The Sensory Neuron as a Transformer: Permutation-Invariant Neural Networks for Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In biological systems, we often observe complex global behavior emerge from a collection of agents interacting with each other in their environment, with each individual agent acting only on locally available information, without knowing the full picture. Such systems have inspired development of artificial intelligence algorithms in areas such as swarm optimization and cellular automata. Motivated by the emergence of collective behavior from complex cellular systems, we build systems that feed each sensory input from the environment into distinct, but identical neural networks, each with no fixed relationship with one another. We show that these sensory networks can be trained to integrate information received locally, and through communication via an attention mechanism, can collectively produce a globally coherent policy. Moreover, the system can still perform its task even if the ordering of its inputs is randomly permuted several times during an episode. These permutation invariant systems also display useful robustness and generalization properties that are broadly applicable. Interactive demo and videos of our results: https://neuripsanon.github.io/results/

1 Introduction

Biological neural networks have a remarkable ability to self-organize by constantly rewiring neural connections, allowing our brains to continually adapt to changes in the environment observed by our sensory neurons. This malleability enables the same region of the brain to process different types of sensory input and perform different functions. For instance, experimental studies [63, 64] reveal that sound and touch sensations will activate neurons in the unused visual cortex for the blind, thereby increasing capacity for processing information from other senses.

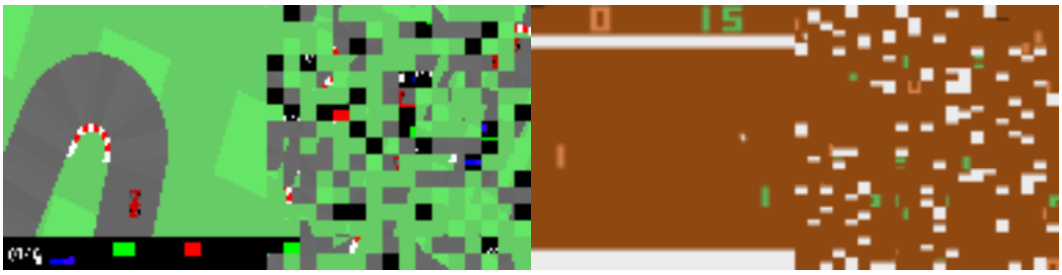

Figure 1: *Comparison of visual input intended for the game player, and what our system receives.* We partition the visual input from CarRacing (Left) and Atari Pong (right) into a 2D grid of small patches, and randomly permute their ordering. Each sensory neuron in the system receives a stream of visual input at a particular permuted patch location, and through coordination, must complete the task at hand, even if the visual ordering is randomly permuted again several times during an episode.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Brains can also adapt to changes in sensory inputs, even when they are fed into the *wrong* channels. In several studies, low resolution visual information channeled from a head-mounted camera of a blind person is received through mechanical pokes on the person's back [6], or converted to a grid of electrodes placed on the tongue [7], enabling the person to regain rudimentary vision from other sensory receptors. In other experiments, subjects are able to ride a "backwards" bicycle [67]—one that turns left when you steer right—or navigate while wearing "upside down" goggles [65]. Our brains can adapt to sudden changes in sensory inputs and make sense of the new mapping of information.

Modern deep learning systems, however, are generally unable to adapt to a sudden reordering of sensory inputs, unless the model is retrained, or if the user manually corrects the ordering of the inputs for the model. However, techniques from continual meta-learning, such as adaptive weights [2, 38, 68], Hebbian-learning [52, 53, 57], and model-based [1, 24, 39, 40] approaches can help the model adapt to such changes, and remain a promising active area of research.

In this work, we investigate agents that are explicitly trained to deal with sudden random reordering of their sensory inputs while performing a task. Motivated by recent developments in self-organizing neural networks [31, 55, 62] related to cellular automata [17, 20, 21, 58, 82], in our experiments, we feed each sensory input (which could be an individual state from a continuous control environment, or a patch of pixels from visual environments) into an individual neural network module that integrates information from only this particular sensory input channel over time. While receiving information locally, each of these individual sensory neural network modules also continually broadcasts an output message. Inspired by the Set Transformer [49, 78] architecture, an attention mechanism combines these messages to form a global latent code which is then converted into the agent's action space. The attention mechanism can be viewed as a form of adaptive weights of a neural network, and in this context, allows for an arbitrary number of sensory inputs that can be processed in any random order.

In our experiments, we find that each individual sensory neural network module, despite receiving only localized information, can still collectively produce a globally coherent policy, and that such a system can be trained to perform tasks in several popular reinforcement learning (RL) environments. Furthermore, our system is able to utilize a varying number of sensory input channels in any randomly permuted order, even when the order is shuffled again several times during an episode.

Permutation invariant systems have several advantages over traditional fixed-input systems. We find that encouraging a system to learn a coherent representation of a permutation invariant observation space leads to policies that are more robust and generalizes better to unseen situations. We show that, without additional training, our system continues to function even when we inject additional input channels containing noise or redundant information. In visual environments, we show that our system can be trained to perform a task even if it is given only a small fraction of randomly chosen patches from the screen, and at test time, if given more patches, the system takes advantage of the additional information to perform better. We also demonstrate that our system is able to generalize to visual environments with different background images, despite training on a single fixed background. Lastly, to make training more practical, we propose a behavioral cloning scheme to convert policies trained with existing methods into a permutation invariant policy with desirable properties.

2 Related Work

In **Neuroscience**, sensory substitution refers to the use of one sensory modality (i.e., hearing) to supply environmental information normally gathered by another sense (i.e., vision). When sensory inputs from an animal's eye are redirected to its auditory cortex, studies [71] have shown that the rewired animal can learn to interpret inputs to its auditory cortex like normal vision. Pioneering work [6] in sensory substitution for human-machine interface enabled blind patients to "see" with stimulation from a vibrating plate resting against the user's back, and later on, with electrical stimulation [7] from a grid placed on the tongue, carrying a low resolution image from a video camera on the user's forehead. In the words of Bach-y-Rita describing their work: "The brain is able to use information coming from the skin as if it were coming from the eyes. We don't see with the eyes or hear with the ears, these are just the receptors, seeing and hearing in fact goes on in the brain." [29]

Unlike the activation functions in artificial neural networks, biological neurons are able to perform a vastly greater range of computations. Individual neurons are able to compute a range of non-linear functions and logical functions including the XOR operation [13]. With the enormous range of logical functions they could compute as the number of inputs increase, entire artificial networks have been used to model the physical properties of an individual biological neuron and its connections [9, 61].

In consciousness studies, Global Workspace Theory (GWT) [5, 23, 77] conjectures that consciousness is accomplished by a distributed network of specialized modules equipped with a common working memory, called a global workspace, whose contents can be broadcast to the system as a whole [4].

In this work, we take inspiration from the complexities of individual neurons and their ability to self-wire and adapt to changing sensory inputs. We feed each sensory input from the environment into individual sensory neural network modules that share the same parameters. Applying concepts from GWT, each "sensory neuron" will integrate its stream of local information into a message that is captured by an attention mechanism to produce a global latent code for downstream use.

Cellular Automata (CA) [20, 21, 58] are computational systems consisting of a grid of cells that perform computation by having each cell communicate with its immediate neighbors and performing a local computation to update its internal state. Such local interactions in a self-organizing system proved to be useful in modeling complex systems [82] and have been applied to model non-linear dynamics in various fields [17]. Cellular Neural Networks [19] were first introduced in the 1980s to use neural networks in place of the algorithmic cells in CA systems. They were applied to perform image processing operations with parallel computation. Eventually, the concept of self-organizing neural networks found its way into deep learning in the form of Graph Neural Networks (GNN) [83].

Taking advantage of modern deep learning tools and compute, recent work demonstrate that self-organized neural networks performing only local computation can generate (and re-generate) coherent images [55] and voxel scenes [74, 88], and even perform image classification [62]. Self-organizing neural network agents have been proposed in the RL domain [14, 15, 59, 60], with recent work demonstrating that shared local policies at the actuator level [44], through communicating with their immediate neighbors, can learn a global coherent policy for continuous control locomotion tasks.

While existing CA-based approaches present a modular, self-organized solution, they are not inherently permutation invariant. In our work, we build on neural CA, and enable each cell to communicate beyond than its immediate neighbors via an attention mechanism that enables permutation invariance.

Meta-learning recurrent neural networks (RNN) [27, 43, 80] have been proposed to approach the problem of learning the learning rules for a neural network using the reward or error signal, enabling meta-learners to learn to solve problems presented outside of their original training domains. The goals are to enable agents to continually learn from their environments in a single lifetime episode, and to achieve much better data efficiency than conventional learning methods such as stochastic gradient descent (SGD). A meta-learned policy that can adapt the weights of a neural network to its inputs during inference time have been proposed in fast weights [68, 70], associative weights [2], hypernetworks [38], and Hebbian-learning [52, 53] approaches. Recently works [47, 66] combine ideas of replication and self-organization from CA with meta-learning RNNs, and have demonstrated that modular meta-learning RNN systems not only can learn to perform SGD-like learning rules, but can also discover more general learning rules that transfer to classification tasks on unseen datasets.

In contrast, the system presented here do not use an error or reward signal to meta-learn or fine-tune its policy. But rather, by using the shared modular building blocks from the meta-learning literature, we focus on learning or converting an existing policy to one that is permutation invariant, and we examine the characteristics such policies exhibit in a zero-shot setting, without additional training.

Attention can be viewed as an adaptive weight mechanism that alters the weight connections of a neural network layer based on what the inputs are. Linear *dot-product* attention have first been proposed for meta-learning [69], and versions of linear attention with $softmax$ non-linearity appeared later [37, 51], now made popular with Transformer [78]. The adaptive nature of attention provided the Transformer with a high degree of expressiveness, enabling it to learn inductive biases from large datasets and have been incorporated into state-of-the-art methods in natural language processing [11, 25], image recognition [26] and generation [30], audio and video domains [35, 45, 75].

Attention mechanisms has found many uses for RL [16, 56, 72, 76, 87]. Our work here specifically uses attention to enable communication between arbitrary number of modules in an RL agent. While previous work [36, 46, 54, 79, 85, 89] utilized attention as a communication mechanism between independent neural network modules of a GNN, our work focuses on studying the permutation invariant properties of attention-based communication applied to RL agents. Related work [50] used permutation invariant critics to improve performance of multi-agent RL. Building on DeepSets [86], Set Transformers [49] investigated the use of attention explicitly for permutation invariant problems that deal with set-structured data, which have provided the theoretical foundation for our work.

3 Method

We model the observation from the external environment as an arbitrarily ordered, variable-length list of sensory inputs that the agent receives, each of which is passed into its own *sensory neuron*, a neural network module. Each sensory neuron only has partial access to the agent's observation, and with no assumption about the ordering of the inputs, at time t, the ith neuron can see only the ith component of the observation $o_t[i]$. In addition, each sensory neuron is also aware of the agent's previous action a_{t-1}. We find that the combination of $(o_t[i], a_{t-1})$ allows the neuron to learn the relationship between its observed signal and the sequence of actions performed. By integrating the information received over time up to time t, each sensory neuron computes a message f_k that is broadcast to the rest of the system. An attention mechanism is responsible for summarizing these messages into a global latent code m_t. We will show that this latent code produced is not only informative, but also consistent even when the ordering of the input list is changed or shuffled.

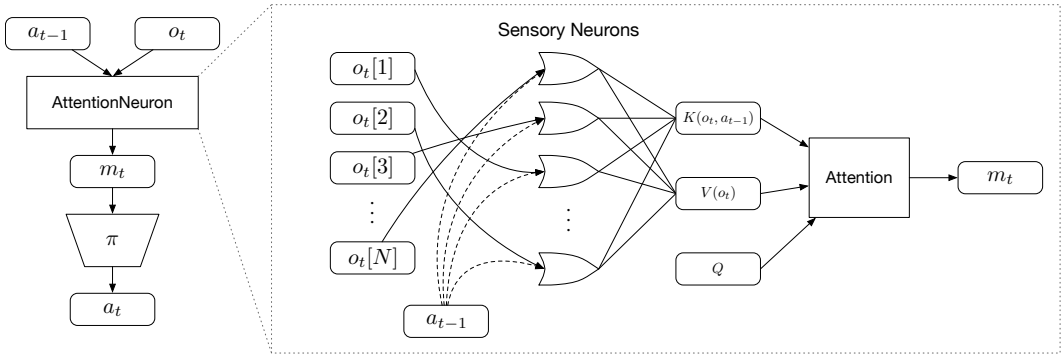

Figure 2: *Overview of Method.* AttentionNeuron is a standalone layer, in which each sensory neuron only has access to a part of the unordered observations o_t. Together with the agent's previous action a_{t-1}, each neuron generates messages independently using the shared functions $f_k(o_t[i], a_{t-1})$ and $f_v(o_t[i])$. The attention mechanism summarizes the messages into a global latent code m_t.

Our method uses the attention mechanism in Transformer [78] which is shown to be permutation invariant [49]. Here, the attention's output m_t does not change under any ordering of observation o_t:

$$m_t = \sigma\Big(\frac{QW_q \times [K(o_t, a_{t-1})W_k]^\intercal}{\sqrt{d_q}}\Big)[V(o_t)W_v] \tag{1}$$

$$K(o_t, a_{t-1}) = \begin{bmatrix} f_k(o_t[1], a_{t-1}) \\ \cdots \\ f_k(o_t[N], a_{t-1}) \end{bmatrix} \in \mathcal{R}^{N \times d_{f_k}}, V(o_t) = \begin{bmatrix} f_v(o_t[1]) \\ \cdots \\ f_v(o_t[N]) \end{bmatrix} \in \mathcal{R}^{N \times d_{f_v}} \tag{2}$$

where $o_t \in \mathcal{R}^{|O|}$ is the agent's observation at time t, $a_{t-1} \in \mathcal{R}^{|A|}$ is its action at time $t-1$, $|O|$ and $|A|$ are the dimensions of the observation and action spaces. The sensory neurons' operations are collectively described by the functions K and V. For $K(o_t, a_{t-1})$, N is the number of sensory neurons and $f_k(o_t[i], a_{t-1}) : \mathcal{R}^{|o_t[i]|+|A|} \mapsto \mathcal{R}^{d_{f_k}}$ is an operation performed by the ith neuron. $V(o_t)$ is a pass-through function ($f_v(o_t) = o_t$). If the dimensionality of $o_t[i]$ is greater than 1, $f_v(o_t[i]) : \mathcal{R}^{|o_t[i]|} \mapsto \mathcal{R}^{d_{f_v}}$ flattens $o_t[i]$ into a one dimensional vector. Although each sensory neuron receives different inputs and therefore may have different internal states, they share the same f_k, whose parameters are shared among all sensory neurons.

Equation 1 describes how the attention mechanism summarizes the output messages from the variable-length list of sensory neurons into a global latent code. $Q \in \mathcal{R}^{M \times d_e}$ carries M embeddings used in the attention mechanism, $W_q \in \mathcal{R}^{d_e \times d_q}$, $W_k \in \mathcal{R}^{d_{f_k} \times d_q}$ and $W_v \in \mathcal{R}^{d_{f_v} \times d_m}$ are the projection matrices for Query, Key and Value, and $\sigma(\cdot)$ is a non-linear activation function. Note that although we can absorb W_k and W_v into $K(o_t, a_{t-1})$ and $V(o_t)$ respectively, we keep them in the equation to illustrate the formation of attention explicitly. Finally, $m_t \in \mathcal{R}^{M \times d_m}$ is the global latent code produced for the downstream system, and it can be reshaped to match the input of the next layer.

It is worthwhile to discuss design choices made. Because the ordering of the input is arbitrary, each sensory neuron is required to interpret their received signal by observing the relationships between actions made and local system responses. To achieve this, we want $f_k(o_t[i], a_{t-1})$ to have temporal memories. In practice, we find both RNNs and feed-forward neural networks (FNN) with stacked

observations work well, with FNNs being more practical for environments with high dimensional observations. In the attention mechanism, we could have also combined $QW_q \in \mathcal{R}^{M \times d_q}$ as a single learnable parameters matrix, but we separate them for two reasons. First, by factoring into two matrices, we can reduce the number of learnable parameters. Second, we find that instead of making Q learnable, which often leads to repetitive codes along each row of m_t, using the positional encoding proposed in Transformer [78] encourages the attention mechanism to generate distinct codes.

As is depicted in Figure 2, our method can be implemented as a standalone neural network layer, and in an RL agent, we use it as the first layer in the policy network (From hereafter we call it the AttentionNeuron layer). Since this layer's output is permutation invariant to inputs of arbitrary length, the global latent code m_t it produces can be treated as a fixed-length *observation* given to commonly used (non-permutation invariant) policy networks. As we will see later, this design makes our agent robust to changes in the input space, and also allows it to generalize to non-trivial unseen scenarios.

4 Experiments

We conduct experiments on several different RL environments to study various properties of permutation invariant RL agents. Due to the nature of the underlying tasks, we will describe the different architectures of the policy networks used and discuss different training methods. However, the AttentionNeuron layers in all agents are similar, so we first describe the common setups. Hyper-parameters, related parametric dimensions and other details for all experiments are summarized in the Appendix.

For non-vision continuous control tasks, the agent receives an observation vector $o_t \in \mathcal{R}^{|O|}$ at time t. We assign $N = |O|$ sensory neurons for the tasks, each of which sees one element from the vector, hence $o_t[i] \in \mathcal{R}^1, i = 1, \cdots, |O|$. We use an LSTM [42] as our $f_k(o_t[i], a_{t-1})$ to generate Keys, the input size of which is $1 + |A|$ (2 for Cart-Pole and 9 for PyBullet Ant). A simple pass-through function $f(x) = x$ serves as our $f_v(o_t[i])$, and $\sigma(\cdot)$ is $tanh$. For simplicity, we find $W_v = I$ works well for the tasks, so the learnable components are the LSTM, W_q and W_k.

For vision based tasks, we gray-scale and stack $k = 4$ consecutive RGB frames from the environment, and thus our agent observes $o_t \in \mathcal{R}^{H \times W \times k}$. o_t is split into non-overlapping patches of size $P = 6$ using a sliding window, so each sensory neuron observes $o_t[i] \in \mathcal{R}^{6 \times 6 \times k}$. Here, $f_v(o_t[i])$ flattens the data and returns it, hence $V(o_t)$ returns a tensor of shape $N \times d_{f_v} = N \times (6 \times 6 \times 4) = N \times 144$. Due to the high dimensionality for vision tasks, we do not use RNNs for f_k, but instead use a simpler method to process each sensory input. $f_k(o_t[i], a_{t-1})$ takes the difference between consecutive frames ($o_t[i]$), then flattens the result, appends a_{t-1}, and returns the concatenated vector. $K(o_t, a_{t-1})$ thus gives a tensor of shape $N \times d_{f_k} = N \times [(6 \times 6 \times 3) + |A|] = N \times (108 + |A|)$ (111 for CarRacing and 114 for Atari Pong). We use $softmax$ as the non-linear activation function $\sigma(\cdot)$, and we apply layer normalization [3] to both the input patches and the output latent code.

4.1 Cart-pole swing up

We examine Cart-pole swing up [33, 34, 90] to first illustrate our method, and also use it to provide a clear analysis of the attention mechanism. We use `CartPoleSwingUpHarder` [32], a more difficult version of the task where the initial positions and velocities are highly randomized, leading to a higher variance of task scenarios. In the environment, the agent observes $[x, \dot{x}, cos(\theta), sin(\theta), \dot{\theta}]$, outputs a scalar action, and is rewarded at each step for getting x close to 0 and $cos(\theta)$ close to 1.

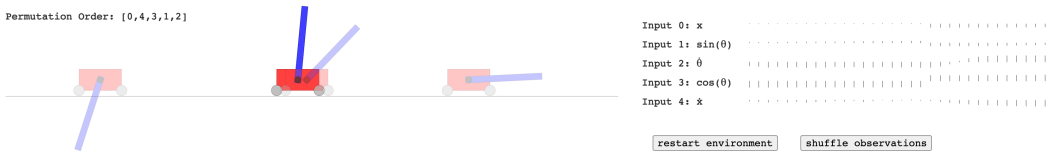

Figure 3: *Interactive demo of CartPoleSwingUpHarder.* In our web demo, the user can shuffle the order of the 5 inputs at any time, and observe how the agent adapts to the new ordering of the inputs.

We use a two-layer neural network as our agent. The first layer is an AttentionNeuron layer with $N = 5$ sensory neurons and outputs $m_t \in \mathcal{R}^{16}$. A linear layer takes m_t as input and outputs a scalar action. For comparison, we also trained an agent with a two-layer FNN policy with 16 hidden units. We use direct policy search to train agents with CMA-ES [41], an evolution strategies (ES) method.

Table 1: *Cart-pole Tests.* For each experiment, we report the average score and the standard deviation from 1000 test episodes. Our agent is trained only in the environment with 5 sensory inputs.

	5 obs	5 obs (shuffled)	10 obs	5 obs + 5 noise
FNN (trained with 5 obs)	593 ± 433	38 ± 120	N/A	N/A
FNN (trained with 10 obs)	N/A	N/A	593 ± 433	137 ± 242
Ours (trained with 5 obs)	472 ± 426	471 ± 426	471 ± 425	461 ± 410

Our agent is able to perform the task and balance the cart-pole from an initially random state. Its average score is slightly lower than the baseline (See column 1 of Table 1) because each sensory neuron requires some time steps in each episode to interpret the sensory input signal it receives. However, as a trade-off for the performance sacrifice, our agent is able to maintain its performance when the input sensor array is randomly shuffled, which is not the case for an FNN policy (column 2). Moreover, although our agent is only trained in an environment with five inputs, it can accept an arbitrary number of inputs in any order without re-training[1]. We test our agent by duplicating the 5 inputs to give the agent 10 observations (column 3). When we replace the 5 extra signals with white noises with $\sigma = 0.1$ (column 4), we do not see a significant drop in performance.

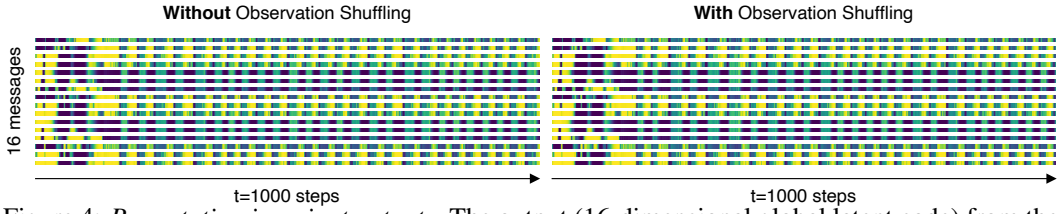

Figure 4: *Permutation invariant outputs.* The output (16-dimensional global latent code) from the AttentionNeuron layer does not change when we input the sensor array as-is (left) or when we randomly shuffle the array (right). Yellow represents higher values, and blue for lower values.

The AttentionNeuron layer should possess 2 properties to achieve these: its output is permutation invariant to its input, and its output carries task-relevant information. Figure 4 is a visual confirmation of the permutation invariant property, whereby we plot the output messages from the layer and their changes over time from two tests. Using same environment seed, we keep the observation as-is in the first test but we shuffle the order in the second. As the figure shows, the output messages are identical in the two roll-outs. We also perform a simple linear regression analysis on the outputs (based on the shuffled inputs) to recover the 5 inputs in their original order. Table 2 shows the R^2 values[2] from this analysis, suggesting that some important indicators (i.e. $\dot{x}$ and $\dot{\theta}$) are well represented in the output.

Table 2: *Linear regression analysis on the output.* For each of the $N = 5$ sensory inputs we have one linear regression model with $m_t \in \mathcal{R}^{16}$ as the explanatory variables.

	x	$\dot{x}$	$cos(\theta)$	$sin(\theta)$	$\dot{\theta}$
R^2	0.354	0.620	0.626	0.233	0.550

Table 3: *PyBullet Ant results.*

	Score
FNN (teacher)	2700 ± 28
FNN (shuffled)	232 ± 112
Ours (ES, shuffled)	2576 ± 75
Ours (BC, shuffled)	2034 ± 948
Ours (BC, shuffled, larger)	2579 ± 457

4.2 PyBullet Ant

While direct policy search methods such as evolution strategies (ES) can train permutation invariant RL agents, often times we already have access to pre-trained agents or recorded human data performing the task at hand. Behavior cloning (BC) can allow us to convert an existing policy to a version that is permutation invariant with desirable properties associated with it.

In Table 3, we train a standard two-layer FNN policy to perform `AntBulletEnv-v0`, a 3D locomotion task in PyBullet [22], and use it as a teacher for BC. For comparison, we also train a two-layer agent with AttentionNeuron for its first layer. Both networks are trained with ES. Similar to CartPole, we expect to see a small performance drop due to some time steps required for the agent to interpret an arbitrarily ordered observation space. We then collect data from the FNN teacher policy to train permutation invariant agents using BC. More details of the BC setup can be found in the Appendix.

[1] Because our agent was not trained with normalization layers, we scaled the output from the AttentionNeuron layer by 0.5 to account for the extra inputs in the last 2 experiments.

[2] R^2 measures the goodness-of-fit of a model. An R^2 of 1 implies that the regression perfectly fits the data.

The performance of the BC agent is lower than the one trained from scratch with ES, despite having the identical architecture. This suggests that the inductive bias that comes with permutation invariance may not match the original teacher network, so the small model used here may not be expressive enough to clone any teacher policy, resulting in a larger variance in performance. A benefit of gradient-based BC, compared to RL, is that we can easily train larger networks to fit the behavioral data. We show that increasing the size of the subsequent layers for BC does increase the performance.

As we will demonstrate next, BC is a useful technique for training permutation invariant agents in environments with high dimensional visual observations that may require larger networks.

4.3 Atari Pong

Here, we are interested in solving screen-shuffled versions of vision-based RL environments, where each observation frame is divided up into a grid of patches, and like a puzzle, the agent must process the patches in a shuffled order to determine a course of action to take. A shuffled version of Atari Pong [10] (See Figure 1, right pair) can be especially hard for humans to play when inductive biases from human priors [28] that expect a certain type of spatial structure is missing from the observations.

But rather than throwing away the spatial structure entirely from our solution, we find that convolution neural network (CNN) policies work better than fully connected multi-layer perceptron (MLP) policies when trained with behavior cloning for Atari Pong. In this experiment, we reshape the output m_t of the AttentionNeuron layer from $\mathcal{R}^{400\times32}$ to $\mathcal{R}^{20\times20\times32}$, a 2D grid of latent codes, and pass this 2D grid into a CNN policy. This way, the role of the AttentionNeuron layer is to take a list of unordered observation patches, and learn to construct a 2D grid representation of the inputs to be used by a downstream policy that expects some form of spatial structure in the codes. Our permutation invariant policy trained with BC is able to consistently reach a perfect score of 21, even with shuffled screens. The details of the CNN policy and BC training can be found in the Appendix.

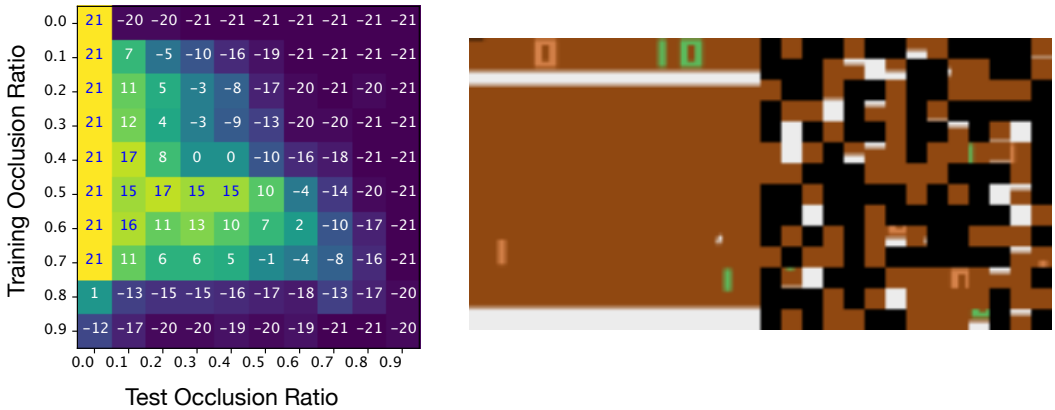

Figure 5: *Mean test scores in Atari Pong, and example of a randomly-shuffled occluded observation.* In the heat map, each value is the average score from 100 test episodes. For comparison, we show the original screen (left), and the agent's observation (right). Discarded patches shown here in black.

Unlike typical CNN policies, our agent can accept a subset of the screen, since the agent's input is a variable-length list of patches. It would thus be interesting to deliberately randomly discard a certain percentage of the patches and see how the agent reacts. The net effect of this experiment for humans is similar to being asked to play a partially occluded and shuffled version of Atari Pong (see Figure 5, right). During training via BC, we randomly remove a percentage of observation patches. In tests, we fix the randomly selected positions of patches to discard during an entire episode.

We present the results in a heat map in Figure 5 (left), where the y-axis shows the patches removed during training and the x-axis gives the patch occlusion ratio in tests. The diagram shows clear patterns for interpretation. Looking horizontally along each row, the performance drops because the agent sees less of the screen which increases the difficulty. Interestingly, an agent trained at a high occlusion rate of 80% rarely wins against the Atari opponent, but once it is presented with the full set of patches during tests, it is able to achieve a fair result by making use of the additional information.

4.4 CarRacing

We find that encouraging an agent to learn a coherent representation of a deliberately shuffled visual scene leads to agents with useful generalization properties. Such agents are still able to perform their task even if the visual background of the environment changes, despite being trained only on a single static background. Out-of-domain generalization is an active area, and here, we combine our method with AttentionAgent [76], a method that uses selective, hard-attention via a patch voting mechanism. AttentionAgents in [76] generalize well to several unseen visual environments where task irrelevant elements are modified, but fails to generalize to drastic background changes in a zero-shot setting.

In this experiment, we combine the permutation invariant AttentionNeuron layer with the policy network used in AttentionAgent. As their hard-attention-based policy is non-differentiable, we train the entire system using ES. We reshape the AttentionNeuron layer's outputs to adapt for the policy network. Specifically, we reshape the output message to $m_t \in \mathcal{R}^{32\times32\times16}$ such that it can be viewed as a 32-by-32 grid of 16 channels. The end result is a policy with two layers of attention: the first layer outputs a latent code book to represent a shuffled scene, and the second layer performs hard attention to select the top $K = 10$ codes from a 2D global latent code book. A detailed description of the selective hard attention policy from [76] and other training details can be found in the Appendix.

We first train the agent in the CarRacing [48] environment, and report the average score from 100 test roll-outs in Table 4. As the first column shows, our agent's performance in the training environment is slightly lower but comparable to the baseline method, as expected. But because our agent accepts randomly shuffled inputs, it is still able to navigate even when the patches are shuffled. Figure 1 (left pair) gives an illustration, where the right screen is what our agent observes and the left is for human visualization. A human will find driving with the shuffled observation to be very difficult because we are not constantly exposed to such tasks, just like in the "reverse bicycle" example mentioned earlier.

Table 4: CarRacing Test Results.

	Training Env	KOF	Mt. Fuji	Ukiyoe	DS
AttentionAgent [76]	901 ± 54	-81 ± 4	-57 ± 38	-107 ± 50	-56 ± 23
Ours	801 ± 147	646 ± 189	503 ± 152	661 ± 140	171 ± 146

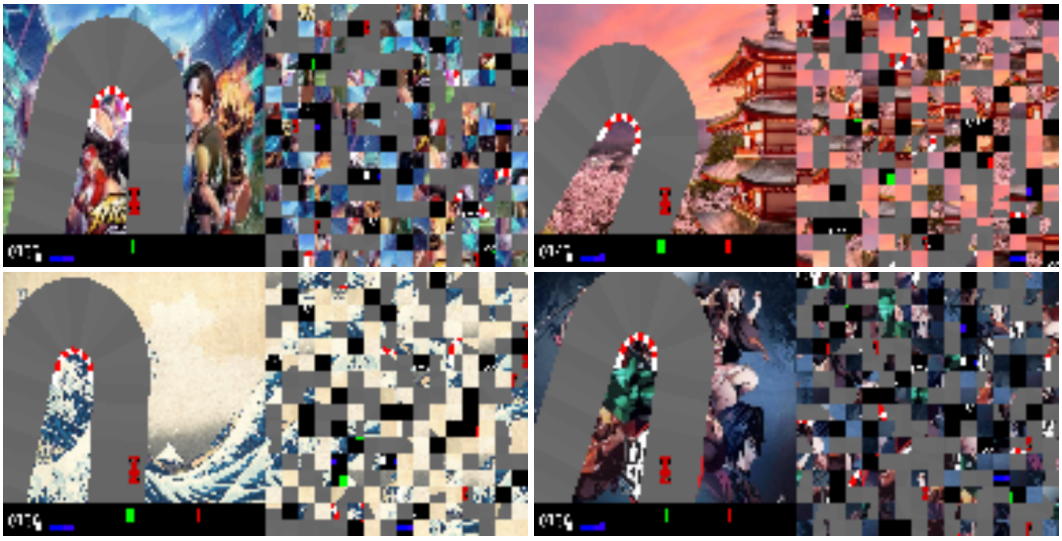

Figure 6: *Screenshots of test environments.* In each pair of images, the left is for human visualization and the right is what our agent sees. From the top left and in the clockwise order, the environments are "KOF", "Mt. Fuji", "DS" and "Ukiyoe".

Without additional training or fine-tuning, we test whether the agent can also navigate in four modified environments where the green grass background is replaced with various images (See Figure 6). As Table 4 (from column 2) shows, our agent generalizes well to most of the test environments with only mild performance drops while the baseline method fails to generalize. We suspect this is because the AttentionNeuron layer has transformed the original RGB space to a useful hidden representation (represented by m_t) that has eliminated task irrelevant information after observing and reasoning about the sequences of (o_t, a_{t-1}) during training, enabling the downstream hard attention policy to work with an optimized abstract representation tailored for the policy, instead of raw RGB patches.

5 Discussion and Future Work

In this work, we investigate the properties of RL agents that can treat their observations as an arbitrarily ordered, variable-length list of sensory inputs. By processing each input stream independently, and consolidating the processed information using attention, our agents can still perform their tasks even if the ordering of the observations is randomly permuted several times during an episode, without explicitly trained for frequent re-shuffling (See Table 5).

Table 5: *Reshuffle observations during a roll-out.* In each test episode, we reshuffle the observations every t steps. For CartPole, we test for 1000 episodes because of its larger task variance. For the other tasks, we report mean and standard deviation from 100 tests. All environments except for Atari Pong have a hard limit of 1000 time steps per episode. In Atari Pong, while the maximum length of an episode does not exist, we observed that an episode usually lasts for around 2500 time steps.

	CartPole	PyBullet Ant	Atari Pong	CarRacing
$t = 25$	107 ± 146	2053 ± 225	-20 ± 1	732 ± 161
$t = 50$	163 ± 198	2319 ± 188	-20 ± 2	772 ± 163
$t = 100$	242 ± 254	2406 ± 178	-10 ± 12	768 ± 167
$t = 200$	318 ± 310	2493 ± 105	-2 ± 17	774 ± 182
$t = 500$	407 ± 380	2548 ± 87	18 ± 9	805 ± 158
No reshuffle	472 ± 426	2576 ± 75	21 ± 0	801 ± 147

Applications By presenting the agent with shuffled, and even incomplete observations, we encourage it to interpret the meaning of each local sensory input and how they relate to the global context. This could be useful in many real world applications. For example, such policies could avoid errors due to cross-wiring or complex, dynamic input-output mappings when being deployed in real robots. A similar setup to the CartPole experiment with extra noisy channels could enable a system that receives thousands of noisy input channels to identify the small subset of channels with relevant information.

Limitations For visual environments, patch size selection will affect both performance and computing complexity. We find that patches of 6x6 pixels work well for our tasks, as did 4x4 pixels to some extent, but single pixel observations fail to work. Small patch sizes also results in a large attention matrix which may be too costly to compute, unless approximations are used [18, 81, 84].

Another limitation is that the permutation invariant property apply only to the inputs, and not to the outputs. While the ordering of the observations can be shuffled, the ordering of the actions cannot. For permutation invariant outputs to work, each action will require feedback from the environment, including reward information, in order to learn the relationship between itself and the environment.

Future Work An interesting future direction is to also make the action layer have the same properties, and model each "motor neuron" as a module connected using attention. With such methods, it may be possible to train an agent with an arbitrary number of legs, or control robots with different morphology using a single policy that is also provided with a reward signal as feedback. It is exciting to see future works that include signals such as environmental rewards to train permutation invariant meta-learning agents that can adapt to not only changes in the observed environment, but also to changes to itself.

6 Societal Impact

Like most algorithms proposed in computer science and machine learning, our method can be applied in ways that will have potentially positive or negative impacts to society. While our small-scale, self-contained experiments study only the properties of RL agents that are permutation invariant to their observations, and we believe our results do not directly cause harm to society, the robustness and flexible properties of the method may be of use for data-collection systems that receive data from a large variable number of sensors. For instance, one could apply permutation invariant sensory systems to process data from millions of sensors for anomaly detection, which may result in both positive or negative impacts, if used in applications such as large-scale sensor analysis for weather forecasting, or deployed in large-scale surveillance systems that could undermine our basic freedoms.

Our work also provides a way to view the Transformer [78] through the lens of cellular automata and self-organizing neural networks. Transformers are known to have potentially negative societal impacts highlighted in studies about possible data-leakage and privacy vulnerabilities [12], malicious misuse and issues concerning bias and fairness [8], and energy requirements for training them [73].

References

[1] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable mpc for end-to-end planning and control. *arXiv preprint arXiv:1810.13400*, 2018.

[2] J. Ba, G. Hinton, V. Mnih, J. Z. Leibo, and C. Ionescu. Using fast weights to attend to the recent past. *arXiv preprint arXiv:1610.06258*, 2016.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] B. J. Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

[5] B. J. Baars. Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. *Progress in brain research*, 150:45–53, 2005.

[6] P. Bach-y Rita, C. C. Collins, F. A. Saunders, B. White, and L. Scadden. Vision substitution by tactile image projection. *Nature*, 221(5184):963–964, 1969.

[7] P. Bach-y Rita and S. W. Kercel. Sensory substitution and the human–machine interface. *Trends in cognitive sciences*, 7(12):541–546, 2003.

[8] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[9] D. Beniaguev, I. Segev, and M. London. Single cortical neurons as deep artificial neural networks. *bioRxiv*, page 613141, 2020.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[12] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*, 2020.

[13] R. D. Cazé, M. Humphries, and B. Gutkin. Passive dendrites enable single neurons to compute linearly non-separable functions. *PLoS Comput Biol*, 9(2):e1002867, 2013.

[14] M. Chang, S. Kaushik, S. M. Weinberg, T. Griffiths, and S. Levine. Decentralized reinforcement learning: Global decision-making via local economic transactions. In *International Conference on Machine Learning*, pages 1437–1447. PMLR, 2020.

[15] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23, 2014.

[16] J. Choi, B.-J. Lee, and B.-T. Zhang. Multi-focus attention network for efficient deep reinforcement learning. *arXiv preprint arXiv:1712.04603*, 2017.

[17] B. Chopard and M. Droz. *Cellular automata*, volume 1. Springer, 1998.

[18] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[19] L. O. Chua and L. Yang. Cellular neural networks: Theory. *IEEE Transactions on circuits and systems*, 35(10):1257–1272, 1988.

[20] E. F. Codd. *Cellular automata*. Academic press, 1968.

[21] J. Conway. The game of life. *Scientific American*, 223(4):4, 1970.

[22] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2020.

[23] S. Dehaene. *Consciousness and the brain: Deciphering how the brain codes our thoughts*. Penguin, 2014.

[24] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[28] R. Dubey, P. Agrawal, D. Pathak, T. L. Griffiths, and A. A. Efros. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217*, 2018.

[29] D. Eagleman. *Livewired: The inside story of the ever-changing brain*. Canongate Books, 2020.

[30] P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020.

[31] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch. Som-vae: Interpretable discrete representation learning on time series. *arXiv preprint arXiv:1806.02199*, 2018.

[32] D. Freeman, D. Ha, and L. Metz. Learning to predict without looking ahead: World models without forward prediction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. https://learningtopredict.github.io.

[33] A. Gaier and D. Ha. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. https://weightagnostic.github.io.

[34] Y. Gal, R. McAllister, and C. E. Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, Apr. 2016.

[35] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.

[36] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021.

[37] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[38] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[39] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018. https://worldmodels.github.io.

[40] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[41] N. Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

[42] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[43] S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[44] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4455–4464. PMLR, 2020.

[45] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021.

[46] C. Joshi. Transformers are graph neural networks. *The Gradient*, 2020.

[47] L. Kirsch and J. Schmidhuber. Meta learning backpropagation and improving it. *arXiv preprint arXiv:2012.14905*, 2020.

[48] O. Klimov. Carracing-v0, 2016.

[49] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 09–15 Jun 2019.

[50] I.-J. Liu, R. A. Yeh, and A. G. Schwing. Pic: permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*, pages 590–602. PMLR, 2020.

[51] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[52] T. Miconi, A. Rawal, J. Clune, and K. O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *arXiv preprint arXiv:2002.10585*, 2020.

[53] T. Miconi, K. Stanley, and J. Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

[54] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[55] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin. Growing neural cellular automata. *Distill*, 2020. https://distill.pub/2020/growing-ca.

[56] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende. Towards interpretable reinforcement learning using attention augmented agents. *arXiv preprint arXiv:1906.02500*, 2019.

[57] E. Najarro and S. Risi. Meta-learning through hebbian plasticity in random networks. *arXiv preprint arXiv:2007.02686*, 2020.

[58] J. Neumann, A. W. Burks, et al. *Theory of self-reproducing automata*, volume 1102024. University of Illinois press Urbana, 1966.

[59] S. Ohsawa, K. Akuzawa, T. Matsushima, G. Bezerra, Y. Iwasawa, H. Kajino, S. Takenaka, and Y. Matsuo. Neuron as an agent, 2018.

[60] J. Ott. Giving up control: Neurons as reinforcement learning agents. *arXiv preprint arXiv:2003.11642*, 2020.

[61] P. Poirazi, T. Brannon, and B. W. Mel. Pyramidal neuron as two-layer neural network. *Neuron*, 37(6):989–999, 2003.

[62] E. Randazzo, A. Mordvintsev, E. Niklasson, M. Levin, and S. Greydanus. Self-classifying mnist digits. *Distill*, 2020. https://distill.pub/2020/selforg/mnist.

[63] L. Renier, A. G. De Volder, and J. P. Rauschecker. Cortical plasticity and preserved function in early blindness. *Neuroscience & Biobehavioral Reviews*, 41:53–63, 2014.

[64] L. A. Renier, I. Anurova, A. G. De Volder, S. Carlson, J. VanMeter, and J. P. Rauschecker. Preserved functional specialization for spatial processing in the middle occipital gyrus of the early blind. *Neuron*, 68(1):138–148, 2010.

[65] P. Sachse, U. Beermann, M. Martini, T. Maran, M. Domeier, and M. R. Furtner. "the world is upside down"–the innsbruck goggle experiments of theodor erismann (1883–1961) and ivo kohler (1915–1985). *Cortex*, 92:222–232, 2017.

[66] M. Sandler, M. Vladymyrov, A. Zhmoginov, N. Miller, A. Jackson, T. Madams, et al. Meta-learning bidirectional update rules. *arXiv preprint arXiv:2104.04657*, 2021.

[67] D. Sandlin. The backwards brain bicycle: Un-doing understanding, 2019.

[68] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[69] J. Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *International Conference on Artificial Neural Networks*, pages 460–463. Springer, 1993.

[70] J. Schmidhuber. A 'self-referential'weight matrix. In *International Conference on Artificial Neural Networks*, pages 446–450. Springer, 1993.

[71] J. Sharma, A. Angelucci, and M. Sur. Induction of visual orientation modules in auditory cortex. *Nature*, 404(6780):841–847, 2000.

[72] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

[73] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[74] S. Sudhakaran, D. Grbic, S. Li, A. Katona, E. Najarro, C. Glanois, and S. Risi. Growing 3d artefacts and functional machines with neural cellular automata. *arXiv preprint arXiv:2103.08737*, 2021.

[75] C. Sun, F. Baradel, K. Murphy, and C. Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*, 2019.

[76] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020. https://attentionagent.github.io.

[77] R. VanRullen and R. Kanai. Deep learning and the global workspace theory. *Trends in Neurosciences*, 2021.

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[79] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[80] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[81] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[82] S. Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419–424, 1984.

[83] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[84] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *arXiv preprint arXiv:2102.03902*, 2021.

[85] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. *arXiv preprint arXiv:1911.06455*, 2019.

[86] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

[87] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, M. Shanahan, V. Langston, R. Pascanu, M. Botvinick, O. Vinyals, and P. Battaglia. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019.

[88] D. Zhang, C. Choi, J. Kim, and Y. M. Kim. Learning to generate 3d shapes with generative cellular automata. *arXiv preprint arXiv:2103.04130*, 2021.

[89] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

[90] X. Zuo. Pytorch implementation of improving pilco with bayesian neural network dynamics models, 2018. https://github.com/zuoxingdong/DeepPILCO.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] . See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] . See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] **(Not yet, but we plan to.)** Upon publication, we will release the code and instructions needed to not only reproduce the results, but guide further exploration of this area, in a GitHub repo.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . See Section A.1 of the Supplementary Material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . See Section A.2 of the Supplementary Material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] . We have cited the RL environments used in this project, by referencing the paper that have introduced the environment (which would have details about the open-source implementation details), or the GitHub repo if a paper is not available.
   (b) Did you mention the license of the assets? [No] . The license of the RL environments can be found on their original GitHub repos. Upon release of our code, we will also reiterate the original license (and copyright information) in any part of our code that have been derived from the original environments.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] . We included an interactive demo of one of the experiments, and also released videos of experimental results. The URL is mentioned in the abstract of this paper: `https://neuripsanon.github.io/results/`
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]