Symbolic Guidance for Constructivist Learning by Neural Model

Steve Kommrusch SK@LEELA-AI.COM
Henry Minsky HQM@LEELA-AI.COM
Leela AI, Cambridge, Massachusetts, USA

Editor: Kristinn R. Thórisson

Abstract

Deep learning has made impressive strides but still lacks key concepts necessary to truly reason and act in the world. In parallel, symbolic learning systems have shown success at certain types of abstract reasoning, as demonstrated in the Abstraction and Reasoning Challenge by Kaggle. Yet, these symbolic learners are challenged when generalizing to data from the analog world. This paper will present and evaluate ideas for using symbolic learning concepts to guide learning of a neural network in a constructivist way. We aim to show how a neural network with internal feedback can be used - somewhat like the brain - to suggest the proper actions to take and predict the results of those actions. In other words, the system will create an internal model of the world on which it can reason. The neurosymbolic system we consider is inspired by the symbolic learning system from Gary Dresher and neural network cortical columns as discussed by Jeff Hawkins. The hybrid system aims to create a synthesis which can generalize on real-world concepts while also quickly learning from few examples as the world changes and prior experience is found to be inaccurate.

Keywords: constructivist, cortical column, neurosymbolic, machine learning, artificial intelligence

1. Introduction

One goal of artificial intelligence research is to build systems that can integrate concepts of the world and take actions to reach goals within that world. In February 2020 at the Turing Award Winners event for AAAI-20, Yann LeCun identified three 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.

The Leela symbolic learning system is capable of forming hypotheses about the world and testing them at the same time actions are performed Kommrusch (2020). As the Leela symbolic learner 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 that state will change when an action is performed.

While Leela is able to form and test concepts in a world with various discrete items, it would benefit from an architecture that allows it to generalize in the way that neural network can with feature embeddings As more information is learned in a neural network, internal weights and hence representations of information get slightly adjusted. In this paper we seek a way to merge success in the symbolic domain with a constructivist learning agent with modern neural network techniques.

Our key contributions are:

1. We propose using the artificial cortical column models to encapsulate information about a concept in ways that are similar to schema representations of information.

2. We present a path for future research on how key constructivist concepts from the work of Piaget and Drescher can be implemented by cortical column models.

3. We integrate ideas from transformer models which allow both cortical column output values and interconnect behaviors to be learned while the learner is acting on the world.

2. Background

We first explain recent connectionist (neural network) approaches to artificial intelligence, then discuss approaches for adjusting and augmenting these approaches, and finally summarize prior work at Leela AI on symbolic constructivist models.

2.1. Transformers: a modern neural network

In general, modern neural networks work by using a model to receive input from the world and produce desired outputs based on a complex learned mathematical function. Hence, neural networks have been built with convolution filters that, given a digital image as input, can learn to recognize common objects Wu et al. (2019). Similarly, neural networks can be trained to produce sequential text output when given text as input (such as large language models like GPT-3 Brown et al. (2020)). These models may have many layers where the output of one layer of analysis is fed into another. They may involve billions of parameters which determine the complex computation done on the input data in order to produce the output. Often millions of labeled examples are needed to produce a network that performs with acceptable accuracy on the given task.

For our proposal in this paper, we will want to propose a neural network approach which allows pieces of information to interact with other pieces of information. We want not just the information itself but also the connections to other information with which it interacts to be learnable. The transformer model Ketan Doshi (2021) introduces a technique where the representation for a word can be learned but a word can also learn which other words in a sentence should be used to create a more complex representation of the meaning. Figure 1 illustrates pictorially how learned embeddings for a word sequence can be processed to create a higher-level embedding which combines information from multiple words in the sequence together. The equation diagrammed in Figure 1 is the self-attention computation below:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

At the lowest level the 'embedding' for a word in a transformer model may be the result of individual word embeddings such as those produced by Word2Vec Mikolov et al. (2013). From these embeddings, query, key, and value vectors are created with learned functions (shown by the blue 'Linear' boxes in the diagram). The Query is used to search for other words in the sequence that are relevant to help improve the 'understanding' of a word. All of the words in the sequence produce Keys, which conceptually identify the type of information represented by a given word. Keys are matched to the Queries via a matrix multiply shown within the Softmax function. The matching is a simple dot product between each Query and each Key for every token in the whole sequence. Finally, given the matches found by the Query and Key math, the Values for the selected words are aggregated together to produce the final new embedding for each word. For example, the embedding at the bottom for the word 'You' is interpreted by Linear mappings to create a Query, Key, and Value. In this way the initial embedding for 'You' can be enhanced to include values from other words such as 'are welcome' such that the next layer embedding can encode that the 'You' is someone who is welcome instead of just 'You'. The next transformer layer above this one may aggregate even further - perhaps even to include information about 'You' from other sentences.

2.2. Inferring and enforcing rules with neural networks

In this paper we wish to explore how rules and symbolic information can be used to interact with neural networks to extract symbolic rules from a trained network. This would be valuable, in part, because neural networks are difficult to interpret. Figure 2 summarizes the approach by Taha and Ghosh Taha and Ghosh (1999), which extracts rules from a neural network and then uses them to inform decisions and revise the neural network. In their paper they acknowledge that rule extraction from a trained network is a non-trivial problem. Yet, they discuss several interesting approaches that help to illuminate the problem of extracting understandable symbolic rules from a neural network.

The converse of deriving rules from a neural network would be to use rules to guide a neural network. In Figure 3, Dash *et al.* describe multiple ways in which prior symbolic knowledge can be used to affect the learning done by a neural network Dash et al. (2022). For example, a loss function can be biased to strongly train a network when certain rules are violated; similarly the training data itself can be adjusted to more quickly provide examples that follow or violate known rules on the data. Using such techniques the neural network can learn to incorporate knowledge about the world that was provided from outside the network.

2.3. Constructivist AI

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

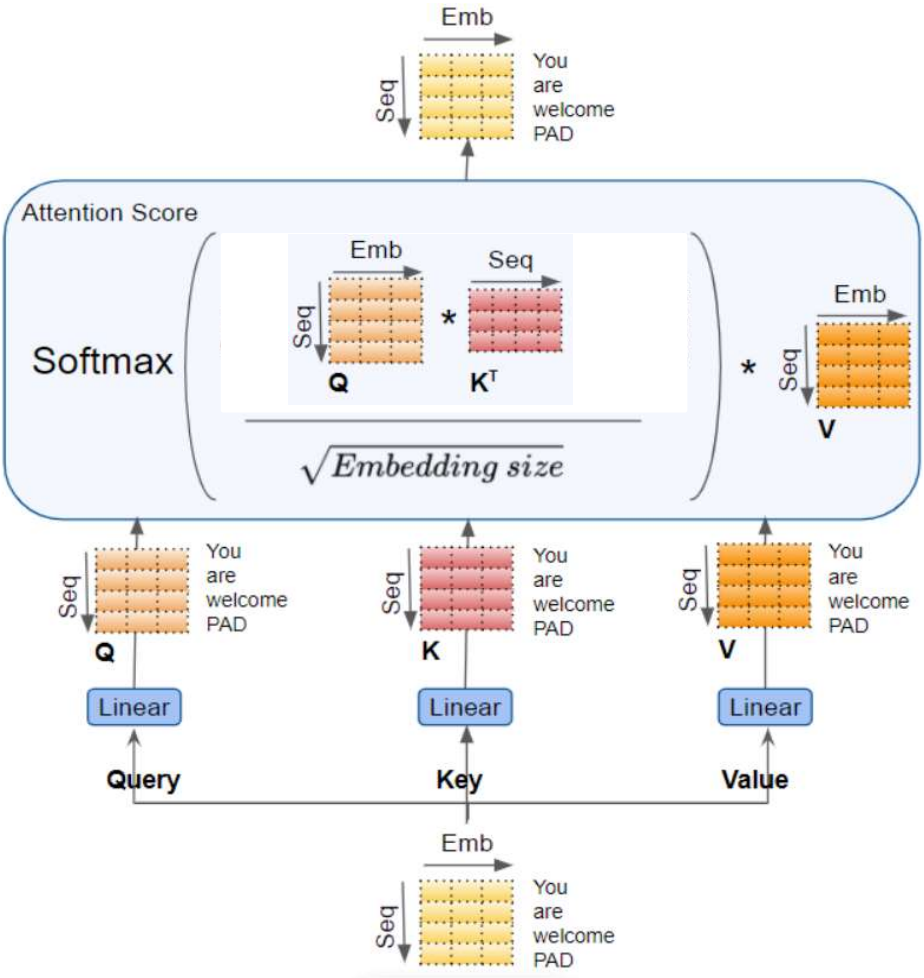

Figure 1: In transformers, matches between Query and Key encodings are used to select and combine Value encodings to next stage Vaswani et al. (2017); Ketan Doshi (2021).

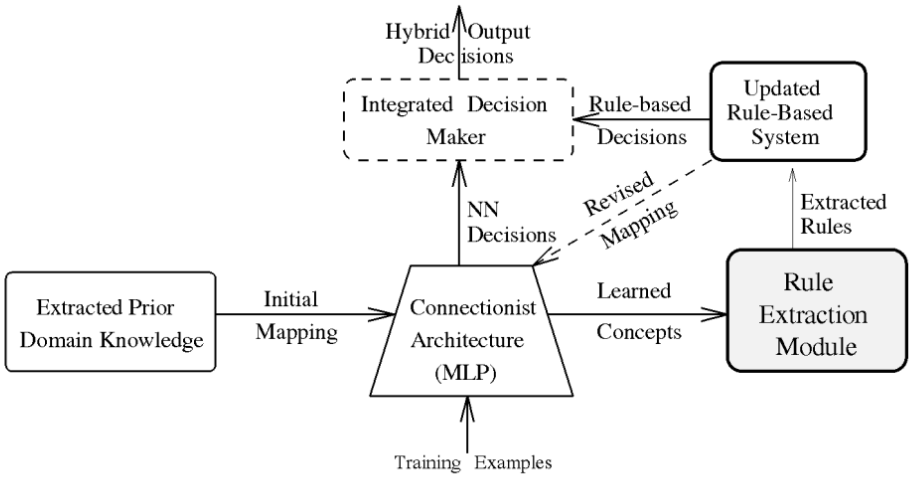

Figure 2: Example system for extracting and using rules from a neural network system Taha and Ghosh (1999).

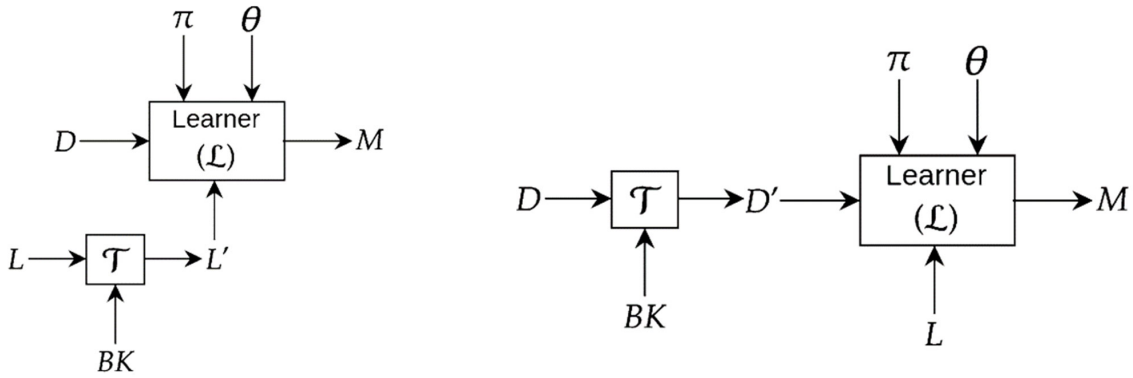

Figure 3: Guiding neural network learning with external background knowledge Dash et al. (2022). BK refers to background knowledge, D is training data, M is the trained model, L is the training loss, π is the reinforcement learning policy, and θ is the set of parameters for the model. Given the goal of having M perform well, the left figure illustrates background knowledge transforming the loss function, the right figure shows it transforming the input data before training.

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 four 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 two types of schema: i.e. passing hand before face
  - Secondary circular reactions: actions involving external objects begin
  - Coordination of secondary circular reactions: 'first proper intelligence'; means and ends; goals; object permanence
  - Tertiary circular reactions: curiosity about object properties
  - Internalization of schemas: insight, creativity, use of 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; 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 4. Leela AI has built a schema system using Python and modern GPU compute libraries that implements key concepts from Drescher's symbolic schema proposal (Kommrusch (2020)). Initially, the learning system has no understanding of how actions affect its sensory input. For example, the system might learn a schema that suggests when a hand is seen on the left side of the visual field, then the action of moving the hand to the right results in the hand being seen in the middle of the visual field. 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.

As shown in Figure 4, a completed schema tracks probabilities related to affecting the world given an action performed and information about the current world state. Initially, an action may be performed for which no schema exists; in this case, a 'bare schema' would be created if certain effects on the world seem even slightly more likely than chance. For

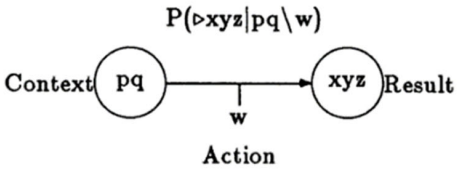

Figure 4: Schema diagram showing context, action, and result (Drescher (1986)). The $P()$ function shown indicates that the schema tracks the probability of the three results `xyz` occurring given action `w` is taken when the 2 conditions `pq` are true.

example, if a hand is moved left then perhaps the eye detects an object in the left field of view more often than if some other action were performed. After creating a schema that only track results, statistics on the world state before the action is performed may indicate that a schema is more reliable if it predicts a given outcome given the observed state of the world. For example, if the eye is currently seeing the hand and we perform the action to move the hand left THEN the eye will see an object (the hand) in the left field of view.

As schemas are built 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.

Like neural networks, the foundational model for the constructivist symbolic learning approach was initially proposed decades ago; learning models based on schemas can help address some of the key challenges facing AI today.

3. Modeling brain components

Figure 5 shows components that have been proposed to organize research on the problem of addressing 'System 2' thinking (Matsuo et al. (2022)). Yann LeCun, a winner of the 2018 Turing Award and a founder of convolutional neural networks has been exploring this approach. Matsuo, LeCun, *et al.* discuss proposals for how different components in this model will learn. Of note is that their concept of a world model has a form of compression or relevance such that unimportant facets of the world are not predicted nor depended on as the system operates. For example, a self-driving vehicle would not need to predict where each leaf in a tree was in order to drive well, and so learning about the world should not consider such detail when learning the driving task.

Figure 6 shows a proposed structure in the human brain Lübke and Feldmeyer (2007). Jeff Hawkins and his team have been investigating ways to utilize artificial cortical columns to model brain functions. Each cortical column is proposed to represent a given concept, and the concepts may enter an anticipatory state given certain conditions. When the condition occurs as predicted, the brain continues making predictions with low-energy computations, but if a prediction fails a learning event triggers and neurons adjust to the new information.

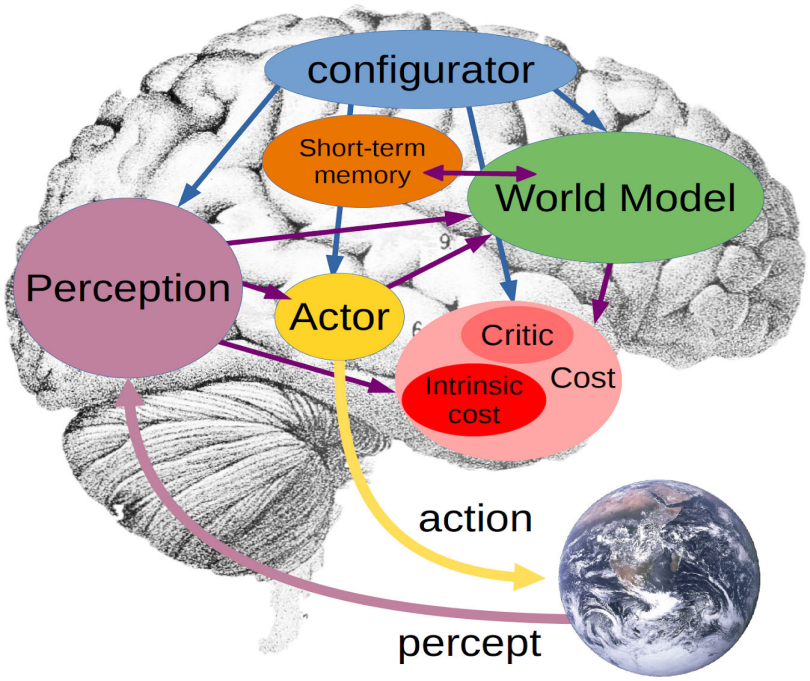

Figure 5: Interconnected artificial neural network models including world model Matsuo et al. (2022).

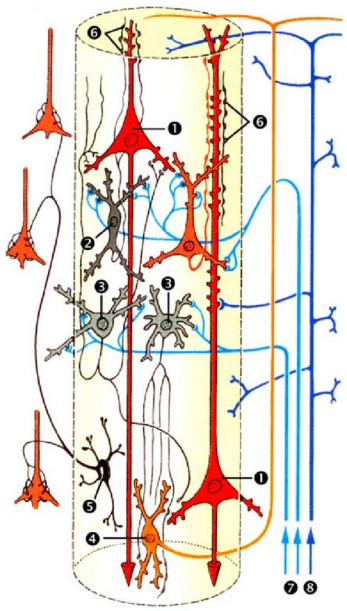

Figure 6: Tightly connected neurons forming a cortical column Lübke and Feldmeyer (2007).

4. Constructivist concepts with artificial cortical columns

We now propose a system that merges concepts from the symbolic schema mechanism used by Leela with neural networks modeling cortical columns. The motivations for this synthesis are:

- Modern connectionist architectures require labeled data in order to learn well; Leela's symbolic schema system can generate hypotheses about the world and test them in a self-supervised way. We want to maintain that feature.
- Leela's symbolic system presumes a world discretized into specific symbols; we want to explore supporting schemas that have learnable context and result representations, such as would be allowed by using learnable embeddings.

In contrast to the world model proposed in Figure 5, this proposal spreads out the world model into cortical columns. Having the columns anticipate when they should fire, effectively distributes a model of the world into the columns themselves. We propose that an artificial cortical column, like their biological counterparts, have a small number of neurons interconnected by only a few layers - perhaps a few hundred neurons in a 4-6 layer structure. Artificial cortical columns may have multiple inputs and multiple outputs.

Four types of artificial cortical columns In the symbolic schema mechanism shown in Figure 4, schemas learn to anticipate changes in symbolic item states based on when an action occurs given certain states in the world of other items. One of our goals in this paper

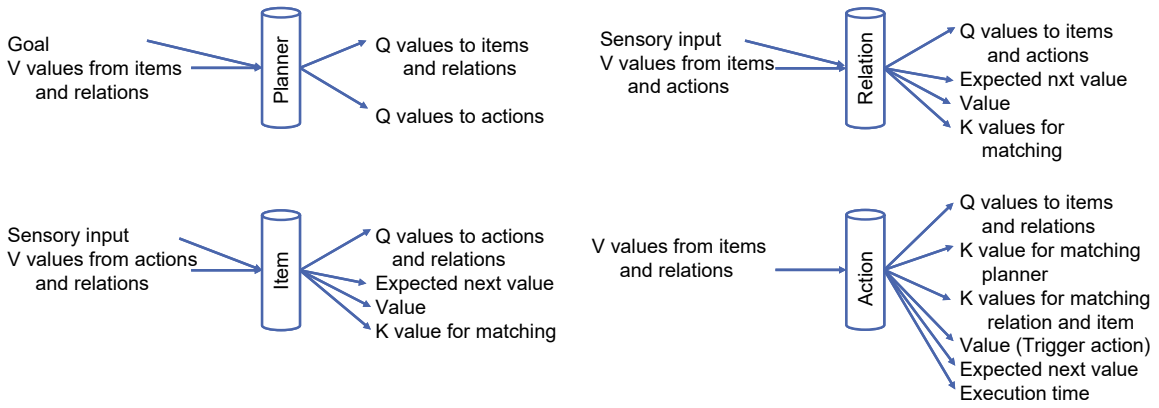

Figure 7: Constructivist concepts can be represented as cortical columns and interconnected with key (K), query (Q), and value (V) embeddings.

is to allow these symbolic items to refer to more abstract concepts or object representations, hence we propose that some cortical columns represent Items. Our cortical columns will create query, key, and value outputs used to interconnect them in ways we describe in other paragraphs of this section. In the symbolic schema mechanism, simple actions can be combined into composite actions as the system learns. In order to allow actions themselves to have a learnable embedding, we propose that some cortical columns represent Actions. In the symbolic schema mechanism, synthetic items can be used to learn certain features of an item. For example, a synthetic item can be used to represent 'if I touch this object I will feel a smooth surface'. We propose that such adjectives and temporary object relations also have learnable embeddings and hence we propose that some cortical columns represent Relations. Finally, when a goal is desired in the symbolic schema system it can usually be defined as a set of items in the world that we wish to 'turn on' - such as 'feel that my stomach is full'. We want goal searches to be a special category of column interactions hence our final cortical column category is Planner. These four cortical column types are shown in Figure 7.

In general, the input provided to cortical columns are computed as a standard attention equation:

$$\text{Input}(Q, K, V) \approx \text{softmax}(QK^T)V$$

Initial training and early schemas We do not detail the initial training but propose that existing training approaches could be used to seed the four types of artificial cortical columns discussed in Figure 7. One could use existing training data (such as the COCO dataset Lin et al. (2014)) to train items to recognize common objects. One could use existing Leela symbolic training to seed initial action and relation columns. Additional sources for initialization may be found useful as the system is developed. The main goal would be to allow the system to begin with initially valuable Q, K,and V matrices for a set of cortical columns representing some knowledge base and then let the system adjust and expand the knowledge.

This initial training will allow initial columns to be associated with labeled data. For example, an item column related to 'table' may be specifically identified. Such initial mapping allows for future guidance even as new columns spin off. For example, predicting that a hammer and nails can assemble wood into a 'table' would be aided if the 'table' item was expected to be detected by a traditional object recognition model running on a video stream. If the 'table' item was expected to be detected but the object recognition did not see a table, backwards propagation to improve the neural networks could be done, correcting the items, actions, and relations that had predicted that a table would be created.

Passive observation may create or delete columns When observing the world, for example by processing a video input feed, item columns will be activated. Consider a case where the network has an existing item for 'cat', 'cow', and other domestic animals. As a cat enters the field of view but before it has been fully recognized the 'cat' cortical column would assert that its 'expected next value' should be true. If the cat continues entering the scene and a cat is confirmed, the system continues as normal - other items in the world that may make use of the 'cat' item can process this information. But consider if instead of a cat a large lion enters the scene; here the 'cat' prediction is not confirmed. One approach to the failure would be to emulate the concept of 'spin-off' from the schema environment and create a new cortical column. A quick back-propagation training would be done such that the 'cat' cortical column would be trained to not recognize the early data as a cat. The new column would be trained once on the lion so that it will recognize it.

Like in the symbolic system proposed by Drescher, metrics related to columns could be used to age them out in order to improve system performance. For example, we have query, key, and values for the various columns; if their value has not changed state in some time, and we see that their queries and keys do not get matches (i.e., the column information is not being used), then the column would be a candidate for removal. Indeed, if the column has been involved in a spin-off (either as the original or spun-off column) then it can be removed with relative safety since it could be recreated if conditions warrant it.

The Relation cortical columns can also be engaged even during passing observation. These columns track common object relations, such as 'being on top of' or 'being held by'. Specific relations (such as 'I see an object near the window in the room') could be turned into items in a method similar to synthetic item generation by the schema mechanism.

Active exploration Action cortical columns are attached to motion actions the system can take to affect the world (such as moving eyes or arms). As time advances, existing item and relation states are processed to create K values for items and relations. The K values from the items and relations may match the Q value for an Action and these are used as input to the Action cortical columns. In turn, the Action columns produce K values which may match Q values from items and relations and then the input to the Item and Relation columns may be affected by the Action value provided.

Like Item columns, Action columns can be duplicated in order to learn different effects for the same action. This can occur when an action which has been reliable in the past becomes inaccurate in the future. This reliability can be tracked as action triggering varies or by detecting when the items and relations, to which the action is connected by Q/K values, do not enter their predicted next state.

In the schema mechanism proposed by Drescher, composite actions can be formed which allow a predicted world state to be achieved through multiple action paths. To aid in such behaviors we propose an interconnection between items, relations, and actions, thereby allowing for complex actions to be built. Consider a goal of 'having food in stomach' as a particular relation. When this relation becomes a value that occurs often, the system could add a new synthetic item to help create action chains. The new item could be created with its Q and K values initially such that the effect of this item currently being on informs the 'having food in stomach' relation. In this way, the relation may learn to anticipate turning on. The new item is also configured so that its Q and K values are relevant to the last action that fired before 'having food in stomach' recently occurred. Now we have a new item in the world, which is a precursor to the last action resulting in 'having food in stomach'. This is a synthetic item of sorts and it may start learning which actions result in causing it to turn on. This can in turn result in further synthetic items being created and attached to 'having food in stomach' by virtue of the Q and K settings. The use of Q and K values to interconnect components means that new columns added to the world do not generally need to be explicitly connected to related columns. The Q and K matching will allow relevant columns to interact with each other.

Planner Given a goal, the planner searches for actions relevant to accomplishing the goal by creating Q values which attempt to match K values from actions. As actions provide V values to the planner, the planner will query items and relations in the world. An algorithmic component for the planner could identify which actions are found to be immediately executable in the current world state but also which might be the final step to achieve the goal. Using these 'first' and 'last' actions found, backwards and forwards chaining could be used to find an action sequence that is expected to follow the goal.

Note that actions include estimations of time to complete. This time could be updated by detecting when the world detects changes after the initiation of an action. That is, Q and K values feed data into the action column and cause it to activate. Later, items and relations which are connected by Q and K values from the action column are seen to change. The time between these events indicate the time to complete the action.

Subactivation and backchaining Through subactivation we can 'imagine' how the cortical columns react when certain goals are considered or certain sensory input occurs. When only reliable actions are modeled in subactivation, items and relations can have training done (based on incorrectly predicted firing) in much the same way Drescher proposed subactivation to work in the symbolic world.

Subactivation is thus a key tool for forward chaining planning steps - the state of items and relations in the imagined world can change as actions are imagined. But consider the case where 'backwards' chaining of the world is desired in order. That is, given a goal state for the world, we want to find the final action needed to achieve the goal, then the action preceding that final step which allows the final action to be realized, and so on back to an action which can be taken in the current world. For such searches, back-propagation could be done without updating weights to find which items needed to be turned on in order for an action to be viable. For example, consider again 'having food in stomach' as a goal. The final action might be 'put fork in mouth' which might depend on 'food on fork' to be a relation in the world; then 'food on fork' could be seen as a new goal and backwards

chaining could continue from there until the current world state allows for the action to be found.

5. Conclusion

In the next decade, artificial intelligence will need to begin learning with fewer labeled examples, reasoning about the world, and planning complex actions. We've presented a set of concepts which have been tested individually and might be combined as we describe to support a constructivist approach with neural network cortical column models. We hope that this approach can be investigated further to gain more understanding about how we advance the field of AI and correctly model complex thought.

References

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tirtharaj Dash, Sharad Chitlangia, Aditya Ahuja, and Ashwin Srinivasan. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1), jan 2022. doi: 10.1038/s41598-021-04590-0. URL https://doi.org/10.1038%2Fs41598-021-04590-0.

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.

Ketan Doshi. Transformers explained visually (part 2): How it works, step-by-step. https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34, 2021. [Online; accessed 26-September-2022].

Steve Kommrusch. Self-supervised learning for multi-goal grid world: Comparing leela and deep q network. In Henry Minsky, Paul Robertson, Olivier L. Georgeon, Milan Minsky, and Cyrus Shaoul, editors, *Proceedings of the First International Workshop on Self-Supervised Learning*, volume 131 of *Proceedings of Machine Learning Research*,

pages 72–88. PMLR, 27–28 Feb 2020. URL https://proceedings.mlr.press/v131/kommrusch20a.html.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv e-prints*, art. arXiv:1405.0312, May 2014.

J Lübke and D. Feldmeyer. Excitatory signal flow and connectivity in a cortical column: focus on barrel cortex. *Brain Struct Funct.*, Jun 2007.

Yutaka Matsuo, Yann LeCun, Maneesh Sahani, Doina Precup, David Silver, Masashi Sugiyama, Eiji Uchibe, and Jun Morimoto. Deep learning, reinforcement learning, and world models. *Neural Networks*, 152:267–275, 2022. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2022.03.037. URL https://www.sciencedirect.com/science/article/pii/S0893608022001150.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

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.

I.A. Taha and J. Ghosh. Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):448–463, 1999. doi: 10.1109/69.774103.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.