PLAYING ATARI WITH CAPSULE NETWORKS:
A SYSTEMATIC COMPARISON OF CNN AND CAPSNETS-BASED AGENTS.

Anonymous authors
Paper under double-blind review

ABSTRACT

In recent years, Capsule Networks (CapsNets) have achieved promising results in tasks in the object recognition task thanks to their invariance characteristics towards pose and lighting. They have been proposed as an alternative to relational insensitive and translation invariant Convolutional Neural Networks (CNN). It has been empirically proven that CapsNets are capable of achieving competitive performance while requiring significantly fewer parameters. This is a desirable characteristic for Deep reinforcement learning which is known to be sample-inefficient during training. In this paper, we conduct a systematic analysis to explore the potential of CapsNets-based agents in the deep reinforcement learning setting. More specifically, we compare the performance of a CNN-based agent with a CapsNets-based agent in a deep Q-network using the Atari suite as the testbed of our analysis. To the best of our knowledge, this work constitutes the first CapsNets based deep reinforcement learning model to learn state-action value functions without the need of task-specific adaptation. Our results show that, in this setting, CapsNets-based architectures require 92% fewer parameters compared to their CNN-based counterparts. Moreover, despite their smaller size, the CapsNets-based agents provide significant boosts in performance (score), ranging between 10% - 77%. This is supported by our empirical results which shows that CapsNets-based agents outperform the CNN-based agent, in a Double-DQN with Prioritized experience replay setting, in eight out of the nine selected environments.

1 INTRODUCTION

In recent years, Convolutional Neural Networks (CNNs) have made breakthroughs in multiple machine learning tasks like natural language processing, computer vision (Kalchbrenner et al., 2014; Krizhevsky et al., 2017). The field of Deep Reinforcement Learning (DRL) has benefited from the remarkable flexibility of CNN based agents as well. CNNs have scalar nature. Having additive nature of neurons at any given layer, they are ambivalent to spatial relationships within their kernel of previous layers (LaLonde & Bagci, 2018). Thus despite their good performance, they have an inherent drawback where they do not consider the spatial relationships between the learned features (Sabour et al., 2017; Wen et al., 2020). For example, for the task of recognizing faces in images, CNNs are capable of learning that regions that resemble a nose or a mouth are relevant. However, when recognizing a face, at test time, they have the weakness of focusing on the occurrence of these "facial parts" and completely ignore the spatial arrangement in which these should occur in order to effectively represent a face.

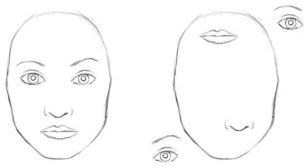


Figure 1: A CNN will classify both images as faces because of facial elements. (Pechyonkin, 2017).

Capsule Networks (CapsNets) were designed to mimic human vision (Hinton et al., 2018; Sabour et al., 2017). They address the inherent limitation of CNNs, while significantly decreasing the

required number of parameters. CapsNets aim to preserve the spatial information (pose and precise location) and attributes (length, thickness etc) by encoding features in vectors rather than scalar values. While vector represents features, the length of vector represents the probability of existence of the entity it's representing. CapsNets in deep learning require less training data, which is a desirable attribute within a DRL setting. The architectural design of CapsNets profits from *dynamic routing*. Routing by agreement is a novel dynamic routing technique, it plays a key role in preserving spatial information.

The architectural overview of capsules draws inspirations from the Multi-Layer Perceptron architecture. This architecture with *routing by agreement* is designed to preserve part-whole relationships (locations, orientations, etc.) between various entities which may be an entity or parts of an entity. For example, the relative positions of a nose and a mouth on a face in a portrait Fig 1. Reinforcement learning approaches such as DQN strive to estimate the action-value function (Mnih et al., 2015; 2013). Traditionally for vision-based tasks, the architecture of an agent uses CNNs and fully connected layers to approximate the optimal action-value function. The CNN based architecture of the agent in various deep reinforcement learning algorithms Mnih et al. (2015); van Hasselt et al. (2015); Schaul et al. (2016) are inspired from Hubel & Wiesel (1963). The agent learns on raw sensory input that uses CNNs to mimic the effects of receptive fields(Mnih et al., 2015). Sabour et al. (2017) used length of a vector from last layer of CapsNets for classification in supervised deep learning. However the length of a vector is not a good candidate for estimating the state-action value function. Here we propose an architecture suitable for an agent inspired on CapsNets. We draw the comparison between CNN-based and CapsNets based agents We conduct comparison of CNNs with CapsNets in selective environments of Atari, which offers multiple diverse tasks. Across multiple environments, the proposed agent uses 92% fewer number of parameters and improves 10%-77% on performance (score) compared to a CNN-based agent.

The main contributions of this paper are:

1. Presenting an empirical study of the difference in performance between CapsNets and standard CNNs in a DRL setting.
2. Introducing a generalized CapsNets-based architecture in DRL, without task-specific adaptation.

2 Related work

On account of the drawbacks of CNNs, Sabour et al. (2017) introduced the idea of CapsNets, but most of the published research on CapsNets is currently focused in the field of deep learning. Bahadori (2018); Phaye et al. (2018); Rawlinson et al. (2018) extend the work of Sabour et al. (2017) to propose new capsule-based architectures. Afshar et al. (2019); Allioui et al. (2019) and LaLonde & Bagci (2018) investigate the performance of CapsNets in medical applications like brain tumour classification, Alzheimer disease detection and Lung segmentation (Setio et al., 2017; Armato et al., 2011; Clark et al., 2013).

While CapsNets have gained popularity in standard Deep learning approaches, their study within a Deep Reinforcement Learning (DRL) context has received significantly less attention. Andersen (2018) tries integrating CapsNets with Deep-Q Learning. They showed that CapsNets based agent underperform with respect to CNNs-based agent. The architecture takes 84x84 input which propagates to output nx16 vector from last capsule layer. **n** being number of actions. The architecture proposed by Andersen (2018), does not take into consideration that vector output from a capsule is not a good fit for action-value estimation. While the value function could have any negative or positive value, the length of the vector output from CapsNets is bounded between zero and one.

Molnar & Culurciello (2020) combines CapsNets with A2C. The study focuses only on maze navigating in the ViZDoom environment. While using a fewer number of parameters the study shows that CapsNets-based agents provide a capable design for a policy function for navigational scenarios. The work also confirms the CapsNets ability to maintain better spatial relationships in varied textured rooms of the environment.

CNNs have proved themselves exceptionally well being part of DRL agents (Mnih et al., 2016; 2015; 2013; van Hasselt et al., 2015; Wang et al., 2016). We draw inspiration from CNN based

architecture to use CapsNets to learn representations for an agent. Inspired from Mnih et al. (2013) we propose a generalised framework for CapsNets-based agent to learn state-action value function with no task specific adjustments.

3 BACKGROUND

3.1 CAPSULE NETWORKS

CapsNets are groups of neurons Fig. 3 capable of learning spatial relations between simple and complex entities (Sabour et al., 2017; Hinton et al., 2018). They encode an object as a vector where its magnitude represents the probability of object occurrence and its orientation represents attributes of the object.

Computer graphics employ *Hierarchical Modeling* for the building complex objects by putting in simpler objects and their known relations (Eck, 2016). The idea of CapsNets is to achieve the capabilities of inverse hierarchical modelling to better understand the scene. Thus a capsule forms a wrapper around a set of neurons, Fig. 2 shows the relation between both ideas. While a neuron receives a scalar value as input and produces a scalar output, a capsule computes vector from input vectors. At this point, the length of the vector can be used to represent the probability that an object exists. We arrange capsules in 2 levels, in lower level *l* they are called primary capsules and upper-level *l+1* they are called secondary capsules.

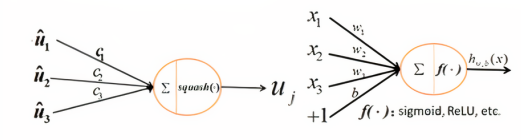


Figure 2: Similarity between a capsule and a neuron (Liao, 2018).

Primary capsules Following the first convolutional layer, the primary capsule (*PrimaryCaps*) is responsible to transform scalar values into a vector. A capsule in the architecture 3 refers to a group of convolution layers. It is the first layer where the process of inverse hierarchical modelling takes place. Capsule here reshapes the feature maps outputs of convolutional layers to output vectors.

Secondary Capsules Following PrimaryCaps is Secondary Capsules (*SecondaryCaps*). They receive an input vector from PrimaryCaps. The weight matrix $\mathbf{W}_{ij}$ transforms output vector of PrimaryCaps to serve as input to SecondaryCaps.

$$\hat{u}_{j|i} = W_{ij} u_i$$

Routing by agreement Routing by agreement is a dynamic routing technique introduced in Sabour et al. (2017). Pooling operation communicates important information to the following layer. Contrary to statically connected pooling layer, the dynamic routing happens during the forward pass. It redirects the output from PrimaryCaps to the most relevant parent in SecondaryCaps. Each capsule *i* (where $1 \leq i \leq N$) in a layer *l* has vector u_i to encode spatial information. The output of PrimaryCaps u_i of ith acts as input to all capsules in layer *l+1* of SecondaryCaps.

Coupling coefficient c_{ij} is iterative determined through routing by agreement. It represents the agreement of a capsule of layer *l* with *l+1*. If the agreement is high, the coupling coefficient for child-parent will increase, otherwise it would decrease. Coupling coefficient plays a role in child-parent relationship to form a parse tree-like structure in CapsNets. The weighted sum (s_j) from all PrimaryCaps contributes to forming the output of SecondaryCaps.

$$s_j = \sum_{i=1}^{N} c_{ij} \hat{u}_{j|i}$$

The length of the output vector from PrimaryCaps is limited between 0 and 1. We employ *squashing function* for it. The length of the vector represents the probability of the existence of an entity represented by a capsule.

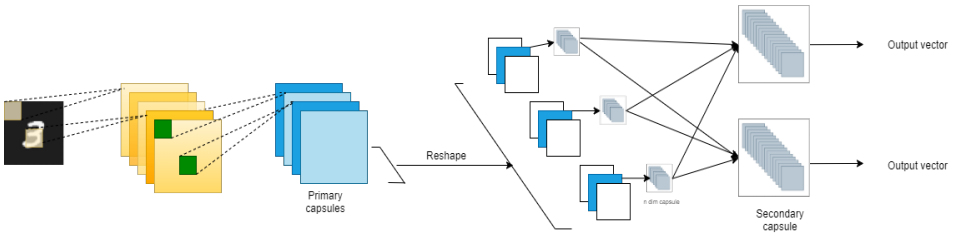


Figure 3: The figure shows fundamental Capsule network architecture.

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$$

The squashing function makes sure to limit the length while still retaining the positional information.

3.2 Deep Reinforcement Learning

The paper studies the utility of CapsNets based representations in Double DQN using prioritised experience replay. it uses *proportional prioritization* of prioritised experience replay.

The Q-learning algorithm is a temporal difference learning algorithm. To update the action-state estimate, TD error is computed at each time step. The Deep Q-learning was first introduced by Mnih et al. (2013) to approximate Q-values for high dimensional sensory input. The Deep Q-learning is known to be unstable and it overestimates the Q-values. To remedy it van Hasselt et al. (2015) proposed Double DQN. They decoupled the networks for selecting and evaluating an action separately. The agent generally selects an action using ϵ-greedy policy. Under ϵ-greedy policy, agents can take a random action with ϵ probability or it selects an action with 1-ϵ based on the value of input state that may lead to a maximum reward.

Experience replay is used to store agent's interaction with environment at each time-step (Mnih et al., 2013). This buffer is used for sampling a batch of experience to train an agent. Schaul et al. (2016) provided a new experience replay design where the most important experiences were replayed to the agent. The importance or priority of experience was calculated using TD error. With the design choice, Schaul et al. (2016) were able to empirically show that experience replay became more efficient and effective, which led to even better and faster learning of an agent. The agent performed better compared to the previous state of the art DQN.

4 Methodology

In this section we describe the several components that constitute our analysis. More specifically, we introduce the two agents to compared and the environment used as testbed for the analysis.

CNN-based Agent For the baseline we choose Double-DQN with prioiritised experience replay (Schaul et al., 2016; van Hasselt et al., 2015). The first alyer in this architecture is a convolutional layer composed by 32, 8x8 convolution kernels with a stride of 4. This first feeds a second convolutional layer of 64, 4x4 kernels with a stride of 2. The third layer receives input from second and has 64, 3x3 kernels with a stride of 1. The last convolutional layer of this set is connected to two FC layers. The first FC layer is composed by 512 neurons while the second FC layer is composed by a number of neurons equal to the output value estimates for the actions of interest. ReLU acts as activation function for all the layers except the last FC layer. The architectural design of the CapsNets based agent is depicted in Fig. 4 (bottom).

CapsNet-based Agent In a DRL agent, CNNs learn relevant visual features with respect to the task at hand while the FC layers aim at learning valuable combinations of these features and map them to value functions related to the actions of interest. In this regard, the FC layers learn the value function based on the features generated by CNNs. We explore the application and utility of CapsNets-based

representations with Double DQN. The architectural design of the CapsNets based agent is depicted in Fig. 4 (top).

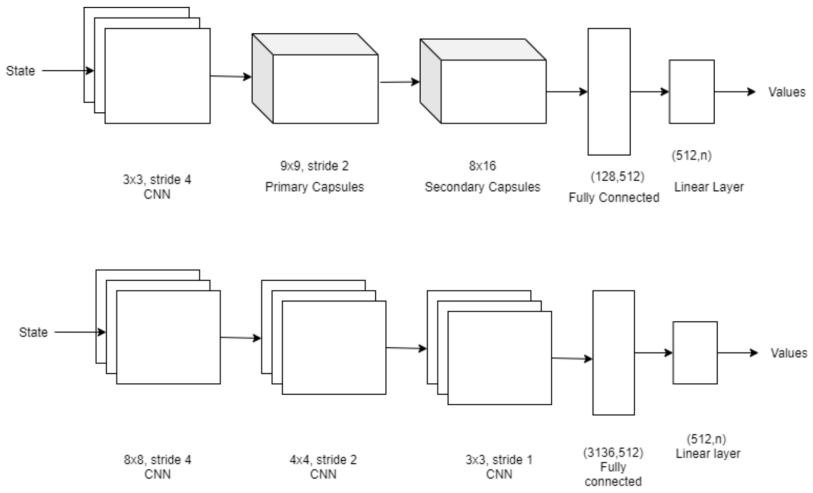


Figure 4: CNN (top) and CapsNets (bottom) based agent's architecture.

Convolutional layer acts as the first layer, as shown in Fig.4. The Convolution layer has 16, 3x3 convolution kernels with a stride of 4 and ReLU activation. This layer detects features from states and serves as an input to the Primary capsule layer. We have 49 capsules in the Primary capsule layer. A Primary capsule layer, here is a collection of convolution capsules. A single convolution capsule comprises of a group of convolution layers with 9x9 kernel and with a stride of 2. Each capsule in the PrimaryCaps receives the input of all Convolution layers. Each primary capsule outputs an 8-dimensional vector. The output from the Primary capsule serves as input to the Secondary capsule layer. Secondary capsule layer has 8 capsules with each Secondary capsule producing a 16-dimensional vector as output. Each of the Secondary capsules receives the input from all Primary capsules. The connection between the PrimaryCaps layer and the SecondaryCaps is controlled by *dynamic routing*. In our study, we use the *routing by agreement* algorithm (Sabour et al., 2017) where each child chooses its parent based on the cosine similarity between its transformed vector output and the vector output of its candidate parent. The dynamic routing between layers utilizes the vector output from capsules to preserve hierarchical relations in a state. Three routing iterations are used between capsule layers in order to find optimal weights for relations between layers.

Environment The Arcade Learning Environment (ALE) (Bellemare et al., 2013) is a popular benchmark composed by a collection of Atari 2600 games . It provides a challenging and diverse set of tasks with respect to visual input, rewards returned by the environment, action space and difficulty. Martnez-Plumed & Hernandez-Orallo (2017) integrate around 40 techniques from a dozen papers in order to determine the difficulty level of the games that are part of the benchmark. To compare the performance of our CapsNets-based agent with respect to the CNN-based agent, we choose a subset of the environment that is diversified in terms of visual input (simple, complex), reward (sparse, dense), action space(3, 4, 6, 8, 18) and difficulty score (Martnez-Plumed & Hernandez-Orallo, 2017). Across various tasks, both agents are asked to collect the maximum reward. The environment gets reset the moment when the agents use all of their lives.

The input states are composed by simple states such as Pong, Boxing to fairly complex input states like Fishing Derby or Alien. Further, the tasks are diversified with respect to rewards collected by the agent. The agents are evaluated with dense rewards environments like Breakout, Pong and sparse reward like Fishing Derby.

Training protocol With Atari we restrict the training of both CNN based and CapsNets based agents to only 20 million steps. The CapsNets based agent uses a batch size of 128 and a Learning rate of 0.00015 with RMSprop optimizer and Prioritised experience replay with alpha = 0.5 and beta with linear annealing from 0.4 to 1. The other hyper-parameters such as discount rate, the size of the experience replay memory, target network updates is the same as Schaul et al. (2016). CNN

Table 1: Performance comparison

ENVIRONMENT			SCORE			PARAMETERS		
NAME	DIFFICULTY	ACTION	CAPSNETS	CNN	PERFORMANCE	CAPSNETS	CNN	DIFFERENCE
Alien	-	18	**1678.20**	1503.79	11.60%	**136,426**	1,693,362	91.94%
Boxing	-2.11368712	18	**92.87**	58.74	58.10%	**136,426**	1,693,362	91.94%
Breakout	-0.44196066	4	**259.4**	191.1	35.74%	**129,244**	1,686,180	92.33%
Fishing Derby	1.28989165	18	**-11.99**	-27.19	55.90%	**136,426**	1,693,362	91.94%
Pong	-0.04440702	3	**20.15**	18.25	10.41%	**130,270**	1,687,206	92.27%
Qbert	1.39864132	6	**9942.95**	5616.26	77.03%	**129,244**	1,686,180	92.33%
Space Invaders	0.16420283	6	787.64	**924.11**	-14.76%	**136,426**	1,693,362	91.94%
Tennis	10.48605210	18	**-7.138**	-23.645	69.79%	**130,270**	1,687,206	92.27%
Tutankham	1.98175005	8	**148.75**	129.20	15.13%	**131,296**	1,688,232	92.22%

based agent trains on hyper-parameters as discussed by Schaul et al. (2016). Epsilon-greedy action selection method is employed to balance our exploration and exploitation. Both CNN based and CapsNets based agents randomly explore for first 50000 steps and then linearly decrease the probability to randomly select an action for next 1e6 steps. At end of 20 million steps, there still remains an exploration probability of 0.01. Evaluation section compares the cumulative reward collected by agents in all tasks. The average is calculated from 4 randomly initialized agents.

5 EVALUATION

In any given task an agent collects rewards to maximize its performance. Cumulative reward collected by an agent is the attribute that links to the agent's success in a given task. Apart from the cumulative rewards, to better understand the CapsNets based representation in DRL environments, we try to understand about agents performance under different attributes of the environments like input states, rewards and action space.

5.1 CUMULATIVE REWARD AND PARAMETERS

The CapsNets-based agent proposed in Sec 4 has 91%-92% lower number of trainable parameters. To highlight the difference, Table 1 presents a comparison of trainable parameters of both agents under different environments. To show the effectiveness of the representations learned via CapsNets, we compare the agents' performance with respect to the cumulative reward collected by them in all of analyzed tasks. Table 1 presents the comparison of performance of both agents. Though CapsNets-based agents have a lower number of training parameters, they outperform their CNN-based counterparts in all selected environments except SpaceInvaders.

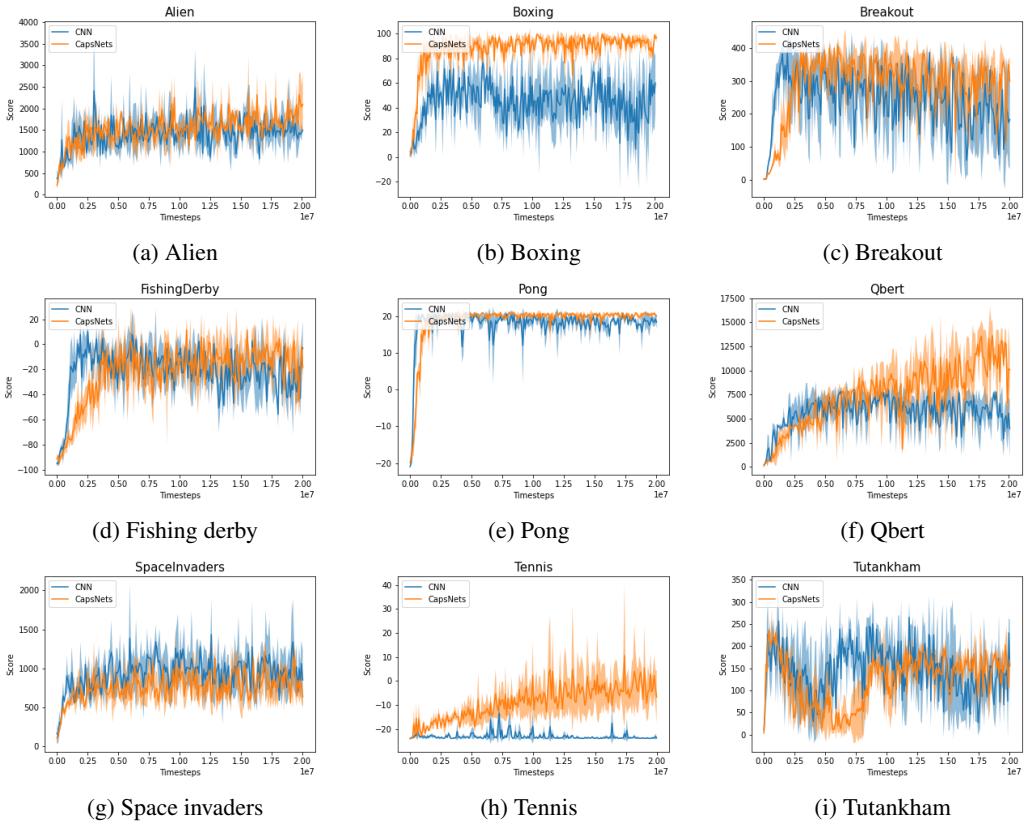
5.2 INPUT STATE

In this section, we reason how the input state of an environment (Fig. 6) is an influencing factor for CapsNets based agent. CapsNets architecture focuses on recognising simple and complex entities. Pong, Boxing, Tennis are one of the visually simple environments referring to them as *V1*. Breakout and QBert are more complex than V1, referred to as *V2*. But V2 is simpler compared to Alien, SpaceInvaders, Tutankham and Fishing Derby of *V3*.

It is observable that in simpler input state of V1, a CapsNets based agent performs excellent. The performance could be highly attributed to the very simple input state. In these environments, there are clear separate entities such as players, ball in the input state. The CapsNets based agent's learning curve is swifter compared to baseline CNNs. With comparably complex V2, the convergence of CapsNets based agent is slower yet they outperform CNN based baseline as well. With added visual complexities and an increase in the number of observable objects, we can observe that convergence slows down further. The same can be concluded for V3.

5.3 ACTION SPACE

The atari suite provides a variety of environments with respect to action space as well. The action space is an important part of an environment since it is directly related to the number of actions available for the agent. A higher action space expresses higher degree of freedom for an agent to

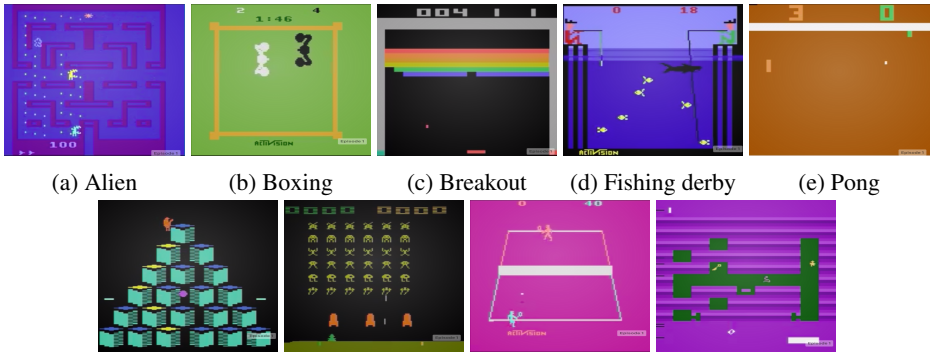


(a) Alien (b) Boxing (c) Breakout

(d) Fishing derby (e) Pong (f) Qbert

(g) Space invaders (h) Tennis (i) Tutankham

Figure 5: Average score collected by the agents in the respective environment. The agents follow an epsilon greedy policy.



(a) Alien (b) Boxing (c) Breakout (d) Fishing derby (e) Pong

(f) Qbert (g) Space invaders (h) Tennis (i) Tutankham

Figure 6: State input of various Atari environments.

choose an action from. For our study we started with low action space of 3 and 4, in Pong and Breakout, respectively. From there, we go to the highest action space available in atari, i.e 18, in Alien, Boxing, Fishing Derby and Tennis. As can be noticed in Table 1, apart from the expected increase in the number of parameters introduced by the fully connected layers, there does not seem to be a direct correlation between an agent's performance and the action space.

5.4 Reward

The agent after interactions with environment gets reward signal and next state. With the goal of maximizing the cumulative rewards, the reward as part of environment governs how an agent comprehends the input state. The environments can broadly be classified into Dense rewards (Alien,

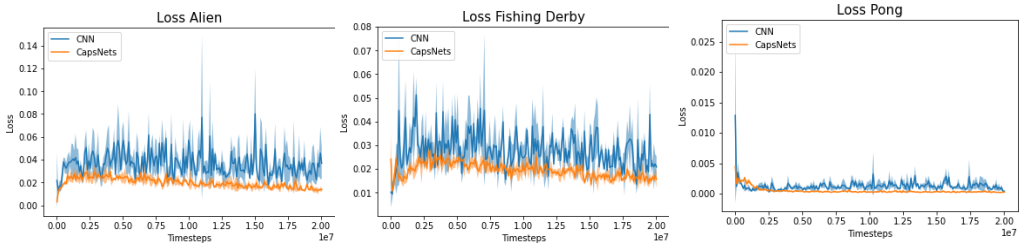


Figure 7: Training loss in the Alien (left), Fishing Derby (center) and Pong (right) environments.

Qbert etc) or Sparse rewards (Montezuma's revenge) environments. For our investigation, we diversify our environments with some dense reward environments such as Alien and some marginally sparse environments such as Fishing derby. DQN suffers from poor sample efficiency when rewards are very sparse in an environment (Gou & Liu, 2019). There is a relation between reward density and convergence of an agent to a value function. In Dense reward environment Alien it takes around 3 million steps for a CapsNets based agent to outperform CNNs while in Fishing Derby, it takes around 13 million.

While we rationalize about the better performance of CapsNets based agent, there is not a single most powerful component that directly contributes to it. It's the combination of all three elements. It is noticeable the performance of the agent in environment Tennis is similar to Boxing although they both have different difficulty level. The leading performance of CapsNets based agent in both environments can be attributed to very simple visual input and high action space. If compared to the difference in the convergence of agents in Alien, which is maze traversal environment and with a highly dense reward with Tutankham which is maze traversal but comparatively sparse reward environment. We notice that the combination of reward and action space contributed more to the performance, compared to the visual input state. The human perception suffers from Crowding, The CapsNets based agent show similar phenomenon in SpaceInvaders, The low performance could be attributed to the combination of crowding and low action space, where there are multiple instances of the same part and whole objects in input state (Pelli, 2008; Sabour et al., 2017).

Also to analyze and gain insight to the potential of CapsNets based representation, we plot and compare the loss Fig. 7 of both agents while training as well. It is noticeable that loss are comparatively smaller in magnitude compared to CNNs based agents. This can be attributed to a lower change in weights. We suggest that vectored representation in CapsNets further helps in stabilizing the change in value function of Double DQN. Part or complex entity, when learned by capsules only adept their vector orientation and length with new input states. Low loss indicates that they do not start representing new entities. The low magnitude of loss in CapsNets agent supports the proposed hypothesis.

6 Conclusion

The paper introduced CapsNets-based Double DQN using Prioritised Experience Replay. We empirically shows how CapsNets based architecture performs well with Double DQN while stabilizing it. The CapsNets-based architecture uses fewer parameters while still outperforming its CNN-based counter part in terms of cumulative reward collected by an agent in a given task. In contrast to previous research, the CapsNets-based Double DQN converges to find a value function.

The presented architecture was found to be the best performing in terms of design and capabilities in the environments. The outcome confirms the initial hypothesis that value function is learned by Fully connected while CapsNets learns to better represent states. It also confirms that the spatial relationships preserved by CapsNets help in improving the performance of the agent in various tasks. Although our evaluation covered a variety of tasks and reward systems, it would be useful to investigate the performance of the agents in other tasks and within other settings as well.

REFERENCES

Parnian Afshar, Konstantinos N. Plataniotis, and Arash Mohammadi. Capsule Networks for Brain Tumor Classification based on MRI Images and Course Tumor Boundaries. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1368–1372, November 2019.

Hanane Allioui, Mohamed Sadgal, and Aziz Elfazziki. Deep MRI Segmentation: A Convolutional Method Applied to Alzheimer Disease Detection. *International Journal of Advanced Computer Science and Applications*, 10(11), 2019. ISSN 21565570, 2158107X. doi: 10.14569/IJACSA.2019.0101151.

Per-Arne Andersen. Deep Reinforcement Learning using Capsules in Advanced Game Environments. *arXiv:1801.09597 [cs, stat]*, January 2018.

Samuel G. Armato, Geoffrey McLennan, Luc Bidaut, Michael F. McNitt-Gray, Charles R. Meyer, Anthony P. Reeves, Binsheng Zhao, Denise R. Aberle, Claudia I. Henschke, Eric A. Hoffman, Ella A. Kazerooni, Heber MacMahon, Edwin J. R. van Beek, David Yankelevitz, Alberto M. Biancardi, Peyton H. Bland, Matthew S. Brown, Roger M. Engelmann, Gary E. Laderach, Daniel Max, Richard C. Pais, David P.-Y. Qing, Rachael Y. Roberts, Amanda R. Smith, Adam Starkey, Poonam Batra, Philip Caligiuri, Ali Farooqi, Gregory W. Gladish, C. Matilda Jude, Reginald F. Munden, Iva Petkovska, Leslie E. Quint, Lawrence H. Schwartz, Baskaran Sundaram, Lori E. Dodd, Charles Fenimore, David Gur, Nicholas Petrick, John Freymann, Justin Kirby, Brian Hughes, Alessi Vande Casteele, Sangeeta Gupte, Maha Sallam, Michael D. Heath, Michael H. Kuhn, Ekta Dharaiya, Richard Burns, David S. Fryd, Marcos Salganicoff, Vikram Anand, Uri Shreter, Stephen Vastagh, Barbara Y. Croft, and Laurence P. Clarke. The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A Completed Reference Database of Lung Nodules on CT Scans: The LIDC/IDRI thoracic CT database of lung nodules. *Medical Physics*, 38(2):915–931, January 2011. ISSN 00942405. doi: 10.1118/1.3528204.

Mohammad Taha Bahadori. Spectral Capsule Networks. pp. 5, 2018. URL `https://openreview.net/forum?id=HJuMvYPaM`.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912.

Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository. *Journal of Digital Imaging*, 26(6):1045–1057, December 2013. ISSN 0897-1889, 1618-727X. doi: 10.1007/s10278-013-9622-7.

David J Eck. *Introduction to Computer Graphics*. David J. Eck, 2016.

Stephen Zhen Gou and Yuyang Liu. DQN with model-based exploration: Efficient learning on environments with sparse rewards. *arXiv:1903.09295 [cs, stat]*, March 2019. URL `https://arxiv.org/abs/1903.09295`.

Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJWLfGWRb`.

D. H. Hubel and T. N. Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of Physiology*, 165(3):559–568, March 1963. ISSN 00223751. doi: 10.1113/jphysiol.1963.sp007079.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 655–665, Baltimore, Maryland, 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1062.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. ISSN 00010782. doi: 10.1145/3065386.

Rodney LaLonde and Ulas Bagci. Capsules for Object Segmentation. *arXiv:1804.04241 [cs, stat]*, April 2018.

Huadong Liao. Capsnet-tensorflow, 2018. URL https://github.com/naturomics/CapsNet-Tensorflow/blob/master/imgs/capsuleVSneuron.png.

Fernando Martnez-Plumed and Jose Hernandez-Orallo. AI results for the Atari 2600 games: Difficulty and discrimination using IRT. In *Evaluating General-Purpose AI*, pp. 6, 2017. URL http://dmip.webs.upv.es/EGPAI2017/Papers/EGPAI_2017_paper_7_FMartinez-Plumed.pdf.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, December 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/mniha16.html.

Thomas Molnar and Eugenio Culurciello. Capsule Network Performance with Autonomous Navigation. *International Journal of Artificial Intelligence & Applications*, 11(1):1–15, January 2020. ISSN 09762191. doi: 10.5121/ijaia.2020.11101.

Max Pechyonkin. Understanding hinton's capsule networks. part i: Intuition. *Deep Learning, Nov*, 3, 2017.

Denis G Pelli. Crowding: A cortical constraint on object recognition. *Current Opinion in Neurobiology*, 18(4):445–451, August 2008. ISSN 09594388. doi: 10.1016/j.conb.2008.09.008.

Sai Samarth R. Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti Bathula. Dense and Diverse Capsule Networks: Making the Capsules Learn Better. *arXiv:1805.04001 [cs]*, May 2018.

David Rawlinson, Abdelrahman Ahmed, and Gideon Kowadlo. Sparse Unsupervised Capsules Generalize Better. *arXiv:1804.06094 [cs]*, April 2018.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3856–3866. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun (eds.), *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/1511.05952.

Arnaud Arindra Adiyoso Setio, Alberto Traverso, Thomas de Bel, Moira S.N. Berens, Cas van den Bogaard, Piergiorgio Cerello, Hao Chen, Qi Dou, Maria Evelina Fantacci, Bram Geurts, Robbert van der Gugten, Pheng Ann Heng, Bart Jansen, Michael M.J. de Kaste, Valentin Kotov, Jack Yu-Hung Lin, Jeroen T.M.C. Manders, Alexander Sóñora-Mengana, Juan Carlos García-Naranjo,

Evgenia Papavasileiou, Mathias Prokop, Marco Saletta, Cornelia M Schaefer-Prokop, Ernst T. Scholten, Luuk Scholten, Miranda M. Snoeren, Ernesto Lopez Torres, Jef Vandemeulebroucke, Nicole Walasek, Guido C.A. Zuidhof, Bram van Ginneken, and Colin Jacobs. Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge. *Medical Image Analysis*, 42:1–13, December 2017. ISSN 13618415. doi: 10.1016/j.media.2017.06.015.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. *arXiv:1509.06461 [cs]*, December 2015.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. ICML'16, pp. 1995–2003. JMLR.org, 2016.

Xin Wen, Zhizhong Han, Xinhai Liu, and Yu-Shen Liu. Point2spatialcapsule: Aggregating features and spatial relationships of local regions on point clouds using spatial-aware capsules. *IEEE Transactions on Image Processing*, 29:8855–8869, 2020.