ROUGE can be found in the appendix, see Section A.

4.2 Experience Replay for Sequence-to-Sequence Models

Typically, Q-learning approaches with deep neural nets (i.e., DQNs) store transitions experienced by the agent in a buffer called experience replay. These transitions are reiterated during the training process. A transition is defined by its state S_t and action A_t at time step t, the next state S_{t+1}, and the reward R_{t+1} received by the agent: $(S_t, A_t, S_{t+1}, R_{t+1})$. For seq2seq models, however, this approach has to be adjusted. As mentioned, the state is defined by the previous hidden state of the decoder and the previous action $[h_{t-1}, y_{t-1}]$. However, given that the hidden state representation is not static, but learned during the training process, it is not suitable to be replayed in later phases of training. Here, the alternative state representation $[x, y_{1:t-1}]$ can provide a solution. Rather than storing single transitions, it allows for storing the entire input and output sequence (x, y, R_T) to represent the states and actions of the whole episode e. The entry is completed by a scalar reward R_T, as only the final transition issues a reward for the full sequence.

Prioritized Experience Replay. This decision has some implications, especially for PER, one of the extensions, which has been combined with others in [4]. In the original paper [16] by Schaul et al., the absolute TD error δ is utilized as the criterion of importance (i.e., priority p) for the transitions in the buffer. However, here it is necessary to deal with whole episodes, which consist of many transitions. Consequently, there is a need to aggregate the TD errors of the steps in episode e. For this approach, there are several options, with summing and averaging being the obvious ones:

- Summing the errors: $p_e = \sum_i^T \delta_i$
- Averaging the errors: $p_e = \frac{1}{T} \sum_i^T \delta_i$

It may be expected that the summation of errors leads to an advantage of longer sequences at the expense of shorter ones; this outcome could be disadvantageous for the overall success. In fact, early experiments have indicated that averaging is superior.

4.3 Teacher Forcing

As discussed, entire episodes must be stored in the experience replay buffer to combine it with sequence-to-sequence models. However, as the name suggests, it is necessary to replay the episodes. This is where a technique that is widely used in supervised learning for seq2seq models comes into play: teacher forcing. Instead of feeding the decoder's output to the input of the next sequential unit (as in the inference stage), the ground-truth sequence is fed into the network. The same idea can be applied to replay episodes, but in place of the ground-truth sequence, the stored output sequence is inputted into the recurrent units of the decoder. The essential difference is that the output sequence does not necessarily have to be one of the "good examples". The examples in the experience replay buffer are usually collected by the model itself, substantially reducing the exposure bias caused by the distributional mismatch of decoder inputs in the training and inference stages. Here, the model's own predictions are replayed in the training stage, syncing the input distributions. Consequently, while the algorithm applied is the same for supervised learning and the RL approach taken in this study, its aim and motivation is entirely different. Moreover, teacher forcing has computational benefits since it allows parallelization.

4.4 Temporal Difference Error

The concrete algorithm for calculating the temporal difference error is depicted in Algorithm 1 and closely resembles the work of Mnih et al. [24]. The main difference is that the algorithm handles a batch of episodes (i.e., a sequence of transitions) instead of a batch of transitions. Since these sequences can be of different lengths m, padding and masking is required, as well as a normalization with m. To allow for parallelization, shifting and zero-padding is required.

Algorithm 1 TD error for a basic sequence-to-sequence DQN

```
Input: Source $x$                                              $x \in \mathbb{N}^{N\times B}$
       (Padded) Output $y$                                     $y \in \mathbb{N}^{M\times B}$
       Reward $r_T$                                            $r_T \in \mathbb{R}^{B}$
       Output Lengths $m$                                      $m \in \mathbb{N}^{B}$
 1: $o, \bar{o} = \text{seq2seq}_\theta(x, y), \text{seq2seq}_{\bar{\theta}}(x, y)$      $o, \bar{o} \in \mathbb{R}^{M\times B\times H}$
 2: $q, \bar{q} = q_\theta(o), q_{\bar{\theta}}(\bar{o})$                       $q, \bar{q} \in \mathbb{R}^{M\times B\times |\mathcal{A}|}$
 3: $\hat{q}_{2:M,i} = \max_{a'} \bar{q}_{2:M,i}(a'), \forall i$                 $\hat{q} \in \mathbb{R}^{M-1\times B}$
 4: $q_p = \text{mask}(q(y))$                                  $q_p, \hat{q}_p \in \mathbb{R}^{M\times B}$
 5: $\hat{q}_p = \text{concat}\,(\text{mask}(\hat{q}), 0_{1\times B})$
 6: $r = \text{concat}(0_{M-1\times B}, r_T)$                          $r \in \mathbb{R}^{M\times B}$
Output: $\frac{1}{B}\sum_{j=1}^{B} \frac{\sum_{i=1}^{M}(r_{i,j}+\gamma\hat{q}_{p_{i,j}}-q_{p_{i,j}})^2}{m_j}$
```

The presented algorithm does not include the six DQN extensions. However, with the algorithm and methodology presented it is straightforward to add them. Some more details and aspects, especially regarding multi-step learning, can be found in the appendix, see Section B.

4.5 Utilization of Demonstration Data

Although it is highly flexible in defining its goals and rewards, RL also has some downsides: it is usually exceedingly sample inefficient and converges much slower than supervised learning. Furthermore, data collection is time-consuming. This is why, as part of this work, a methods are explored with which available information can be utilized to accelerate convergence.

Preloading Replay Buffer. DQNs learn from transitions being collected by the agent and stored in the experience replay buffer. In the case of this work, however, it is assumed human demonstration data is already at hand, as there are many corpora of natural language available to use. The simplest way to leverage such data is to preload it into the replay buffer instead of filling the buffer with random experiences in the beginning. For practical reasons, the experience replay buffer typically has a size limit. This is why older transitions get replaced by more recent experiences. However, to prevent the displacement of exemplary data, such data is excluded from the "first in, first out" replacement policy and instead is permanently stored in the buffer.

Tranfer Learning. This approach is inspired by classic transfer learning. As Yosinski et al. [25] have shown, layers in a deep neural network for image classification, that were trained on a specific task can be transferred to others to varying degrees. The new model is then able to train faster. In particular, early layers in the network are rather general, agnostic regarding the specifics of the input, and therefore easily transferable. Similar work was conducted by [26] for RNNs and the area of NLP. However, the authors suggest that, in this domain, a semantic relatedness between the tasks are more significant than in computer vision. In the context of this work, this method entails pretraining a typical seq2seq network using supervised learning. The parameters of this model, or specifically, the parameters of the encoder, decoder, and the embedding layers while discarding those of the generator, are utilized to initialize the Q-learning model, which has its own randomly initialized generator. Thus, the recurrence and embeddings may not have to be learned from scratch. This approach is similar in conception to those in [1], [6], and [3]. However, for PG methods, it is not necessary to replace the generator because both generators produce probabilities for the defined set of tokens. On the contrary, in DQNs, the output layer utilizes a linear activation function. Additionally, the number of neurons in the output layer differ when employing distributional RL.

Multitask Learning. Transfer learning works optimally when the training data and training objective of both tasks are similar. In this study, however, the objectives differ substantially, as [3] suggests. On the one hand, there is the MLE criterion; on the other hand, Q-values, the estimated future rewards, are to be predicted. Thus, instead of using transfer learning, it would be possible to treat these objectives as two different tasks, but to employ a shared "feature extractor", which, in this case, is the encoder RNN, the decoder RNN and the embedding layers. The general idea is known as multitask learning, and it has been successfully applied to a broad range of applications, including NLP [27] and computer vision [28]. Originally, multitask learning was described by [29]: it is usually implemented by sharing hidden layers between several tasks while having task-specific output layers. These tasks are learned jointly by alternating the optimization steps for each. The different tasks benefit from each other as they introduce regularization and reduce the hypothesis space.

Vocabulary Size	Dataset Size	Word Minimum Frequency
111	1, 311	900
201	3, 484	680
401	9, 230	130
806	18, 523	50

Table 1: Dataset vocabulary sizes

5 EXPERIMENTS

For subsequent experiments, a basic single-turn dialogue generation task is assummed, based on the Cornell Movie Dialogue dataset [30] and evaluated on both, BLEU and ROUGE-W. Four models are investigated:

- As comparative model, a supervised trained seq2seq model conditioned on the maximum likelihood objective is employed. Such a model can be considered a strong baseline because the CE loss is known to approximate BLEU and ROUGE quite well.
- A seq2seq network trained purely with RL, based on the Rainbow method introduced in including the methodological modifications needed presented. In this setup, the replay buffer is preloaded with demonstration data.
- A transfer learning model is not fully evaluated because early experiments have shown it to converge to suboptimal solutions.
- A multitask network, which jointly trains the supervised learning and RL models described above.

Multiple experiments have been conducted to assess the scalability of the presented model. Generally, four settings are considered as seen in Table 1. In each case, the action space is approximately doubled, resulting in vocabulary sizes of 111, 201, 401 and 806. The dataset size exhibits disproportionate growth ranging from 1,311 to 18,523 examples.

The implementation of the work conducted is available online.[1]

6 RESULTS

6.1 Scalability

The results, which are displayed in Table 2, demonstrate that it is possible to train a seq2seq network with the methods of value-based RL. For limited problem sizes, these methods are able to match or even surpass ambitious baselines such as supervised trained models in their stronghold settings. While conducting the experiments, however, it became evident that the model is subject to scalability constraints. With the initial parameter setting of $N = 21$ for the number of quantiles, it was not possible to scale to an action space size of 401 without a drop in performance. Instead, the hyperparameter had to be reduced to 5, which seems to lift the upper limit of its scalability to the 806 setting.

6.2 Exemplary Outputs

The evaluation in this section refers to the 401 setting, as this is the setting for which both RL models are still able to match or

[1]https://github.com/ScientiaEtVeritas/rainbow-dialogues

Metric / Reward	Model		Vocabulary Size 111	201	401	806
BLEU	SL		0.71	0.70	0.77	**0.77**
	RL		**0.74**	**0.71**	**0.81**	0.71
	MTL	SL	0.73	**0.71**	**0.81**	0.67
		RL	0.73	**0.71**	0.75	0.42
ROUGE	SL		**0.6**	0.60	0.63	0.60
	RL		**0.6**	**0.63**	**0.65**	**0.61**
	MTL	SL	0.58	0.61	0.64	0.49
		RL	0.57	0.60	0.60	0.36

Table 2: Evaluation of presented models, namely supervised learning (SL), reinforcement learning (RL), multitask learning (MTL) with its respective SL and RL output layers, in different setups to assess the scalability

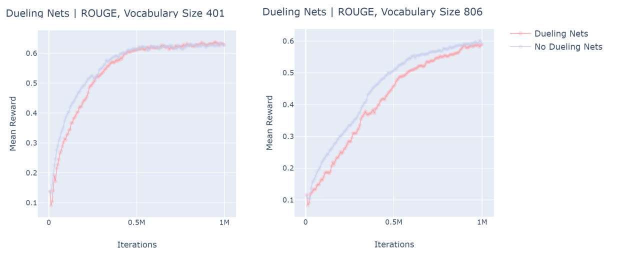
surpass the supervised trained model on the respective evaluation metric. Generally, the trained models produce largely similar results. The RL$^{\text{BLEU}}$ model outputs the same sequences as the supervised model for 80.92 per cent of unique sources in the corpus. A similar outcome holds for the RL$^{\text{ROUGE}}$ model with 80.80%. Even among themselves, they share outputs in a similar order of magnitude (i.e., 81.69%).

Nevertheless, in Table 3, ten random examples are presented for which the models returned different sequences. From studying these examples, possible reasons for the differences include, that the target is still incompletely or partly learned or, given the nature of the dataset, in which a source can have several targets, the models simply approach different targets. Interestingly, there are also a few examples that reveal the very distinct nature of learning. For instance, the RL$^{\text{ROUGE}}$ model outputs "*What Why?*" for "*What time is it?*" and scores a medium reward on the "*Why?*" and "*What?*" target with it. However, this output sequence ultimately yields a higher combined reward on ROUGE-W (1.51) than predicting either of the targets (1.396). This example highlights the importance of choosing an adequate reward function and the necessity that such a reward function encompasses "semantic coherence" as in [3].

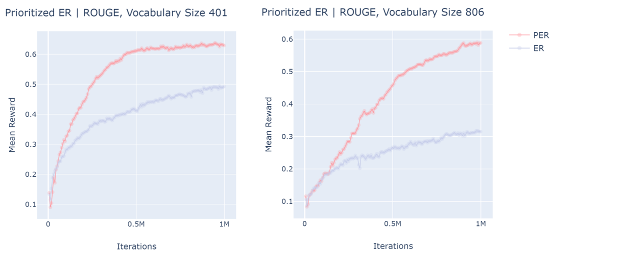
6.3 Ablation Study

The presented model, which is based on Rainbow, combines several DQN improvements. In order to reach a better understanding of how these extensions behave in this work's high-dimensional seq2seq setup, an ablation study was conducted. In this study, the influence on performance and convergence of some algorithmic ingredients was assessed by removing them one at a time from the complete setup. The results are displayed in Figure 1.

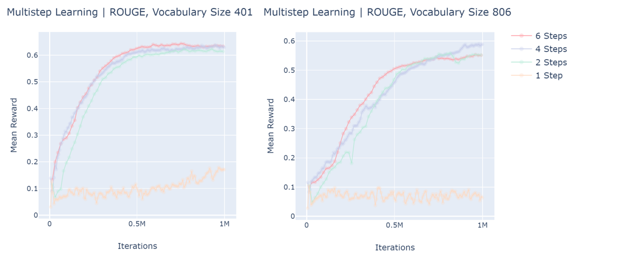
Dueling nets. Although the dueling architecture is motivated by problems with many similar-valued actions and larger action spaces, the ablation study does not indicate a significant impact of the extension on the model. No effect is observed, either in terms of performance or in light of convergence speed. These results are in line with findings by [4]. It is notable, however, that the performance of the dueling architecture lags behind in the early stages of training, although it catches up later.



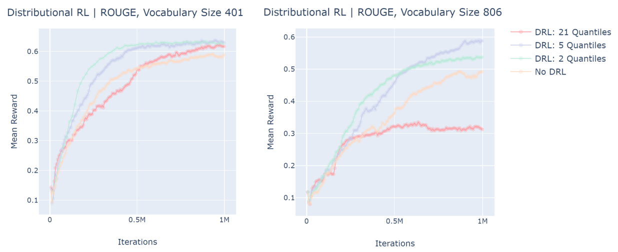
(a) Ablated dueling nets



(b) Ablated prioritized experience replay



(c) Ablated multi-step learning



(d) Ablated distributional reinforcement learning

Figure 1: Ablation for different DQN improvements

Prioritized experience replay. This extension is an essential contribution to scalability and performance, given that the results unambiguously and significantly worsen for the ablated model. To a certain extent, these results can also validate the modifications made to PER in this paper. Presumably, PER is especially effective in this setup because the memory is preloaded with demonstration data.

Multi-step learning. Based on the ablation study, it is clear that the most influential extension is multi-step learning. The ablated model did not learn at all for a vocabulary size of 806, while it performed only slightly better in a setup with 401. [4] also found it to be the most important extension, with ablation resulting in

Source	Target(s)	RL^{BLEU}	RL^{ROUGE}	SL
Of course we are. Of course.	Then you'll be my best man.	Then you'll be my best man.	Then you'll be my best, go.	Then you'll be fine.
And if I didn't want to come?	I guess I'd ask you.	I guess I'd ask you to ask me.	I guess I'd ask you why.	I guess I'd ask you to ask me.
So did you do it?	Did I kill him ? Of course not.	Did I kill him?	Did I kill him? Of course not!	Did I kill him? Of course not!
What time is it?	– Why?	Why?	What Why?	What?
	– What?			
You believe that?	You don't believe me? Okay, fine.	I do if you do.	No.	You think bad, and bad is what you get.
Are you okay?	– I'm fine.	I'll be fine.	I'm fine.	She'll be fine.
	– I'll be fine.			
	– She'll be fine.			
	– I think so.			
	– … (*8 more targets*)			
Yeah.	*132 targets*	Yeah.?	I like nothing I bad?. say	You're with me now.
She's not at home?	No … they said she was here …	No … they said she was it.	No … they said she want, that's!	No … they didn't < u > here –
She's out with a friend.	Oh … do you know when she'll be home?	Oh … do you know when she'll be back?	Oh … do you know when she'll be?	Oh … do you know when she'll be home?
Now what?	– We just don't stop.	Don't go.	We don't know it.	Now what what?
	– Now what what?			
	– Don't go.			
	– … (*2 more targets*)			

Table 3: Examples for which the presented models generate different responses; the models included are the reinforcement learning models conditioned on either BLEU or ROUGE (RL^{BLEU}, RL^{ROUGE}) and the supervised trained model (SL).

a substantial drop in early and final performance. However, the adverse effects of ablation are much more strongly reflected in the paper's setup. Presumably, the reasons for this outcome are to be found in the different nature of the problem. Here, rewards are issued only at the very end of the sequence, which is cataclysmic in combination with the last action always being the end token. In this particular case, only $q(s, \texttt{</s>})$ is able to obtain immediate, unbiased targets. Conversely, all the other actions' targets can rely solely on the model itself, via bootstrapping.

Distributional reinforcement learning. The quantile regression extension exercises a noticeable effect on the performance. That said, the model is highly sensitive to the number of quantiles N chosen. In the original paper, [19], the authors suggested N to be 32. However, the researchers only probed their models on small action spaces, which are different in magnitude compared to this paper's setup. In general, distributional RL requires the model to learn more and make auxiliary predictions, increasing the difficulty of the task while introducing synergy effects and easing approximation. Moreover, the output layer size is defined by $|\mathcal{A}| \cdot N$, which will dominate the model's size and complexity for larger action spaces, introducing a disproportion between the problem's complexity and the model's complexity. This relationship is shown in Table 4. While $N = 21$ works well on vocabulary sizes of 111 and 201, it already slightly hurts performance for 401, and it fails for 806. For the latter two sizes, a value between 2 and 5 seems to be a reasonable choice. By means of these parameters, distributional RL contributes to scalability and final performance. To conclude, there are presumably two opposite effects resulting from QR-DQN: the model benefits from learning auxiliary tasks while a larger quantile number leads to a larger model and more difficult prediction task, necessitating a careful trade-off.

6.4 Utilization of Demonstration Data

This section deals with techniques intended to utilize the provided demonstration data. The learning curves for the different models are depicted in Figure 2.

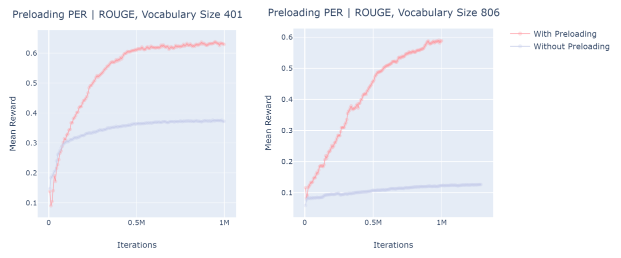
Preloading PER.. Preloading the prioritized replay buffer is demonstrated to be a key element in all presented settings. With preloading, the model does not need to rely on random sampling only but can utilize demonstration data. The number of potential sequences grows exponentially with the vocabulary size, which is why, without preloading, the model is practically unable to learn at all in a setting with a vocabulary size of 806, while it seems to converge to a suboptimal solution in a 401 setting. However, this result is notable, because in other settings like [31] preloading had less of an impact. Moreover, in a replay buffer with up to 1 million entries, the amount of demonstration data is vanishingly small.

# Quantiles N	# Parameters (401)	# Parameters (806)
No Distributional RL	4.617,305	5,306,820
2	5,020,109	6,115,434
5	6,228,521	8,541,276
21	12,673,385	21,479,100
51	24,757,505	45,737,520

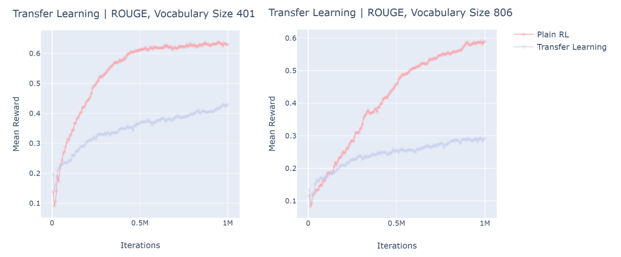
Table 4: Influence of the number of quantiles N on the number of parameters

Transfer learning. Transferring the weights of the supervised learning model to the recurrent unit and embedding layers of the RL model worsens performance significantly. While these models tend to start slightly better compared to pure RL settings, their learning curve quickly flattens, and they converge to suboptimal solutions. Through transfer learning, the hypothesis space seems to be narrowed in a disadvantageous way, hinting that predicting token probabilities based on the CE loss and predicting Q-values based on BLEU or ROUGE as a reward are very different tasks. This outcome suggests the problem is more suitable to be framed in a multitask learning than a transfer learning setup.

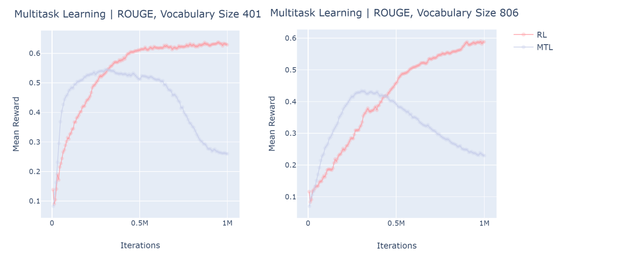
Multitask learning. The experiments conducted show multitask learning models to have a strong early performance (i.e., a comparably quite steep increase of the average reward yielded in early training stages). However, these models reach a premature plateau before they eventually diverge. Nevertheless, this result may suggest there is potential in this approach, while further exploration of multitask learning setups may help overcome the caveats in this specific setting.



(a) Preloading the prioritized experience replay buffer



(b) Utilizing transfer learning



(c) Multitask learning

Figure 2: Evaluating pretraining techniques

7 CONCLUSION

In this paper, a framework was developed, allowing the application of value-based reinforcement learning methods to sequence-to-sequence models for the first time. This framework contrasts sharply with existing approaches which focus solely on policy gradient methods and actor-critic setups because they are easy to pretrain. However, this work follows a long-term goal of making reinforcement learning approaches usable in the area of natural language processing beyond the fine-tuning of supervised trained models.

The presented model demonstrates the theoretical possibility of training a sequence-to-sequence model in a Rainbow setup, an advanced single-actor DQN agent. In practice, such a model is still highly limited by its scalability. However, it is the first step towards a generally applicable approach and an important baseline for future improvements. Furthermore, the ablation study included here provides valuable insights into the behaviour of several DQN improvements in a high-dimensional NLP setup. More specifically, multi-step learning, prioritized experience replay and distributional reinforcement learning were found to be essential components enabling the model to learn in the investigated settings. Additionally, the paper explored how demonstration data can be utilized. In this context, the preloading of the replay buffer with such data was identified as an indispensable prerequisite for learning in higher-dimensional spaces.

8 FUTURE WORK

There are several directions in which this research can be furthered. Particular attention should be given to the question of how scalability can be improved.

Recent improvements on single-actor DQNs. While Rainbow is still considered to be state-of-the-art, there have recently been some major improvements in the area of distributional reinforcement learning. Models such as IQN [32] and FQF [33] already match or even surpass the performance of Rainbow, even without combining orthogonal enhancements. Both papers encourage using their approaches to distributional RL in a Rainbow-like setup. Also, they might be especially effective for high-dimensional spaces as they avoid the excessive growth of the output layer with the number of quantiles.

Distributed DQNs. A potential approach for significantly improving the model's scalability is to switch to a distributed architecture. By decoupling data collection and learning, models such as ApeX [34], R2D2 [11], and Agent57 [35] are able to increase final performance substantially while reducing wall-clock learning speed against all single-actor agents.

Dealing with high dimensionality. Several tricks and methods are primarily motivated by vast action spaces. Most recently, a promising contribution was made with AQL in [36], whose method relies on a proposal network to suggest potential actions. With value penalties, as used in [37] or [38], an additional loss component is added, which penalizes variance on outputs helping with rare actions. Another idea is action branching proposed by [39], who architecturally divide the action space into smaller chunks.

Representation of the action space. It might be beneficial for RL problems, especially value-based methods, if actions are not formed at word-level, but at byte-level or character-level, resulting in a significant reduction of the action space. There is a chance that DQNs are able to cope better with longer action sequences than an increased action space.

ACKNOWLEDGMENTS

We thank Ngoc Quan Pham and Stefan Constantin for sparking discussions.

REFERENCES

[1] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06732.
[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099, 2015. URL http://arxiv.org/abs/1506.03099.
[3] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016. URL http://arxiv.org/abs/1606.01541.
[4] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017. URL http://arxiv.org/abs/1710.02298.

[5] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[6] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *CoRR*, abs/1607.07086, 2016. URL http://arxiv.org/abs/1607.07086.

[7] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563, 2016. URL http://arxiv.org/abs/1612.00563.

[8] Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. Reliability and Learnability of Human Bandit Feedback for Sequence-to-Sequence Reinforcement Learning. *arXiv e-prints*, art. arXiv:1805.10627, May 2018.

[9] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017. URL http://arxiv.org/abs/1705.04304.

[10] Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015. URL http://arxiv.org/abs/1507.06527.

[11] Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lyTjAqYX.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[14] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

[15] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.

[16] Tom Schaul, Rémi Munos, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[17] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015. URL http://arxiv.org/abs/1511.06581.

[18] Richard S. Sutton. Learning to predict by the methods of temporal differences. In *MACHINE LEARNING*, pages 9–44, 1988.

[19] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional Reinforcement Learning with Quantile Regression. *arXiv e-prints*, art. arXiv:1710.10044, October 2017.

[20] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *CoRR*, abs/1707.06887, 2017. URL http://arxiv.org/abs/1707.06887.

[21] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. *CoRR*, abs/1706.10295, 2017. URL http://arxiv.org/abs/1706.10295.

[22] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

[23] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-1013.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

[25] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL http://arxiv.org/abs/1411.1792.

[26] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1046. URL https://www.aclweb.org/anthology/D16-1046.

[27] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL http://doi.acm.org/10.1145/1390156.1390177.

[28] Ross Girshick. Fast R-CNN. *arXiv e-prints*, art. arXiv:1504.08083, April 2015.

[29] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *ICML*, 1993.

[30] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. *arXiv e-prints*, art. arXiv:1106.3077, June 2011.

[31] Emil Larrson. Evaluation of pretraining methods for deep reinforcement learning, 2018.

[32] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit Quantile Networks for Distributional Reinforcement Learning. *arXiv e-prints*, art. arXiv:1806.06923, June 2018.

[33] Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tieyan Liu. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. *arXiv e-prints*, art. arXiv:1911.02140, November 2019.

[34] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. *CoRR*, abs/1803.00933, 2018. URL http://arxiv.org/abs/1803.00933.

[35] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari Human Benchmark. *arXiv e-prints*, art. arXiv:2003.13350, March 2020.

[36] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *CoRR*, abs/2001.08116, 2020. URL https://arxiv.org/abs/2001.08116.

[37] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2015. URL http://arxiv.org/abs/1409.0473.

[38] Wojciech Zaremba, Tomas Mikolov, Armand Joulin, and Rob Fergus. Learning simple algorithms from examples. *CoRR*, abs/1511.07275, 2015. URL http://arxiv.org/abs/1511.07275.

[39] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. *CoRR*, abs/1711.08946, 2017. URL http://arxiv.org/abs/1711.08946.

Appendices

A REWARDS

A.1 BLEU

BLEU is a metric first presented by Papineni et al. in [22]. It is primarily used in machine translation and other language generation problems. The values range from 0 to 1. BLEU calculates a modified n-gram precision p_n, for which it counts the number of matches between the n-grams of the candidate and the n-grams of the reference divided by the total number of n-grams in the candidate. In order to prevent abundances of high-frequency words, the number of matches for a word is clipped after its maximum reference count. However, because this measure still enables very short candidates to achieve high-scoring results, a brevity penalty as a multiplicative factor is introduced to mimic some kind of recall.

$$\mathrm{BP} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-\frac{r}{c})}, & \text{if } c \leq r \end{cases} \tag{2}$$

It is possible to combine the scores of different n-gram sizes by calculating the geometric mean. The final equation is given by:

$$\mathrm{BLEU} = \mathrm{BP} \cdot \left(\sum_{n=1}^{N} w_n \log(p_n) \right) \tag{3}$$

where the weight w_n is usually the uniform distribution $1/N$.

The original definition of the brevity penalty, as indicated in Equation 2, has no solution at $c = 0$. This is particularly problematic in this RL setting with Q-learning, as the model tends to generate empty candidates in the early stages of training. Nevertheless, the generated candidates require evaluation in order to add them to the experience replay buffer. Hence, as part of this work, the brevity penalty is defined as zero if $c = 0$, which is tantamount to $\mathrm{BLEU} = 0$.

$$\mathrm{BP} = \begin{cases} 0, & \text{if } c = 0 \\ 1, & \text{if } c > r \\ e^{(1-\frac{r}{c})}, & \text{if } 0 < c \leq r \end{cases} \tag{4}$$

A.2 ROUGE

ROUGE was presented in [23] and while it has especially been developed to evaluate text summarization tasks, it can be applied to all kinds of language generation problems.

As part of this work, more precisely, ROUGE-W functions as the reward and as an evaluation metric. In contrast to BLEU and other versions of ROUGE, the longest common subsequence (LCS) is determined instead of n-gram overlaps. This approach means consecutive matches are not required, as it allows in-sequence matches on sentence-level order. Moreover, no predefined n-gram length needs to be specified, and it works for any sequence length. One drawback, however, is that consecutive matches are assigned the same score as non-consecutive matches. To address this issue, with ROUGE-W weights are introduced. The F1 metric is applied to take recall and precision equally into account.

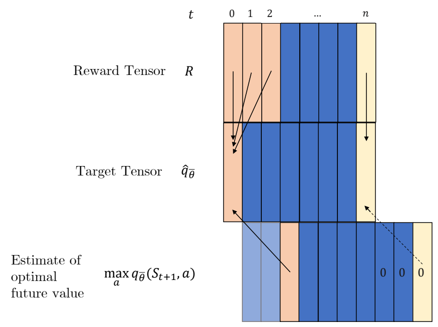

t
0
1
2
...
n
Reward Tensor R
Target Tensor $\hat{q}_{\bar{\theta}}$
Estimate of optimal future value $\max_a q_{\bar{\theta}}(S_{t+1}, a)$
0
0
0

Figure 3: Calculating targets for multi-step Q-learning in a seq2seq setup

B IMPLEMENTATION

B.1 Episodes

Most of the implementation effort is required because the presented model does not work with batches of single transitions as in Rainbow reference implementations, but rather with batches of whole episodes (i.e., sequences of transitions). This difference adds another dimension to the tensors and calculations.

The additional complexity can be observed, for instance, when calculating the target $\hat{q}_{\bar{\theta}}$ for multi-step learning:

$$\hat{q}_{\bar{\theta}} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{(n)} \max_{a'} q_{\bar{\theta}}(S_{t+n}, a') \tag{5}$$

This procedure requires the combination of the reward tensor R and the tensor for estimates of optimal future values $\max_{a'} q_{\bar{\theta}}(S_{t+n}, a')$, as visualized in Figure 3. Due to taking multi-steps, a window of n rewards must be considered at a specific time step t while there is a $n-1$ shift for the value estimate tensor. To obtain the target via simple addition of the tensors, the estimates tensor can be transformed to be of the same shape, discarding the first $n-1$ steps while zero-padding n final states. The reward tensor, conversely, can be dewindowed utilizing convolutions (see Section B.3).

Furthermore, the differences in length of the episodes necessitate the careful application of sequence padding and masking as one proceeds.

B.2 Normalization

In many implementations of seq2seq models, bucketing and padding of sequences is applied. However, bucketing cannot be used in conjunction with an experience replay buffer which samples whole episodes (i.e., sequences instead of single transitions). The lengths of sequences in a batch are completely randomized, which may lead to a considerable gap between minimum and maximum sequence length in a batch. In early experiments, this led to the effect that normalization of the loss by the number of tokens is superior to normalization by the batch size or no normalization at all.

B.3 Multi-Step Learning as Convolution

For the multi-step learning case, the target is obtained by $R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{(n)} \max_{a'} q_{\bar{\theta}}(S_{t+n}, a')$. Thus, instead of utilizing a single reward, the next n steps are summed while being exponentially decayed using the discount factor γ. However, in sharp contrast to most applications of multi-step learning, this paper works with batches of sequences, which allows viewing the term as a convolution. Specifically, it is a one-dimensional, axis-aligned convolution whose input array's values beyond the edge are filled with a constant value of 0. The kernel can be calculated in advance with $[\gamma^{n-1}, \gamma^{n-2}, \ldots, \gamma, 1]$.