NATURAL LANGUAGE STATE REPRESENTATION FOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in Reinforcement Learning have highlighted the difficulties in learning within complex high dimensional domains. We argue that one of the main reasons that current approaches do not perform well, is that the information is represented sub-optimally. A natural way to describe what we observe, is through natural language. In this paper, we implement a natural language state representation to learn and complete tasks. Our experiments suggest that natural language based agents are more robust, converge faster and perform better than vision based agents, showing the benefit of using natural language representations for Reinforcement Learning.

1 INTRODUCTION

"The world of our experiences must be enormously simplified and generalized before it is possible to make a symbolic inventory of all our experiences of things and relations."

(Edward Sapir, Language: An Introduction to the Study of Speech, 1921)

Deep Learning based algorithms use neural networks in order to learn feature representations that are good for solving high dimensional Machine Learning (ML) tasks. Reinforcement Learning (RL) is a subfield of ML that has been greatly affected by the use of deep neural networks as universal function approximators [Csáji, 2001; Lu et al., 2017]. These deep neural networks are used in RL to estimate value functions, state-action value functions, policy mappings, next-state predictions, rewards, and more [Mnih et al., 2015; Schulman et al., 2017; Hafner et al., 2018], thus combating the "curse of dimensionality".

The term representation is used differently in different contexts. For the purpose of this paper we define a **semantic representation** of a state as one that reflects its meaning as it is understood by an expert. The semantic representation of a state should thus be paired with a reliable and computationally efficient method for extracting information from it. Previous success in RL has mainly focused on representing the state in its raw form (e.g., visual input in Atari-based games [Mnih et al., 2015]). This approach stems from the belief that neural networks (specifically convolutional networks) can extract meaningful features from complex inputs. In this work, we challenge current representation techniques and suggest to represent the state using natural language, similar to the way we, as humans, summarize and transfer information efficiently from one to the other [Sapir, 2004].

The ability to associate states with natural language sentences that describe them is a hallmark of understanding representations for reinforcement learning. Humans use rich natural language to describe and communicate their visual perceptions, feelings, beliefs, strategies, and more. The semantics inherent to natural language carry knowledge and cues of complex types of content, including: events, spatial relations, temporal relations, semantic roles, logical structures, support for inference and entailment, as well as predicates and arguments [Abend & Rappoport, 2017]. The expressive nature of language can thus act as an alternative semantic state representation.

Over the past few years, Natural Language Processing (NLP) has shown an acceleration in progress on a wide range of downstream applications ranging from Question Answering [Kumar et al., 2016; Liu et al., 2018], to Natural Language Inference [Parikh et al., 2016; Chen et al., 2017; 2018] through Syntactic Parsing [Williams et al., 2018; Shi et al., 2018; Shen et al., 2019]. Recent work has shown

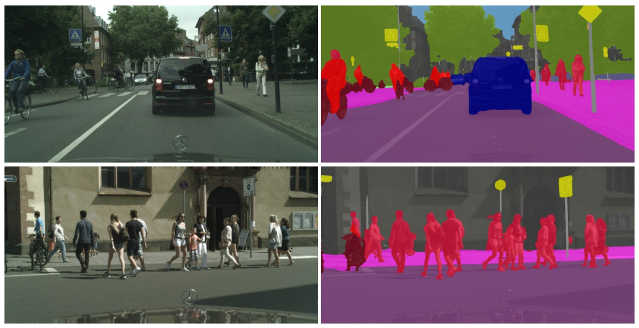


Figure 1: Example of Semantic Segmentation [Kundu et al., 2016].

the ability to learn flexible, hierarchical, contextualized representations, obtaining state-of-the-art results on various natural language processing tasks [Devlin et al., 2019]. A basic observation of our work is that natural language representations are also beneficial for solving problems in which natural language is not the underlying source of input. Moreover, our results indicate that natural language is a strong alternative to current complementary methods for semantic representations of a state.

In this work we assume a state can be described using natural language sentences. We use distributional embedding methods[1] in order to represent sentences, processed with a standard Convolutional Neural Network for feature extraction. In Section 2 we describe the basic frameworks we rely on. We discuss possible semantic representations in Section 3, namely, raw visual inputs, semantic segmentation, feature vectors, and natural language representations. Then, in Section 4 we compare NLP representations with their alternatives. Our results suggest that representation of the state using natural language can achieve better performance, even on difficult tasks, or tasks in which the description of the state is saturated with task-nuisances [Achille & Soatto, 2018]. Moreover, we observe that NLP representations are more robust to transfer and changes in the environment. We conclude the paper with a short discussion and related work.

2 PRELIMINARIES

2.1 REINFORCEMENT LEARNING

In Reinforcement Learning the goal is to learn a policy $\pi(s)$, which is a mapping from state s to a probability distribution over actions $\mathcal{A}$, with the objective to maximize a reward $r(s)$ that is provided by the environment. This is often solved by formulating the problem as a Markov Decision Process (MDP) [Sutton & Barto, 1998]. Two common quantities used to estimate the performance in MDPs are the value $v(s)$ and action-value $Q(s, a)$ functions, which are defined as follows: $v(s) = \mathbb{E}^\pi[\sum_t \gamma^t r_t | s_0 = s]$ and $Q(s, a) = \mathbb{E}^\pi[\sum_t \gamma^t r_t | s_0 = s, a_0 = a]$. Two prominent algorithms for solving RL tasks, which we use in this paper, are the value-based DQN [Mnih et al., 2015] and the policy-based PPO [Schulman et al., 2017].

Deep Q Networks (DQN): The DQN algorithm is an extension of the classical Q-learning approach, to a deep learning regime. Q-learning learns the optimal policy by directly learning the value function, i.e., the action-value function. A neural network is used to estimate the Q-values and is trained to minimize the Bellman error, namely

$$||r(s) + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s, a)||_2^2 \,.$$

Proximal Policy Optimization (PPO): While the DQN learns the optimal behavioral policy using a dynamic programming approach, PPO takes a different route. PPO builds upon the policy gradient

[1]Distributional methods make use of the hypothesis that words which occur in a similar context tend to have similar meaning [Firth, 1957], i.e., the meaning of a word can be inferred from the distribution of words around it. For this reason, these methods are called distributional methods.

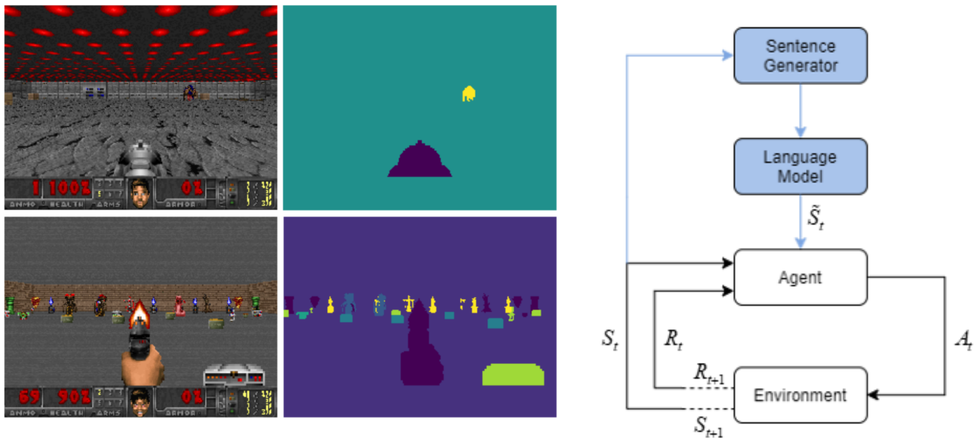


Figure 2: Left: Raw visual inputs and their corresponding semantic segmentation in the VizDoom enviornment. Right: Our suggested NLP-based semantic state representation framework.

theorem, which optimizes the policy directly, with an addition of a trust-region update rule. The policy gradient theorem updates the policy by

$$\nabla_\theta \log \pi_\theta(a|s) \mid_{\theta=\theta_k} Q^{\pi_{\theta_k}}(s,a) \,.$$

2.2 Deep Learning for NLP

A word embedding is a mapping from a word w to a vector $\mathbf{w} \in \mathbb{R}^d$. A simple form of word embedding is the Bag of Words (BoW), a vector $\mathbf{w} \in \mathbb{N}^{|D|}$ ($|D|$ is the dictionary size), in which each word receives a unique 1-hot vector representation. Recently, more efficient methods have been proposed, in which the embedding vector is smaller than the dictionary size, $d \ll |D|$. These methods are also known as distributional embeddings.

The distributional hypothesis in linguistics is derived from the semantic theory of language usage (i.e. words that are used and occur in the same contexts tend to have similar meanings). Distributional word representations are a fundamental building block for representing natural language sentences. Word embeddings such as Word2vec [Mikolov et al., 2013] and GloVe [Pennington et al., 2014] build upon the distributional hypothesis, improving efficiency of state-of-the-art language models.

Convolutional Neural Networks (CNNs), originally invented for computer vision, have been shown to achieve strong performance on text classification tasks [Johnson & Zhang, 2015; Bai et al., 2018], as well as other traditional NLP tasks [Collobert et al., 2011]. In this paper we consider a common architecture [Kim, 2014], in which each word in a sentence is represented as an embedding vector, a single convolutional layer with m filters is applied, producing an m-dimensional vector for each n-gram. The vectors are combined using max-pooling followed by a ReLU activation. The result is then passed through multiple hidden linear layers with ReLU activation, eventually generating the final output.

3 Semantic Representation Methods

Contemporary methods for semantic representation of states currently follow one of three approaches: (1) raw visual inputs [Mnih et al., 2015; Kempka et al., 2016], in which raw sensory values of pixels are used from one or multiple sources, (2) feature vectors [Todorov et al., 2012; He et al., 2018], in which general features of the problem are chosen, with no specific structure, and (3) semantic segmentation maps [Ronneberger et al., 2015; He et al., 2017], in which discrete or logical values are used in one or many channels to represent the general features of the state.

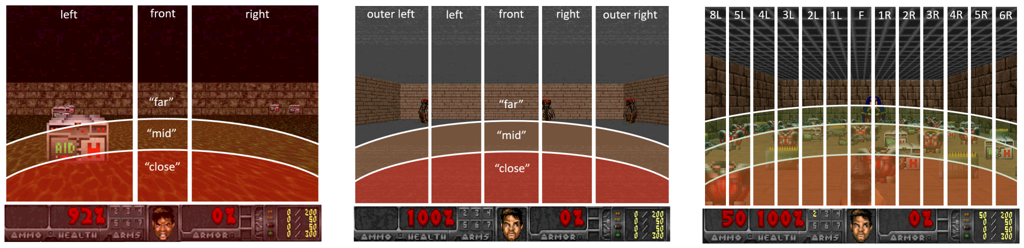


Figure 3: Frame division used for describing the state in natural language.

The common approach is to derive decisions (e.g., classification, action, etc.) based on information in its raw form. In RL, the raw form is often the pixels representing an image – however the image is only one form of a semantic representation. In Semantic Segmentation, the image is converted from a 3-channel (RGB) matrix into an N-channel matrix, where N is the number of classes. In this case, each channel represents a class, and a binary value at each coordinate denotes whether or not this class is present in the image at this location. For instance, Fig. 1 considers an autonomous vehicle task. The raw image and segmentation maps are both sufficient for the task (i.e., both contain a sufficient semantic representation). Nevertheless, the semantic segmentation maps contain less **task-nuisances** [Achille & Soatto, 2018], which are random variables that affect the observed data, but are not informative to the task we are trying to solve.

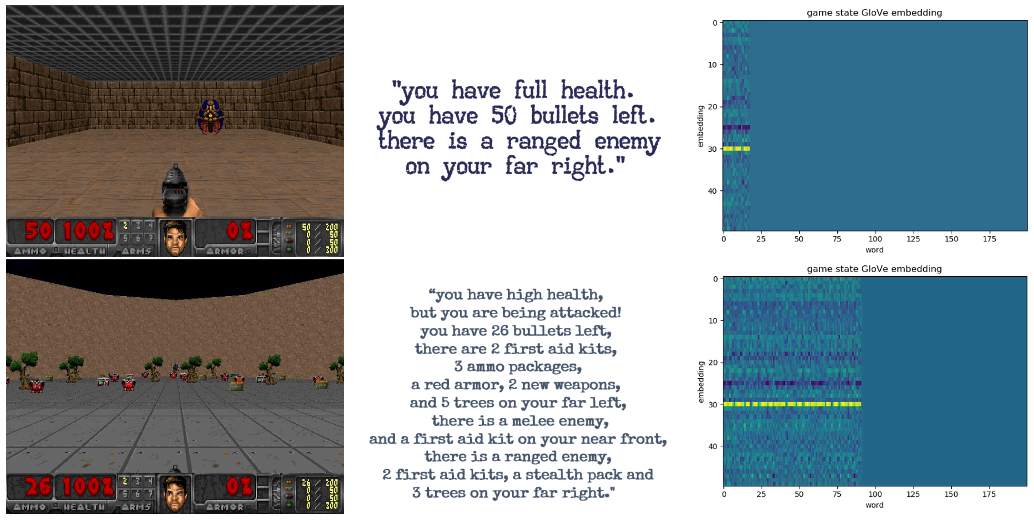
In this paper we propose a forth method for representing a state, namely using natural language descriptions. One method to achieve such a representation is through Image Captioning [Tran et al., 2016; Hossain et al., 2019]. Natural language is both rich as well as flexible. This flexibility enables the algorithm designer to represent the information present in the state as efficiently and compactly as possible. As an example, the top image in Fig. 1 can be represented using natural language as "*There is a car in your lane two meters in front of you, a bicycle rider on your far left in the negative lane, a car in your direction in the opposite lane which is twenty meters away, and trees and pedestrians on the side walk.*" or compactly by "*There is a car two meters in front of you a pedestrian on the sidewalk to your right and a car inbound in the negative lane which is far away.*". Language also allows us to efficiently compress information. As an example, the segmentation map in the bottom image of Fig. 1 can be compactly described by "*There are 13 pedestrians crossing the road in front of you*". In the next section we will demonstrate the benefits of using natural-language-based semantic state representation in a first person shooter enviornment.

4 Semantic State Representations in the Doom Environment

In this section we compare the different types of semantic representations for representing states in the ViZDoom environment [Kempka et al., 2016], as described in the previous section. More specifically, we use a semantic natural language parser in order to describe a state, over numerous instances of levels varying in difficulty, task-nuisances, and objectives. Our results show that, though semantic segmentation and feature vector representation techniques express a similar statistic of the state, natural language representation offers better performance, faster convergence, more robust solutions, as well as better transfer.

The ViZDoom environment involves a 3D world that is significantly more real-world-like than Atari 2600 games, with a relatively realistic physics model. An agent in the ViZDoom environment must effectively perceive, interpret, and learn the 3D world in order to make tactical and strategic decisions of where to go and how to act. There are three types of state representations that are provided by the environment. The first, which is also most commonly used, is raw visual inputs, in which the state is represented by an image from a first person view of the agent. A feature vector representation is an additional state representation provided by the environment. The feature vector representation includes positions as well as labels of all objects and creatures in the vicinity of the agent. Lastly, the environment provides a semantic segmentation map based on the aforementioned feature vector. An example of the visual representations in VizDoom is shown in Section 2.2.

In order to incorporate natural language representation to the VizDoom environment we've constructed a semantic parser of the semantic segmentation maps provided by the environment. Each


"you have full health. you have 50 bullets left. there is a ranged enemy on your far right."

"you have high health, but you are being attacked! you have 26 bullets left, there are 2 first aid kits, 3 ammo packages, a red armor, 2 new weapons, and 5 trees on your far left, there is a melee enemy, and a first aid kit on your near front, there is a ranged enemy, 2 first aid kits, a stealth pack and 3 trees on your far right."

Figure 4: Natural language state representation for a simple state (top) and complex state (bottom). The corresponding embedded representations and shown on the right.

state of the environment was converted into a natural language sentence based on positions and labels of objects in the frame. To implement this, the screen was divided into several vertical and horizontal patches, as depicted in Fig. 3. These patches describe relational aspects of the state, such as distance of objects and their direction with respect to the agent's point of view. In each patch, objects were counted, and a natural language description of the patch was constructed. This technique was repeated for all patches to form the final state representation. Fig. 4 depicts examples of natural language sentences of different states in the enviornment.

4.1 Experiments

We tested the natural language representation against the visual-based and feature representations on several tasks, with varying difficulty. In these tasks, the agent could navigate, shoot, and collect items such as weapons and medipacks. Often, enemies of different types attacked the agent, and a positive reward was given when an enemy was killed. Occasionally, the agent also suffered from health degeneration. The tasks included a basic scenario, a health gathering scenario, a scenario in which the agent must take cover from fireballs, a scenario in which the agent must defend itself from charging enemies, and a super scenario, where a mixture of the above scenarios was designed to challenge the agent.

More specifically, in the **basic scenario**, a single monster is spawned in front of the agent. The purpose of this scenario is to teach the agent to aim at the enemy and shoot at it. In the **health gathering** scenario, the floor of the room is covered in toxin, causing the agent to gradually lose health. Medipacks are spawned randomly in the room and the agent's objective is to keep itself alive by collecting them. In the **take cover** scenario, multiple fireball shooting monsters are spawned in front of the agent. The goal of the agent is to stay alive as long as possible, dodging inbound fireballs. The difficulty of the task increases over time, as additional monsters are spawned. In the **defend the center** scenario, melee attacking monsters are randomly spawned in the room, and charge towards the agent. As opposed to other scenarios, the agent is incapable of moving, aside from turning left and right and shooting. In the **defend the line** scenario, both melee and fireball shooting monsters are spawned near the opposing wall. The agent can only step right, left or shoot. Finally, in the **"super" scenario** both melee and fireball shooting monsters are repeatably spawned all over the room. the room contains various items the agent can pick up and use, such as medipacks, shotguns, ammunition and armor. Furthermore, the room is filled with unusable objects, various types of trees, pillars and other decorations. The agent can freely move and turn in any direction, as well as shoot. This scenario combines elements from all of the previous scenarios.

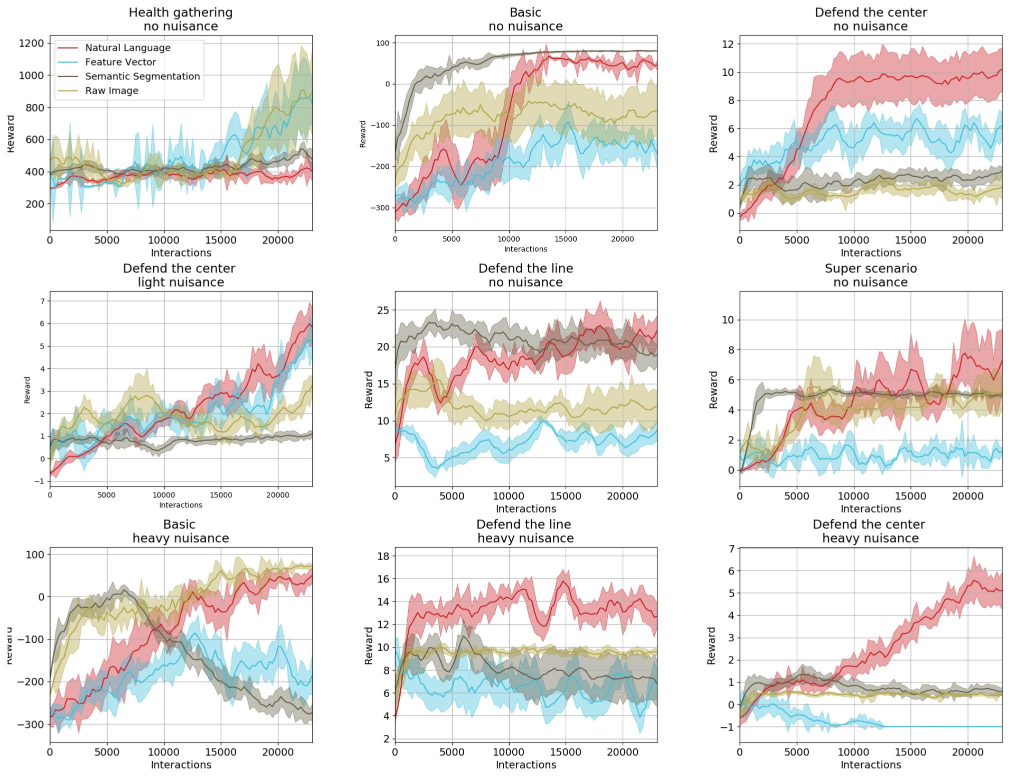


Figure 5: Comparison of representation methods on the different VizDoom scenarios using a DQN agent. X and Y axes represent the number of iterations and cumulative reward, respectively. Last three graphs (bottom) depict nuisance-augmented scenarios.

Our agent was implemented using a Convolutional Neural Network as described in Section 2.2. We converted the parsed state into embedded representations of fixed length. We tested both a DQN and a PPO based agent, and compared the natural language representation to the other representation techniques, namely the raw image, feature vector, and semantic segmentation representations.

In order to effectively compare the performance of the different representation methods, we conducted our experiments under similar conditions for all agents. The same hyper-parameters were used under all tested representations. Moreover, to rule out effects of architectural expressiveness, the number of weights in all neural networks was approximately matched, regardless of the input type. Finally, we ensured the "super" scenario was positively biased toward image-based representations. This was done by adding a large amount items to the game level, thereby filling the state with nuisances (these tests are denoted by 'nuisance' in the scenario name). This was especially evident in the NLP representations, as sentences became extensively longer (average of over 250 words). This is contrary to image-based representations, which did not change in dimension.

Results of the DQN-based agent are presented in Fig. 5. Each plot depicts the average reward (across 5 seeds) of all representations methods. It can be seen that the NLP representation outperforms the other methods. This is contrary to the fact that it contains the same information as the semantic segmentation maps. More interestingly, comparing the vision-based and feature-based representations render inconsistent conclusions with respect to their relative performance. NLP representations remain robust to changes in the environment as well as task-nuisances in the state. As depicted in Fig. 6, inflating the state space with task-nuisances impairs the performance of all representations. There, a large amount of unnecessary objects were spawned in the level, increasing the state's description length to over 250 words, whilst retaining the same amount of useful information. Nevertheless, the NLP representation outperformed the vision and feature based representations, with high robustness to the applied noise.

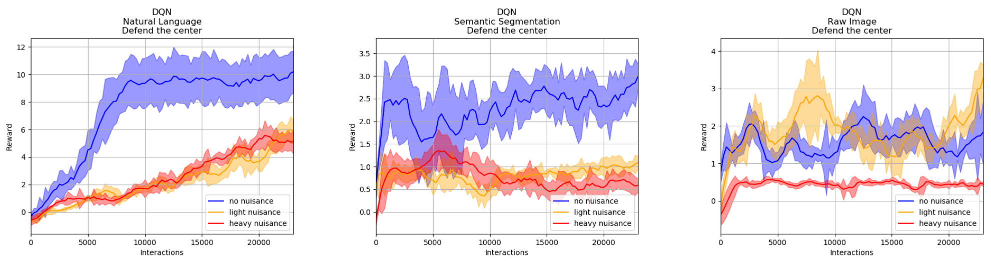


Figure 6: Robustness of each representation type with respect to amount of nuisance.

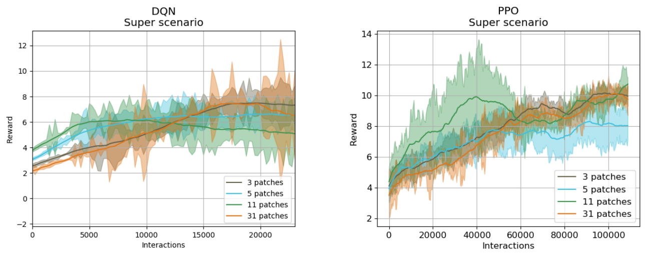


Figure 7: Average rewards of NLP based agent as a function of the number of patches in the language model.

In order to verify the performance of the natural language representation was not due to extensive discretization of patches, we've conducted experiments increasing the number of horizontal patches - ranging from 3 to 31 patches in the extreme case. Our results, as depicted in Fig. 7, indicate that the amount of discretization of patches did not affect the performance of the NLP agent, remaining a superior representation compared to the rest.

To conclude, our experiments suggest that NLP representations, though they describe the same raw information of the semantic segmentation maps, are more robust to task-nuisances, allow for better transfer, and achieve higher performance in complex tasks, even when their description is long and convoluted. While we've only presented results for DQN agents, we include plots for a PPO agent in the Appendix, showing similar trends and conclusions. We thus deduce that NLP-based semantic state representations are a preferable choice for training VizDoom agents.

5 Related Work

Work on representation learning is concerned with finding an appropriate representation of data in order to perform a machine learning task [Goodfellow et al., 2016]. In particular, deep learning exploits this concept by its very nature [Mnih et al., 2015]. Work on representation learning include Predictive State Representations (PSR) [Littman & Sutton, 2002; Jiang et al., 2016], which capture the state as a vector of predictions of future outcomes, and a Heuristic Embedding of Markov Processes (HEMP) [Engel & Mannor, 2001], which learns to embed transition probabilities using an energy-based optimization problem.

There has been extensive work attempting to use natural language in RL. Efforts that integrate language in RL develop tools, approaches, and insights that are valuable for improving the generalization and sample efficiency of learning agents. Previous work on language-conditioned RL has considered the use of natural language in the observation and action space. Environments such as Zork and TextWorld [Côté et al., 2018] have been the standard benchmarks for testing text-based games. Nevertheless, these environments do not search for semantic state representations, in which an RL algorithm can be better evaluated and controlled.

Eisenstein et al. [2009] use high-level semantic abstractions of documents in a representation to facilitate relational learning using Inductive Logic Programming and a generative language model. Branavan et al. [2012] use high-level guidance expressed in text to enrich a stochastic agent, playing against the built-in AI of Civilization II. They train an agent with the Monte-Carlo search framework in order to jointly learn to identify text that is relevant to a given game state as well as game strategies based only on environment feedback. Narasimhan et al. [2018] utilize natural language in a model-based approach to describe the dynamics and rewards of an environment, showing these can facilitate transfer between different domains.

More recently, the structure and compositionality of natural language has been used for representing policies in hierarchical RL. In a paper by Hu et al. [2019], instructions given in natural language were used in order to break down complex problems into high-level plans and lower-level actions. Their suggested framework leverages the structure inherent to natural language, allowing for transfer to unfamiliar tasks and situations. This use of semantic structure has also been leveraged by Tennenholtz & Mannor [2019], where abstract actions (not necessarily words) were recognized as symbols of a natural and expressive language, improving performance and transfer of RL agents.

Outside the context of RL, previous work has also shown that high-quality linguistic representations can assist in cross-modal transfer, such as using semantic relationships between labels for zero-shot transfer in image classification [Socher et al., 2013; Frome et al., 2013].

6 Discussion and Future Work

Our results indicate that natural language can outperform, and sometime even replace, vision-based representations. Nevertheless, natural language representations can also have disadvantages in various scenarios. For one, they require the designer to be able to describe the state exactly, whether by a rule-based or learned parser. Second, they abstract notions of the state space that the designer may not realize are necessary for solving the problem. As such, semantic representations should be carefully chosen, similar to the process of reward shaping or choosing a training algorithm. Here, we enumerate three instances in which we believe natural language representations are beneficial:

Natural use-case: Information contained in both generic and task-specific textual corpora may be highly valuable for decision making. This case assumes the state can either be easily described using natural language or is already in a natural language state. This includes examples such as user-based domains, in which user profiles and comments are part of the state, or the stock market, in which stocks are described by analysts and other readily available text. 3D physical environments such as VizDoom also fall into this category, as semantic segmentation maps can be easily described using natural language.
Subjective information: Subjectivity refers to aspects used to express opinions, evaluations, and speculations. These may include strategies for a game, the way a doctor feels about her patient, the mood of a driver, and more.
Unstructured information: In these cases, features might be measured by different units, with an arbitrary position in the state's feature vector, rendering them sensitive to permutations. Such state representations are thus hard to process using neural networks. As an example, the medical domain may contain numerous features describing the vitals of a patient. These raw features, when observed by an expert, can be efficiently described using natural language. Moreover, they allow an expert to efficiently add subjective information.

An orthogonal line of research considers automating the process of image annotation. The noise added from the supervised or unsupervised process serves as a great challenge for natural language representation. We suspect the noise accumulated by this procedure would require additional information to be added to the state (e.g., past information). Nevertheless, as we have shown in this paper, such information can be compressed using natural language. In addition, while we have only considered spatial features of the state, information such as movement directions and transient features can be efficiently encoded as well.

Natural language representations help abstract information and interpret the state of an agent, improving its overall performance. Nevertheless, it is imperative to choose a representation that best fits the domain at hand. Designers of RL algorithms should consider searching for a semantic representation that fits their needs. While this work only takes a first step toward finding better semantic state representations, we believe the structure inherent in natural language can be considered a favorable candidate for achieving this goal.

REFERENCES

Omri Abend and Ari Rappoport. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 77–89, 2017.

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

SRK Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1657–1668, 2017.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2406–2417, 2018.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018.

Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pp. 958–967. Association for Computational Linguistics, 2009.

Yaakov Engel and Shie Mannor. Learning embedded maps of markov processes. In *in Proceedings of ICML 2001*. Citeseer, 2001.

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pp. 2121–2129, 2013.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018.

MD Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CSUR)*, 51(6):118, 2019.

Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. Hierarchical decision making by generating and following natural language instructions. In *Advances in neural information processing systems*, 2019.

Nan Jiang, Alex Kulesza, and Satinder Singh. Improving predictive state representations via gradient descent. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 103–112, 2015.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. IEEE, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pp. 1378–1387, 2016.

Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3168–3175, 2016.

Michael L Littman and Richard S Sutton. Predictive representations of state. In *Advances in neural information processing systems*, pp. 1555–1561, 2002.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1694–1704, 2018.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pp. 6231–6239, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874, 2018.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255, 2016.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Edward Sapir. *Language: An introduction to the study of speech*. Courier Corporation, 2004.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4631–4641, 2018.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pp. 935–943, 2013.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Guy Tennenholtz and Shie Mannor. The natural language of actions. In *International Conference on Machine Learning*, pp. 6196–6205, 2019.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. Rich image captioning in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 49–56, 2016.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267, 2018.

7 APPENDIX

ViZDoom is a ”Doom” based research environment that was developed at the Pozna University of Technology. It is based on ”ZDoom” game executable, and includes a Python based API. The API offers the user the ability to run game instances, query the game state, and execute actions. the original purpose of VizDoom is to provide a research platform for vision based reinforcement learning. Thus, a natural language representation for the game was needed to be implemented. ViZDoom emulates the ”Doom” game and enables us to access data within a certain frame using Python dictionaries. this makes it possible to extract valuable data including player health, ammo, enemy locations etc. Each game frame contains ”labels”, which contains data on visible objects in the game (the player, enemies, medkits, etc). basically, anything that is not a wall, floor or a ceiling has a label. We used ”Doom Builder” in order to edit some of the scenarios and design a new one. enviroment rewards are presented in Section 7.0.3.

7.0.1 NATURAL LANGUAGE STATE SPACE

Natural language representation should contain data which can be deduced by a human playing the game. For example, even though a human does not know the exact distance between objects, it can classify them as ”close” or ”far”. However, objects that are outside the player’s field of vision can not be a part of the state. Furthermore, a human would most likely refer to object locations in relative to itself, using simple directions such as ”right” or ”left”.

7.0.2 LANGUAGE MODEL IMPLEMENTATION

To convert each frame to a natural language representation state, the list of available labels is iterated, and a string is built accordingly. The main idea of our implementation was to divide the screen into multiple vertical patches, count the amount of different objects inside by their types, and parse it as a sentence. the decision whether an object is close or far can be determined my calculating the distance from it to the player, and using two threshold levels. Object descriptions can be concise or detailed, as needed. we experimented with said mechanic:

- **Patch Size** the screen can be divided between patches equally, or by determined rations. Here, our main guideline was to keep the ”front” patch narrow enough so it can be used as ”sights”.
- **Patch Count** our initial experiment was with 3 patches, and later we added 2 more patches classified as ”outer left” and ”outer right”.
- **Object Description** initially, we referred to all of the monster labels as ”enemy”. however, in some scenarios it might be better to distinguish between enemy types. for example, we might want to prioritize eliminating enemies that can attack from range over melee enemies.
- **Distance Thresholds** we used 2 thresholds, which allowed us to classify the distance of an object from the player as ”close”,”mid”, and ”far. depending on the task, the value of the threshold can be changed, as well as adding more thresholds.
- **Sentence Length** different states might generate sentence with different size. a maximum sentence length is another parameter that was tested.

All those parameters affect the length of the natural language sentence representing it. Obviously, The number of objects in the states affect it’s length as well. Section 7.0.3 presents some data regarding the average word count in some of the game sceanrios.

After the sentence describing the state is generated, it is preprocessed before being inserted to the model. we used Gensim to remove special characters extra spaces. As previously mentioned, each word in the natural language state is transformed to a Dx1 embedding vector. Words that were not found in the vocabulary were replaced with an ”OOV” vector. Then, all the vectors of the same state are concatenated to a NxDx1 matrix, representing the state. We experimented with both Word2Vec and GloVe pretrained embedding vectors. Eventually, we used the latter, as it consumes less memory and speeds up the training process. The length of the state sentence is one of the hyperparameters of the agents; shorter sentences are zero padded, where longer ones are trimmed. In order to make the state representation efficient without losing much of it’s data, we chose maximum sentence length of 200 words, and the GloVe dimension, Consequentially, the agent’s input is a 200X50X1 matrix.

7.0.3 Model implementation

All of our models were implemented using PyTorch. The DQN agents used a single network that outputs the Q-Values of the available actions. The PPO agents used an Actor-Critic model with two networks; the first outputs the policy distribution for the input state, and the second network outputs it's value. As mentioned earlier, we used 3 main common neural network architectures:

- **Convolutional Neural Network** used for the raw image and semantic segmentation based agents. VizDoom's raw output image resolution is 640X480X3 RGB image. we experimented with both the original image and it's down-sampled version. the semantic segmentation image is of resolution 640X480X1, where the pixel value represents the object's classification. It was generated using the VizDoom label API and simple processing. the network consisted of two convolutional layers, two hidden linear layers and an output layer. the first convolutional layer has 8 6X6 filters with stride 3 with ReLU activation. the second convolutional layer has 16 3X3 filters with stride 2 with ReLU activation. the fully connected layers has 32 and 16 units, both of them are followed by ReLU activation. the output layer's size is the amount of actions the agent has available in the trained scenario.
- **Multilayer Perceptron** Used in the feature vector based agent. Naturally, some discretization is needed in order to build a feature vector, so some of the state data is lost. the feature vector was made using features we extracted from the VizDoom API, and it's dimensions were 90 X 1. The network is made of two hidden fully connected layers, each of them followed with a ReLU activation. The first layer has 32 units, and the second one one has 16 units. The output layer's size was the amount of actions available to the agent.
- **TextCNN** Used in the natural language based agent. As previously mentioned, each word in the natural language state is transformed to a 200X50X1 matrix. the first layers of the TextCNN are convolutional layers with 8 filter which are designed to scan input sentence, and return convolution outputs of sequences of varying lengths.the filters varying in width, so that each of them will learn to identify different lengths of sequences in words. Longer filters have higher capability of extracting features from longer word sequences. the filters we have chosen have the following dimensions: 3X50X1, 4X50X1, 5X50X1, 8X50X1,11X50X1. Following the convolution layer there is a ReLU activation and a max pool layer. the max pool shrinks the data dimension, adds non-linearity, as well as in-variance to sequence locations in the input. After that there are two fully connected layers; The first layer has 32 units, and second one has 16 units. Both of them are followed by ReLU activation.

All of the architectures has the same output, regardless of the input type. The DQN network is a regression network it's and it's output unit's size is the amount of available actions, each indicating the respective Q-Value. The PPO agent has 2 networks; actor and critic. the actor network has a Softmax activation with size equals to the available amount of actions. the critic network is a regression model, so it has a single output representing the state's value

There are probably better network architectures and hyperparameters, which can optimize an agent's reward on the Doom environment. Comprehensive architecture and hyperparameters search was not conducted, as our goal was comparing data representations and their impact on Reinforcement Learning.

We experimented with a wide set of algorithm hyperparameters. Listed in Section 7.0.3 are some that yielded good results for the DQN, as well as Section 7.0.3 contains similar list for PPO. Reward plots for the PPO algorithm can be found in Figure 8.

Hyperparameter	**Value**
epochs	100
steps per epoch	500
frame skip	4
batch size	100
learing rate	0.00025

Table 1: DQN hyperparameters

Hyperparameter	**Value**
environment steps	10^6
num steps	128
num minibatch	16
clip param	0.1
value loss coef	0.5
entropy coef	1
learning rate	0.00025

Table 2: PPO hyperparameters

Patches Count	**Scenario**	**Average Word Count**
3	basic	23.1 ± 1.9
5	basic	23.7 ± 1.5
11	basic	24.6 ± 3.8
21	basic	26.7 ± 1.8
3	basic light nuisance	110.5 ± 9.9
5	basic light nuisance	160.3 ± 27.7
11	basic light nuisance	215.2 ± 24.0
21	basic light nuisance	256.3 ± 24.7
3	basic heavy nuisance	176.0 ± 28.0
5	basic heavy nuisance	271.3 ± 66.3
11	basic heavy nuisance	439.9 ± 65.9
21	basic heavy nuisance	600.3 ± 74.9
3	defend the center	27.7 ± 7.8
5	defend the center	28.4 ± 9
11	defend the center	28.9 ± 9.8
21	defend the center	32.7 ± 11.1
3	defend the center light nuisance	72.5 ± 13.8
5	defend the center light nuisance	109.4 ± 12.6
11	defend the center light nuisance	161.9 ± 14.1
21	defend the center light nuisance	180.1 ± 18.3
3	defend the center heavy nuisance	112.3 ± 12.3
5	defend the center heavy nuisance	178.0 ± 22.4
11	defend the center heavy nuisance	271.5 ± 13.4
21	defend the center heavy nuisance	374.1 ± 24.5
3	super scenario	128.5 ± 31.9
5	super scenario	181.1 ± 46.6
11	super scenario	269.7 ± 105.8
21	super scenario	416.2 ± 96.7

Table 3: statistics of words per state as function of patches.

SCENARIO	LIVING REWARD	KILL REWARD	DESCRIPTION
Basic	-1	100	aim and shoot at a single target
Health Gathering	1	0	collect health packs
Take Cover	1	0	dodge incoming missiles
Defend the Center	0	1	rotate and shoot incoming enemies
Defend the Line	0	1	shoot enemies while dodging missiles
Super Scenario	0	1	mixture of all the above

Table 4: Doom scenarios

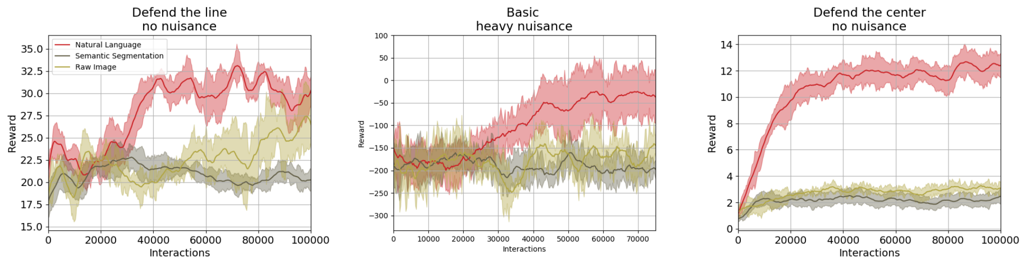


Figure 8: PPO - state representation and their average rewards, various degrees of nuisance