Self-Supervised Learning for Multi-Goal Grid World: Comparing Leela and Deep Q Network

Steve Kommrusch STEVEKO@CS.COLOSTATE.EDU
Colorado State University

Henry Minsky HQM@LEELA-AI.COM
Milan Minsky MILAN@LEELA-AI.COM
Cyrus Shaoul CYRUS@LEELA-AI.COM
Leela AI

Editors: Minsky, H. and Robertson, P. and Georgeon, O. L. and Minsky, M. and Shaoul, C.

Abstract

Modern machine learning research has explored numerous approaches to solving reinforcement learning with multiple goals and sparse rewards as well as learning correct actions from a small number of exploratory samples. We explore the ability of a self-supervised system which automatically creates and tests symbolic hypotheses about the world to address these same issues. Leela is a system which builds an understanding of the world using constructivist artificial intelligence. For our study, we create an $N * N$ grid world with goals related to proprioceptive or visual positions for exploration. We compare Leela to a DQN which includes hindsight for improving multigoal learning with sparse rewards. Our results show that Leela is able to learn to solve multigoal problems in an $N * N$ world with approximately $160N^2$ exploratory steps compared to $360N^{2.7}$ steps required by the DQN.

Keywords: Multi-goal, constructivist, deep Q network, machine learning, artificial intelligence

1. Introduction

One goal of artificial intelligence research is to build systems which can integrate concepts of the world and takes actions to reach goals within that world. In February 2020 at the Turing Award Winners event for AAAI-20, Yann LeCun identified 3 key challenges for AI today: learning with fewer labeled examples and/or fewer trials; learning to reason (that is, rational and logical thinking); and learning to plan complex actions Bengio et al. (2020). Hence, the challenge of integrating data about the world from a small amount of training information is recognized as crucial to improving AI in the future.

We introduce Leela, which is a symbolic learning system capable of forming hypotheses about the world and testing them as actions are performed. As it grows its knowledge using self-supervised exploratory actions, it can be given goals which it can achieve using its learned knowledge. The foundations for Leela were initially laid out decades ago by the child development theories of Jean Piaget Piaget (1964) and the computational models of Gary Drescher Drescher (1991). Leela builds models of the world using 'schemas', which

allow it to reason about which actions are possible in the current world state, and what aspects of the world change when an action is performed.

We evaluate Leela by comparing its ability to learn about an $N * N$ grid world with 3 different goal categories (hand position, eye position, and visual field). Each category allows for $N * N$ possible subgoals, providing for a total of $3 * N * N$ possible goals to be specified to the learning system. We show that the self-supervised approach used by Leela is able to learn to solve goals with fewer steps, even as the problem size scales, and to learn with less wall clock time as the number of goals scales.

Our key contributions are:

1. We introduce Leela in detail including the schema mechanism and goal solution methods.
2. We demonstrate that Leela can learn more efficiently from exploratory actions than a modern Deep Q Network even when improvement to the network is made to handle the problem of sparse rewords and enhance its learning per action step.
3. We demonstrate that Leela trains to solve multiple goals in parallel in less time than a Deep Q Network.
4. We demonstrate that a self-supervised system which explores and finds patterns can learn efficient and actionable information about the world in which it acts.

2. Background

Jean Piaget proposed a theory of childhood cognitive development in which a child learns about objects in the world through sensorimotor experience related to moving its hands and visually perceiving the world Piaget (1964). As the child grows it learns more complex concepts such as object permanence. His ideas included the concept of schemas to organize when an action (like picking up an object) is applicable and what the result of the action is on the world. Piaget organized a child's learning process into 4 main stages which exemplify how knowledge gained in previous stages is used to build more complex concepts in later stages.

- Sensorimotor stage: 0-2 years
  - Simple reflexes (moving hands, eyes, etc)
  - Primary circular reactions: coordination of 2 types of schema: i.e. passing hand before face
  - Secondary circular reactions: actions involving external objects begin
  - Coordination of 2nd circular reactions: first proper intelligence; means and ends; goals; object permanence
  - Tertiary circular reactions: curiosity about object properties
  - Internalization of schemas: insight, creativity, use primitive symbols
- Pre-operational stage: 2-7 years

  - Child can form stable concepts and magical beliefs; cannot yet mentally manipulate information; increased play

- Concrete operational stage: 7-11 years
  - Child can think logically; understand reversibility; can see viewpoints of others; improved classification skills

- Formal operational stage: 11-16+ years
  - Development of abstract reasoning; utilize metacognition; multistep problem solving

Gary Drescher proposed a system to bring Piaget's concepts into the field of computing and artificial intelligence Drescher (1991). The basis for Drescher's approach is a software model for the schema concept introduced by Piaget, shown in Figure 1. The model used for learning is that of a robot with a hand and eye which can act on and observe the world. Initially, the learning system has no understanding of how actions affect its sensory input. For example, a schema might be learned that when the world context includes the hand on the left side of the visual field, then the action of moving the hand to the right has the result that the hand is seen in the middle of the visual field. Leela has been trained in environments with obstacles and blocks which can be grasped and moved, but in order to more directly study the learning capabilities relative to a Deep Q Network, the environment studied in this paper only involves hand-eye coordination learning. Readers interested in details of the schema proposal and learning mechanism are encouraged to read Drescher's book: Made Up Minds Drescher (1991), but we will provide a brief summary here.

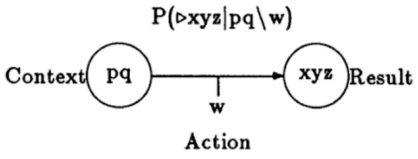


Figure 1: A schema.

Figure 1 shows a schema with items p and q in its context, action w, and items x, y, z in its result. A schema also maintains some auxiliary data, such as the schema's reliability— that is, the reliability with which the predicted result will actually follow the schema's action (provided that the context is satisfied). Reliability is measured by recording:

- **P(▷R|CA), the conditional probability of a transition to the result state (R) given context conditions (C) and action (A).**
- **P(▷R|$C\neg A$), the conditional probability of a transition to the result under the same conditions except without the action.**

Figure 1: Schema diagram showing context, action, and result Drescher (1986).

As Leela builds schemas that represent knowledge about the world, goals can be achieved by a planner which chains actions together from actions which are possible in the current

world context to a final action which has a result that meets the goal. This is done by finding a series of actions whose results are in the context of a following action until the results match the goal.

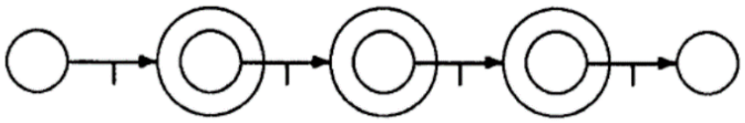


Figure 2: Schemas can be chained together to reach a goal during planning Drescher (1986).

Like neural networks, the foundational model for the Leela approach to learning was initially proposed decades ago; and like neural networks, the software ideas proposed benefit from the faster hardware available today. As we will demonstrate, Leela's learning models based on schemas can help address some of the key challenges facing AI today.

3. Methodology

Our experiments compare Leela to a DQN model in an $N * N$ grid world. Both Leela and the DQN receive as inputs the proprioceptive hand and eye positions, as well as the image of the visual field, as shown in Figure 3. Proprioception is the term for the body's sense for where body parts are. For these experiments, there are no objects in the world besides the hand and eye, but this still allows for exploring the ability to learn hand-eye coordination. For both Leela and the DQN, there are 3 goal categories available to the model: move the hand to proprioceptive position x,y; move the eye to proprioceptive position x,y; or move the hand and/or eye such that the hand is seen at position x,y in the visual field. Given an $N * N$ grid, there are $3 * N * N$ possible goals which can be presented to the model for solution. There are 8 possible actions that can be taken in pursuit of the goal: move hand forward, backward, left, right, and move eye forward, backward, left, or right.

3.1. Leela model

Leela learns about the world by taking actions (which can be random, biased to explore unknown areas, or in pursuit of a given goal). Leela gathers statistics on the effects of an action given the world state when the action was taken and spawns of schemas representing what was learned. To achieve a goal, Leela tries to chain schemas together to reach a goal state from the current state. In a richer environment, Leela is able to learn actions in a grid world which includes objects in it, and inputs may include vision details (such as color and shape) and a sense of touch. For this comparison with DQN, objects, vision details, and the sense of touch are not a part of the world. To explore environment scaling, we scale the size of the grid world from 5x5 to 15x15. To explore multi-goal behavior, we train models to solve 1, 2, or 3 goal categories simultaneously.

Leela learns from actions by forming schemas which can be thought of as hypotheses about the world. For example, Leela might take the 'move hand left' action and notice

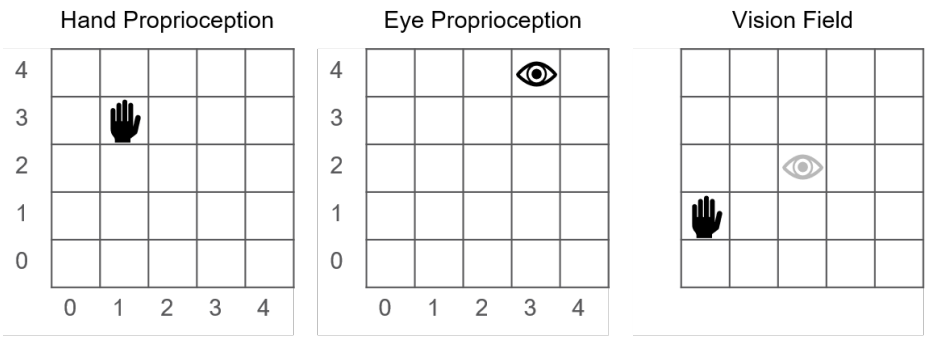


Figure 3: Inputs to Leela and DQN include the hand and eye proprioceptive position, and the visual field. The position of the eye determines the center of the visual field, hence the location of the hand in the visual field is based on the relative position of the hand and eye.

that a result was 'the hand is now visible in the center of the visual field'. It might form a schema theorizing that moving the hand left has this result, but later takes the action and the same result does not occur. Given the new observation, Leela can add context to the schema so that it ultimately learns that moving the hand left only results in seeing the hand in the center of the visual field if the hand was originally just to the right of the center of the field.

3.2. DQN model

We compare Leela with a modern reinforcement learning algorithm. We based our model on the Deep Q Network (DQN) tutorial for PyTorch Paszke et al. (2019), which includes a replay buffer Lin (1992); Lillicrap et al. (2015), and a target network Mnih et al. (2015) to help with stability. As we will show, the base network performs poorly on the multi-goal learning problem due in part to a sparse reward, so we added hindsight experience replay Andrychowicz et al. (2017) to create a system which can be fruitfully compared to the current Leela performance.

Model inputs For the DQN model, the input is the current world state, including the desired goal, and the output is the action the model recommends taking. Figure 4 shows the full set of inputs. Just as with the Leela model, the DQN receives data about the current world state through hand and eye proprioception (a sense of where the hand and eye are in space) as well as a visual field which is the same dimension as the grid world. These 3 input forms are provided as 3 N×N input values. A fourth N×N matrix encodes the goal state:

`IN[3][0][x]`	specifies the X position of the goal
`IN[3][1][y]`	specifies the Y position of the goal
`IN[3][2:N-1][0]`	set to 1 for goal to move hand to a given proprioceptive position

`IN[3][2:N-1][1]` set to 1 for goal to move eye to a given proprioceptive position

`IN[3][2:N-1][2]` set to 1 for goal to see hand at a given position in visual field

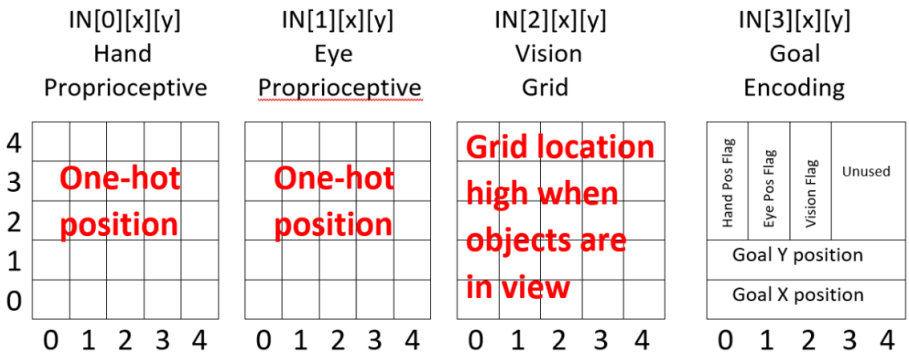


Figure 4: Input tensor has 4 2D grids. 3 grids representing sensory input, and one grid representing the goal to achieve.

The network output is the 8 possible actions: move hand forward, backward, left, right, and move eye forward, backward, left, or right.

Model training The model is trained by using back-propagation as the model optimizes an action policy given the input state Andrychowicz et al. (2017). In typical reinforcement learning, an action policy determines which action to apply in which state: $\pi(s) : \mathbb{S} \rightarrow \mathbb{A}$. Reinforcement learning works by learning a Q-function $Q(s, a)$ which represents the reward from the environment achieved by taking action a in state s. For any given state s, the action a with the highest $Q(s, a)$ value is the recommended action to take, as it achieves the highest reward. The optimal Q-function $Q^*(s, a)$ is given by the Bellman equation, shown in Equation (1). In this equation, $Q^*(s, a)$ is the expected value over all possible next states of the immediate reward $r(s, a)$ achieved by taking action a in state s plus the best Q-value of the possible future states discounted by a depreciation factor λ.

$$Q^*(s, a) = \mathbb{E}_{s' \sim p(\cdot|s,a)}[r(s, a) + \gamma \max_{a' \in A} Q^*(s', a')] \tag{1}$$

From the Bellman equation, we can see that having frequent non-zero rewards will help a system learn the correct Q-function and, hence, the correct action to take in a given state. But our goal in this paper is to model a system which is only rewarded when the goal is achieved, which models situations in which a distance to goal metric for an incremental reward may be difficult to evaluate. As we will show, sparse rewards can create training convergence challenges to traditional reinforcement learning models. In order to learn with sparse rewards, the model needs to get sufficiently lucky to take the correct action when it happens to be right next to a goal, then the system can learn how to take actions that get it closer to the goal, and so on. But with enough random actions occurring, the ability to learn the Q function can become intractable. Andrychowicz et al. address the issue of

learning with sparse rewards in a multi-goal system by introducing hindsight learning. In hindsight learning, the current state and goal state are given and the system chooses an action given it's currently poorly performing Q-function. This action is not likely to have a reward, but the training can be adjusted to reward the state achieved. Hindsight works by adding a training result as if the state reached was the goal. For example, if the model is asked to achieve goal g from current state s and it takes action a resulting in state s', then a training datum is created which teaches the model that if the goal had been to reach s' from s then the correct action was a. In this way, every step taken by the model provides some teachable information.

4. Experiments

The methodology we use to compare Leela with a DQN involves training a model to solve hand, eye, and vision goals (hpxy, vpxy, or vfxy), or a training a model to solve a subset of 1 or 2 of these 3 goal categories. For evaluating goal learning, the goal target is chosen randomly from among those goals on which the model was trained. For DQN testing, the start position is chosen randomly; for Leela testing, the start position is wherever the hand and eye currently are (which becomes random as Leela improves on meeting the previous random goal). The average number of steps required by a well performing model for such goals is less than one side of the $N * N$ grid (i.e., for a 5x5 grid the average steps for random start and end positions is 3.2). For both Leela and the DQN, we limit the model to take $4 * N$ steps in an attempt to reach a goal at which point a new goal can be specified. We track and record the number of action steps required for the model to learn to reach the goal in under $2 * N$ steps, and to reach the goal in under N steps.

The test we created for Leela takes self-supervised actions in the world without a specified goal and then every 500 steps it checks to see how well Leela can solve goals. The test will choose 10 random goals and track the average number of steps needed to reach the goal. The test steps the world until a goal is reached or $4N$ steps have been taken. Since the DQN network is reporting a result that is averaged over 200 attempts, the Leela test will only claim success on averaging less than $2 * N$ steps after 2 sets of 10 goals have been below that average. The test completes after 2 sets of 10 goals have averaged below N steps.

Tests for both Leela and the Deep Q Network were run on the CPU on a system with Intel Core i7-6700HQ @2.6GHz (4 cores, 8 threads, 3.5GHz max).

4.1. Leela study

Our Leela test case starts out with no schemas about the world and begins taking random actions in pursuit of learning. Initially, when requested to reach a goal Leela will not reach it within the $4N$ step limit. After learning more about the world, Leela will reach the goal more and more consistently within the allotted limit. During early testing, Leela may be asked to reach a goal for which it cannot form any path to attempt and in such cases Leela will continue to explore and learn randomly.

4.1.1. Leela ablation

Leela has a variety of parameters that affect how schemas are created during the learning process. Like the hyperparameters of a deep neural network, adjusting these parameters can significantly affect the network performance. Table 1 summarizes data gathered when selected parameters were varied. Results are the average of 2 runs of a 9x9 grid solving all 3 goal categories (hand, eye, vision) to an average path length of less than 9 steps.

Table 1: Leela parameter variations tested on 9x9 grid with all 3 goal categories

Varied Parameter	Description	Value	Steps
context interval-scale	Scaling of confidence intervals for context p|on versus p|off estimates. Larger values spin off new schemas slower.	0.3 0.6 1.2 2.4 4.8	24,500 14,000 15,500 15,500 48,750
min-item-transitions-context-spinoff	Number of transitions required before an item is considered for context spinoff.	2 4 8 16	15,500 14,000 13,500 14,500
results interval-scale	Scaling of confidence intervals for result p|on versus p|off estimates. Larger values spin off new schemas slower.	0.001 0.005 0.01 0.02 0.10	15,000 13,750 15,500 12,250 16,750

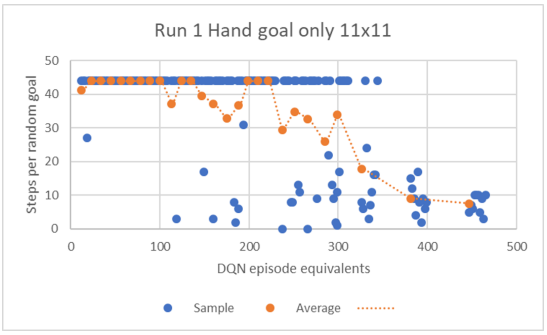
Further experiments with larger grid sizes guided the final Leela model to use 1.2 for the context interval-scale, 2 for the min-item-transitions-context-spinoff, and 0.01 for the results interval-scale. These parameters were used for the results in tables 3 and 4.

4.1.2. Leela result charts

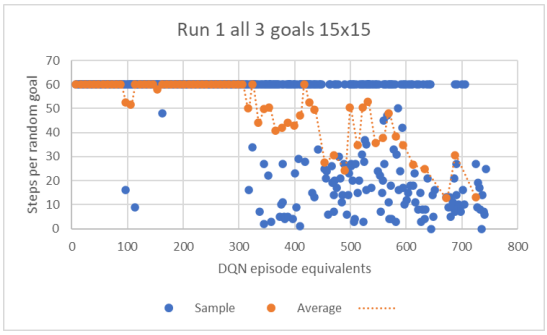
Figure 5 shows Leela performance during training. Training for Leela involves random exploration of the world resulting in schema creation. When given a specific goal, for example move hand to position 2,3, Leela will try to chain together schemas from the current state of the world to produce the goal state. Partially complete understandings of the world can still result in some successful paths when random goals are requested, and this figure show this process. At first, Leela will not understand the world enough to chain together schemas and the maximum step length of $4N$ steps will occur when trying to solve goals. As Leela understands the world, the average number of steps taken to reach a random goal drops below N in all experiments.

Figure 5(a) shows a run of Leela when being tested to solve only hand position goals in an 11x11 grid world. Figure 5(b) shows a run when Leela was tested to solve a random mix of hand, eye, and vision goals in a 15x15 grid world. The 'DQN episode equivalent' is computed to allow for comparison with DQN training charts. An episode in DQN is an attempt to reach the goal and hence the episode equivalent for Leela is a number of steps

related to the goal length (i.e., $4N$ steps at first are equal to one episode in both Leela and DQN). The blue data represents individual path attempts, which max out at $4N$ and show the typical variation during testing and training. Any blue dot at the top of the graph ($4N$) represents a failure of the model to find the path. The gold dots are the average of all 10 samples for a given test group.



(a) Single-goal learning trend



(b) Multi-goal learning trend

Figure 5: Leela training trends.

A key feature of Leela is that schemas when learned are not forgotten, so as Leela advances in learning about the world, the model does not suffer the catastrophic forgetting problem seen with DQN models Beaulieu et al. (2020).

4.2. DQN study

For the DQN analysis, we report the results during training as the model is continuously provided with goals to reach on its input. Unlike Leela, which can mix periods of exploration with goal searches, the DQN only learns by taking actions in quest of a goal which allows for a reward function to guide learning by the model.

4.2.1. DQN ablation

There are many parameters which can affect deep neural network performance, Table 2 summarizes data gathered while varying certain parameters. Results are the average of 2 runs of a 7x7 grid solving all 3 goal categories (hand, eye, vision) to an average path length of less than 7 steps.

Further experiments with larger grid sizes guided the final DQN model parameter values used for the Leela to DQN comparison studies. These values are shown in Table 3.

4.2.2. DQN result charts

Figure 6 shows the importance of using hindsight to enhance the ability of the DQN network to learn from sparse rewards. While Leela is able to use actions on the world to create and extend hypotheses, the DQN needs a reward function to learn well. The blue lines in the graphs are the path length to reach the goal each episode, the gold line is the average path length over 200 episodes. For example, when there are no blue lines reaching to the max

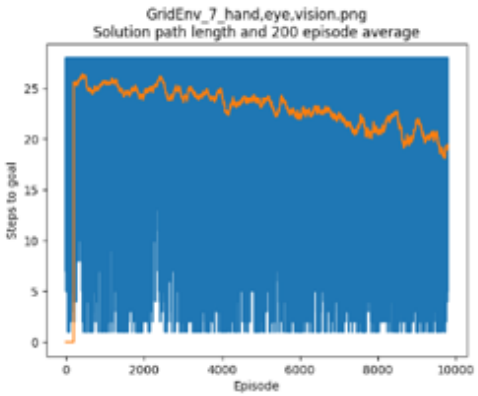
Table 2: DQN parameter variations tested on 7x7 grid with all 3 goal categories

Parameter	Description	Value	Steps
Layer width	Vary the neuron widths for a network with 3 hidden layers, 4xNxN inputs, and 8 outputs.	15N, 10N, 5N 30N, 20N, 10N 60N, 40N, 20N	71,113 52,830 67,551
Network depth	Number of hidden neural layers	2 (30N, 20N) 3 (30N,20N,10N) 4 (30N,20N,15N,10N)	54,503 52,830 58,184
ϵ_decay	ϵ is the chance to take a random action to explore instead of the predicted best action. It decays using $\epsilon = 0.05 + 0.9e^{-episode/\epsilon\_decay}$	300N 400N 500N	45,562 52,830 47,024
dropout	Test having a random weight dropout layer inserted before output layer.	No dropout 10% dropout	52,830 41,999
γ	Reward for future states decay in value by γ each step.	0.5 0.8 0.95	37,475 52,830 39,660
lr	The learning rate (lr) is used to scale the gradient when training with stochastic gradient decent (SGD).	$0.10/N^2$ $0.25/N^2$ $0.45/N^2$	65,104 52,830 Unknown
memsize	Size of replay memory used as fifo for state/action/reward/next state learning.	$750N^2$ $1000N^2$ $1500N^2$ $2000N^2$ $3000N^2$ $4000N^2$	51,520 49,807 42,004 52,830 45,740 41,474
Nonlin	Type of non-linearity used between layers.	leaky_relu relu tanh	52,830 Unknown 59,747
Reward	Reward scoring can include 0 or 1 for achieving goal, and 0 or $-1/N$ for each step taken that does not reach goal.	+1 goal, $-1/N$ step +1 goal, 0 step 0 goal, $-1/N$ step	52,830 52,700 36,745

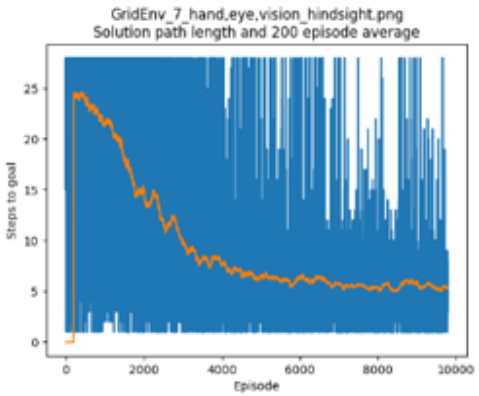
Table 3: Final DQN parameter selection

Parameter	Value
Neuron layers	3 layers with 30N, 20N, and 10N neurons each.
ϵ_decay	400N
dropout	Dropout is not used (not valuable for larger grids)
γ	0.8
lr	$0.25/N^2$
memsize	$2000N^2$
Nonlin	leaky_relu
Reward	+ 1 for achieving goal, and -1/N for each step taken that does not reach goal (works well for larger grids)

path length (4 times the length of a grid side) then for that period the network solved all the goal challenges presented. Hindsight learning, discussed in Section 3, learns 2 datum per action step - it trains on the benefit of the action taken with respect to the actual goal requested, and it trains on the benefit of the given action as if the next state reached was the goal. The gold lines in Figure 6 show a model with hindsight learns to solve multiple goals in a 7x7 grid learns faster than a model without hindsight. Without hindsight, learning the multi-goal problem on a 9x9 grid did not converge with our model.



(*a*) Multigoal sparse reward barely improving on 7x7 grid without hindsight

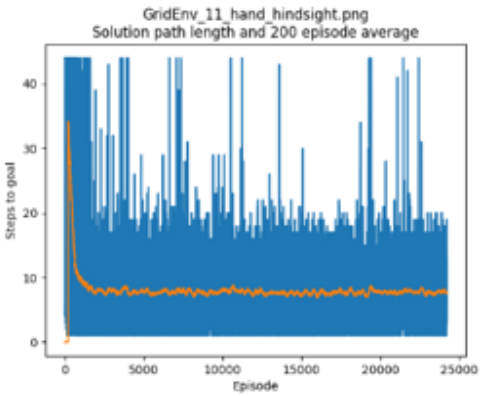


(*b*) Multigoal learning converges to an average solution below 7 steps in about 4,000 steps with hindsight on 7x7 grid

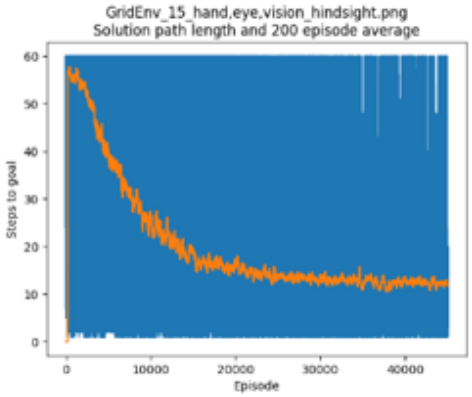
Figure 6: The benefit of hindsight. Average steps to reach a random goal shown in gold.

Figure 7 show training runs of the neural network for different goal groups. Figure 7(*a*) shows the results for attempting to reach a random hand-position goal in an 11x11 grid world as training progresses. As the system learns, the average path to reach the goal drops from the limit of 44 steps ($4N$) to the 'success' criterion of under 11 steps. IN[1][x][y] and IN[2][x][y] (the eye position and vision field) are all 0 in this case to prevent noise into the

hand position training. As shown by the per-episode data illustrated with blue lines in Figure 7(*a*), from about episode 12,000 to about 19,000 there were no failures to reach a random goal, but then a period of 'forgetting' began where the model drifted into a less successful configuration. Figure 7(*b*) shows results for a 15x15 environment which randomly selects between a hand, eye, or vision goals.



(*a*) Training for single goal in 11x11 grid shows occasional failure to reach goal after model learned



(*b*) Training for 3 goal categories in 15x15 grid shows frequent failure to reach goal after model learned

Figure 7: Charts showing DQN training and success rates.

4.3. Comparisons between Leela and DQN

Table 4 compares how quickly Leela and the DQN reach a given goal success metric as measured in action steps. The columns denote the steps required to achieve an average solution length less than 2N or less than N for both a single-goal problem as well as a multi-goal problem. The final row is a rough estimate of the scaling equation for the number of steps to reach the training goal based on the grid size. The equations are of the form cN^p when N is the length of one side of the grid, p is computed using the number of steps needed for a 5x5 world and a 15x15 world as: $p = ln(steps_{15x15}/steps_{5x5})/ln(3)$. Then c is chosen such that: $c = steps_{15x15}/15^p$. As shown in the table, Leela learns faster per step and is performing better and better per step relative to the DQN.

Table 5 and Table 6 show execution time metrics comparing Leela and DQN. Note that the DQN is implemented in the well-optimized PyTorch environment for machine learning, while the Leela code has not yet been tuned for performance. The data shows that for our problem sizes Leela outperforms the DQN in training time. As the grid size grows, however, Leela is showing a higher scale factor based on grid size. However, when goal categories are considered, we see that Leela is better able to learn about the world through exploration and scales better as new goals are added to the problem.

Figure 8 shows 4 training charts which demonstrate the ability of Leela to learn information useful for multiple goals in parallel as action are taken. Subfigures (a) and (c) show that Leela does not take significantly longer to perform well on the multigoal problem.

Table 4: Number of action steps until the system is trained to a given quality level (average of 2 runs)

Grid	Hand avg $< 2N$		Hand avg $< N$		All 3 avg $< 2N$		All 3 avg $< N$	
$N * N$	Leela	DQN	Leela	DQN	Leela	DQN	Leela	DQN
5x5	2750	3238	3000	4108	3250	20122	4250	28214
7x7	5000	4984	6500	7012	6250	40993	8250	58725
9x9	8250	11089	11500	13642	10250	90363	15500	155997
11x11	11250	15220	14500	19909	15000	150576	21500	255558
13x13	18250	24193	23750	33133	24250	250209	28750	445356
15x15	27500	45232	34000	57723	28250	350592	39500	555895
Approx steps for $N * N$	$94N^{2.1}$	$68N^{2.4}$	$86N^{2.2}$	$86N^{2.4}$	$137N^{2.0}$	$306N^{2.6}$	$162N^{2.0}$	$358N^{2.7}$

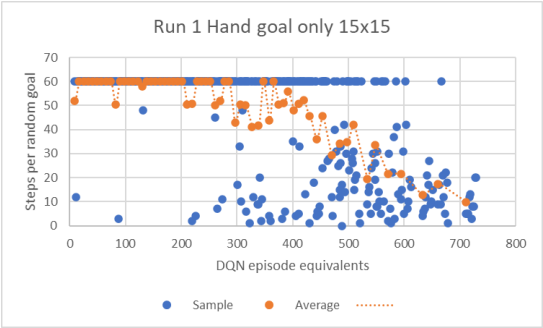
Table 5: Time in seconds until the system is trained to a given quality level (average of 2 runs)

Grid	Hand avg $< 2N$		Hand avg $< N$		All 3 avg $< 2N$		All 3 avg $< N$	
$N * N$	Leela	DQN	Leela	DQN	Leela	DQN	Leela	DQN
5x5	6	8	6	11	7	59	8	84
7x7	20	17	25	26	24	151	33	222
9x9	63	54	91	68	81	441	130	779
11x11	155	115	209	150	211	1105	333	1914
13x13	482	200	666	275	696	2048	866	3707
15x15	1330	478	1712	611	1268	3654	1911	5840
Approx time for $N * N$	$N^{4.9}/455$	$N^{3.7}/50$	$N^{5.1}/659$	$N^{3.7}/33$	$N^{4.7}/290$	$N^{3.8}/7$	$N^{5.0}/381$	$N^{3.9}/6$

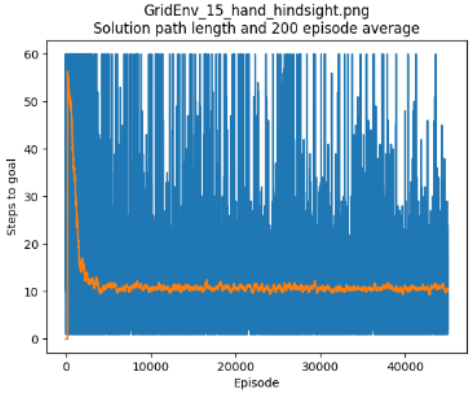
Table 6: Time in seconds until the system is trained to a given quality level on 15x15 grid (average of 2 runs)

Number of goal categories	Number of possible goals	average < 15 steps	
		Leela	DQN
1 (hand or eye)	225	1712	611
2 (hand + eye)	450	1811	2071
3 (hand, eye, vision)	675	1911	5840
Approximate time for G goal categories		$1611+100G$	$620G^2$

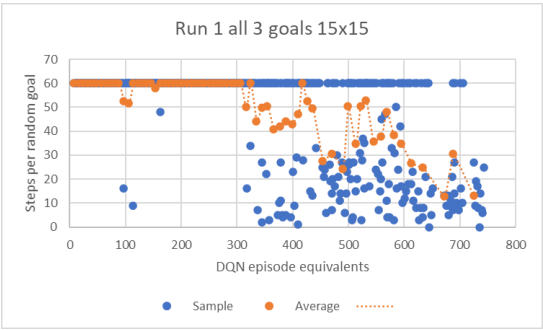
Subfigures (b) and (d) show that the DQN takes almost 10 times as many episodes to learn 3 goal categories instead of one, and, as discussed relative to Table 4, the DQN requires more exploration of the world to learn even a single goal.



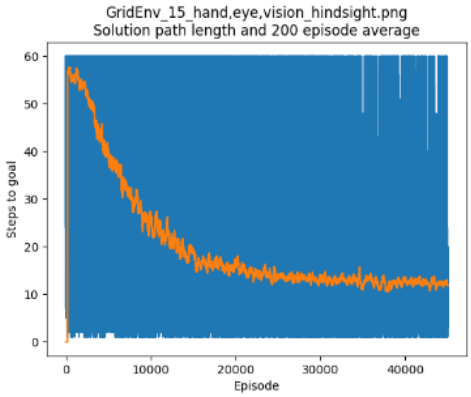
(*a*) Leela reaches training target in under 1000 DQN episode equivalents for single goal category



(*b*) Single-category DQN takes almost double the episodes needed by Leela to reach training target



(*c*) When presented with 3 goal categories, Leela solves goals almost as fast as single category test



(*d*) DQN takes almost 10 times as long to train for 3 goal categories as it did for one

Figure 8: Leela easily transfers knowledge gained allowing efficient multigoal learning.

5. Related Work

In the field of neuroscience, research by Cisek and Kalaska Cisek and Kalaska (2010) explores the mechanisms by which actions are chosen while interacting with the world. In their work, they present neurophysiolagical data in support of the idea that based on a perception of the world, various actions are presented by the frontoparietal sensory motor control system and then selected by systems in the prefrontal cortex. This mechanism has more in common with the Leela approach than the DQN approach we have presented. Leela will learn actions

which can be activated by the current world context, and these are then selected based on a goal planner when desired. The DQN combines the action generation with the goal planning by having the current world state and goal be input to a network which predicts the best action.

Survey Besold et al. (2017) introduces the general topic of combining symbolic AI with neural network models. Indeed, using the demonstrated generalization capability of a neural network with the ability of Leela to learn concepts about the world in an unsupervised way could be a fruitful path for future work. The survey paper also discussed the 'binding problem' which is the challenge of learning how to build schemas which generalize based on variables in the environment. The Leela approach could be enhanced in the future with reference to ongoing research in this area.

Baker et al. use reinforcement learning to discover tool use Baker et al. (2019). Their system is able to achieve fascinating results using machine learning. The system trains agents to play 'hide and seek' in a world with movable obstacles (which the hiders learn to use to build a room) and movable ramps (which the seekers learn to use to get over the walls of the room). The system still uses rewards from the environment to guide learning by the agents, and the number of episodes used by the system is in the hundreds of millions. But such advances are of keen interest in relation to the learning approach taken by Leela.

Beaulieu et al. have introduced a meta-learning technique which aims to eliminate the catastrophic forgetting problem seen in neural networks Beaulieu et al. (2020). Their approach is tested on the problem of neural network classifiers, but they propose that the technique could be extended to reinforcement learning networks learning to perform actions. The meta-learning network presented learns to modulate the activations of a prediction network which allows for selective plasticity when learning new data. Using Leela to interact with a network in a similar way may allow Leela to guide learning in a DQN such that benefits of both techniques could be combined.

6. Conclusion

In the next decade AI will need to begin learning with fewer labeled examples, reasoning about the world, and planning complex actions. We've presented Leela, which uses the schema mechanism for learning concepts about the world in a constructivist way which does not require labeled examples or an explicit reward from the environment. We compared Leela's schema-based learning to a DQN enhanced with hindsight learning and found that Leela was able to learn hand-eye coordination with fewer action steps on the world than the DQN. Also, we showed that Leela forms concepts about goals naturally during exploration, allowing learning which can be applied to multiple goal categories efficiently, while a DQN model took significantly longer to train as more goals were added during training.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Informa-*

tion Processing Systems 30, pages 5048–5058. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7090-hindsight-experience-replay.pdf.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent Tool Use From Multi-Agent Autocurricula. *arXiv e-prints*, art. arXiv:1909.07528, September 2019.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O. Stanley, Jeff Clune, and Nick Cheney. Learning to Continually Learn. *arXiv e-prints*, art. arXiv:2002.09571, February 2020.

Yoshua Bengio, Geoffrey E Hinton, and Yann LeCun. AAAI-20: Turing award winners event. *Association for the Advancement of Artificial Intelligence*, 2020. URL https://vimeo.com/390347111.

Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017. URL http://arxiv.org/abs/1711.03902.

Paul Cisek and John F. Kalaska. Neural mechanisms for interacting with a world full of action choices. *Annual Review of Neuroscience*, 33(1):269–298, 2010. doi: 10.1146/annurev.neuro.051508.135409. URL https://doi.org/10.1146/annurev.neuro.051508.135409. PMID: 20345247.

Gary L. Drescher. Genetic ai: Translating piaget into lisp. Technical report, USA, 1986.

Gary L. Drescher. *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1991. ISBN 0262041200.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints*, art. arXiv:1509.02971, Sep 2015.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992699. URL https://doi.org/10.1007/BF00992699.

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 00280836. URL http://dx.doi.org/10.1038/nature14236.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Jean Piaget. Part i: Cognitive development in children: Piaget development and learning. *Journal of Research in Science Teaching*, 2(3):176–186, 1964. doi: 10.1002/tea.3660020306. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/tea.3660020306.