A Neurobiologically-inspired Deep Learning Framework for Autonomous Context Learning

David W. Ludwig II
Department of Computer Science
Middle Tennessee State University
Murfreesboro, TN, USA
dwl2x@mtmail.mtsu.edu

Lucas W. Remedios
Department of Computer Science
Middle Tennessee State University
Murfreesboro, TN, USA
lwr2k@mtmail.mtsu.edu

Joshua L. Phillips
Department of Computer Science
Middle Tennessee State University
Murfreesboro, TN, USA
Joshua.Phillips@mtsu.edu

***Abstract*—Neurobiologically-inspired working memory models demonstrate human/animal capabilities to rapidly adapt and alter responses to the environment via context-switching and error monitoring. However, the application of these models outside of reinforcement learning problems has been relatively unexplored. We present a new framework compatible with Tensorflow/Keras enabling the integration of working memory-inspired mechanisms into typical neural network architectures. These mechanisms allow models to autonomously learn multiple tasks, statically or dynamically allocated. We also examine the generalization of the framework across a variety of multi-context supervised learning and reinforcement learning tasks. The resulting experiments successfully integrate these mechanisms with multi-layer and convolutional neural network architectures and the diversity of problems solved demonstrates the framework's generalizability across a variety of architectures and tasks.**

***Index Terms*—cognitive architectures, machine learning, task-switching, multiobjective, working memory**

I. Introduction

As humans, we are particularly good at reacting and adapting to new situations in the environment. Not only can we quickly learn from our actions, we can also recognize changes in the environment when something unexpected occurs. In a situation that you are unfamiliar with, you would likely assume your actions and decisions are incorrect when the expected outcome is not observed. When an environment is more familiar and you do not observe the outcome that you expected (e.g. turning on a TV, flipping a light switch), you are more quickly to conclude that something in the environment has changed, rather than your actions being incorrect. Our ability to rapidly adapt and change our behavior based on the context of a situation is in major part due to one of our brain's executive functions known as *working memory*.

Working memory is a form of short-term memory that plays a critical role in influencing our decision-making process. Instead of requiring the complete unlearning/relearning of a task, working memory facilitates the ability to rapidly and dynamically alter our responses to changes in the current situation. While there are many models that can accurately demonstrate and explain the mechanisms involved with working memory [1]–[10], they are not currently utilized much outside of the realm of working memory research. Furthermore, while mechanisms for including contextual information in traditional neural networks have been explored before [11], current solutions often require the inclusion of additional contextual input and prohibit autonomy. The incorporation of the context-driven learning/switching mechanisms from working memory models could allow a typical neural network to autonomously detect and respond to changes in the expected output without providing any explicit context or assistance.

In this paper, we present a general-use framework for implementing the mechanisms inspired by working memory models into common neural network architectures. As this framework is built around Tensorflow Keras, the new framework components can be added to existing model architectures with minimal adjustment. Along with the design and development of the mechanisms and components, we also aim to demonstrate the framework's generalizability by applying it across a variety of problems in both supervised learning and reinforcement learning settings. These experiments include the utilization of multilayer neural networks, convolutional neural networks (CNNs) and reinforcement learning using both 1-step and n-step Q-learning.

II. Background

A. Working Memory

The mechanisms involved in the definition and separation of contextual information stems from the current understanding of working memory and its neurobiological basis in the mesolimbic dopamine system (MDS) and prefrontal cortex (PFC). One study by O'Reilly, Noelle, Braver, *et al.* [1] provides much insight into these mechanisms and our ability to rapidly update goals and focus on particular tasks. Working memory is an activation-based memory, meaning that memory is retained through persistent neural firing rather than weight-based updates. Learning through weight-based updates is significantly slower and results in perseveration as previously learned information must be unlearned or relearned if task demands have changed. Through activation-based memory and association of contextual information with these neural states, the information contained in working memory can be rapidly updated. It is believed that the PFC maintains a memory representation of the targeted dimension or feature, providing a form of rule-like "top-down" support, or "biasing," to influence the perceptual processing and the selection of actions [1].

While the PFC retains the memory representations, the updates to working memory are handled via the MDS. It has been shown that the contents of working memory are updated when neurotransmitter dopamine levels are phasically elevated [4], [5], [7], [8], [12]. Because they are responsible for broadcasting dopamine signals [13], the MDS functions as a form of gating mechanism that can update or protect the memory representations currently stored in the PFC [4]–[6], [8], [12], [14]. When an expected reward is not delivered, updates to working memory are triggered via a negative error signal. Learning in the presence of these large negative error signals is believed to be limited since the contextual information was likely incorrect [2], [12].

B. Holographic Reduced Representations

Some more recent models have abstracted away some of the neural details in the above models [9], [10]; in particular, activation-based recurrent layers for handling PFC representations are instead encoded using holographic reduced representations (HRRs) [15]–[17]. An HRR is a fixed-width vector that is created by generating real-values from a Gaussian distribution [18]. Any abstract concept can be represented by a single HRR vector; for example, the color blue, a ball, etc.. These HRRs can be combined through the use of circular convolutions, resulting in a new HRR that represents the combination of the overall concept. This new HRR is highly likely to be orthogonal to each of the source HRRs without any increase in dimensionality. While the HRRs themselves are primarily used to automate the distributed encodings of conceptual information, the orthogonality of the resulting circular convolutions allow an artificial neural network to easily distinguish and separate the values of actions based on context.

The computation of the circular convolution, typically denoted as $\circledast$, can be performed on any two vectors of the same length, n, with a time complexity of $O(n \log n)$ through the use of Fast-Fourier transforms (FFTs). The circular convolution is defined by the equation:

$$c \circledast x \equiv f^{-1}(f(c) \odot f(x)) \tag{1}$$

where f is a discrete Fourier transform, f^{-1} is the inverse discrete Fourier transform, and $\odot$ is the component-wise multiplication of two vectors [18].

C. n-task Learning

Inspired by the models above, n-task learning (nTL) is an approach to autonomous multi-task learning with reinforcement learning problems in mind [19]. Analogous to rule-like representations in the PFC, the model implements task contexts in the form of abstract task representations (ATRs), each of which internally is a unique HRR. By also encoding the states and actions each with a unique HRR, state/action pairs can be contextualized by convolving them with the active context ATR. The resulting HRR can then be given to the model as input, allowing it to discretely learn and predict the value of an action during a particular state and context. The expected reward is also modeled internally for each of the available contexts using standard temporal-difference (TD) learning. When the TD-error surpasses a certain threshold, it indicates the reward received by the model was unexpected, thus triggering a context-switch, and the next context ATR is used. These ATRs may be allocated statically, or dynamically by monitoring the average expected reward across all tasks. If the average expected reward across all contexts falls below a certain threshold, a new context may be added, however the model's weights will be re-initialized.

D. Autonomous Context Learning

The research above demonstrates how working memory models have been used to explain the neural basis of context-switching behavior [1]–[10], [19]. In order to fully autonomize the selection and utilization of contexts, a neural network model requires several internal mechanisms. First is the ability to maintain a form of context for each of the possible functions. The number of contexts maintained could be specified explicitly upon model creation or dynamically allocated as needed. The model must then be capable of applying contextualization information to the inputs using these maintained contexts. Finally, the model would require self-monitoring mechanisms to determine and switch to the appropriate contexts by analyzing the provided feedback used to adjust the target value to properly adapt to the task.

While the models described here are capable of autonomous context learning, there are certain aspects of their implementation that can be quite limiting. For example, the utilization of these mechanisms outside of the reinforcement learning domain has been relatively unexplored. Though the working memory models described above have successfully demonstrated the mechanisms required for context learning/switching, they are rarely utilized outside the realm of working memory research. nTL provides a robust framework for solving a broad set of reinforcement learning problems through the integration of context learning/switching mechanisms. While successful, the approach used to model the expected reward makes learning temporally-extended reinforcement learning tasks rather difficult. Furthermore, nTL also requires weight re-initialization when dynamically allocating task contexts. We aim to address these issues by creating a new general-use deep learning framework. The framework should enable integration of context switching mechanisms inspired by the working memory models mentioned previously into common/existing neural network architectures with minimal modifications required. Additionally, the new components in the framework should be compatible with more learning approaches outside the realm of reinforcement learning.

III. Methods

A. Framework Design & Implementation

The framework introduces various new components that are compatible with Keras/Tensorflow and require only minor modifications to standard training loops to execute. These modifications are described with each of the corresponding

components below. The source code for the framework and experiments is publicly at https://github.com/DLii-Research/context-learning.

1) Context Layer and Abstract Task Representations: One of the fundamental aspects of the framework is its ability to utilize ATRs to differentiate data based on an active context. While there are different potential methods for generating these ATRs [19], the framework continues to implement these ATRs as HRR vectors inline with the working memory models mentioned previously. Each ATR is a unique random unitary HRR, and each ATR represents a single context. With a neural network, the input/activation vector can be convolved with the active context's ATR and fed forward through the network. The circular convolution results in a new vector that is roughly orthogonal to each of the original vectors, enabling the model to separate learning of the individual tasks and reduce potential catastrophic interference effects.

To integrate ATRs into the network, we introduce the concept of a *context layer*. This layer is analogous to PFC layers in the previously described working memory models and the ATRs are analogous to PFC stripes within the PFC layers which encode specific contextual activation patterns. The context layer maintains a unique ATR for each of the contexts, convolving the previous layer's output with the active context's ATR to produce the new output. Inserting this layer into a neural network is enough to grant the network the ability to learn and predict under different contexts without providing any additional input. As the ATRs are HRRs internally, the layer contains no trainable weights, and thus it depends on a subsequent downstream layer to properly handle the error-monitoring context-switching mechanisms. The gradients generated at the subsequent layer are used to compute the weight deltas to be used for calculating the *context loss* as described in the next subsection.

2) Context Switching Mechanisms: In order for the context layer to switch contexts autonomously, we developed a loss-based switching mechanism inspired by the working memory models and the mesolimbic dopamine system mentioned previously. This mechanism is integrated as a module within the context layer and invoked from the training/evaluation loops. To reliably switch contexts, it is important to first determine the loss contributed strictly by the given context layer. Using the computed backpropogation gradients at the subsequent layer, we can calculate the deltas of the context layer's weights to monitor the error caused by the layer. By accumulating the mean-squared deltas over each batch of a training epoch, we can obtain a single positive value, ΔC that represents the overall error/loss for the current context. If the subsequent layer contains bias weights, the gradients for these weights may be used to compute the context deltas to reduce the time complexity of the calculation. Otherwise the mean-squared deltas/gradients for the non-bias weights are used. In order for the model to assess its performance on a task under a particular context, the expected loss is maintained internally for each context using a TD-learning approach with the following:

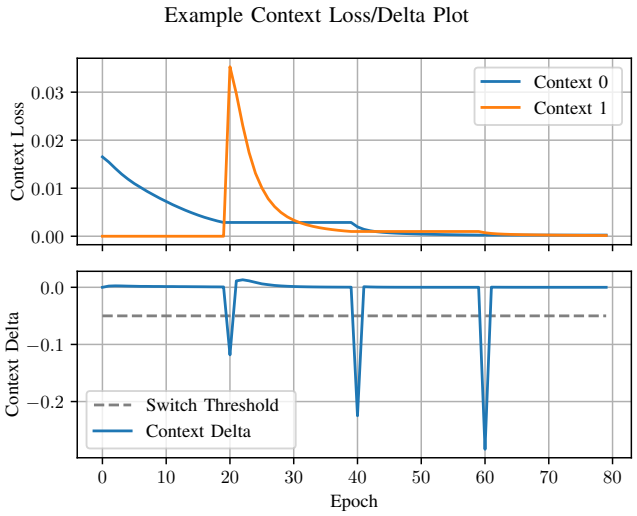


Fig. 1: The top plot represents the modeled context loss ($A(atr)$), while the bottom plot represents the computed context delta (δ) under the active context ATR. The task begins with context 0, and the context delta is initialized to the first loss observed. When the task is switched, a large context delta is produced. When the context delta falls below the threshold, the model switches to the next context. The cycle repeats as the task is switched every 20 epochs.

$$A(atr) \leftarrow A(atr) + \alpha_A[\Delta C - A(atr)] \tag{2}$$

where atr is the context ATR and α_A is a learning rate parameter. During early training, the values of ΔC are not so important since large context loss is expected; but as the network learns the function, this loss approaches zero and the values of ΔC become more meaningful. To determine when a context switch should occur, the context delta needs to be computed by taking the difference between the expected context loss and the current context loss as shown in the equation:

$$\delta = A(atr) - \Delta C \tag{3}$$

At the end of the training/evaluation epoch, the computed value of δ can be directly compared against a threshold value t_{switch} (in the range $(-\infty, 0]$). If δ exceeds this threshold, the active context is swapped with the next-in-line and, in compliance with the working memory models, the value of $A(atr)$ is not updated. In a similar fashion, contexts can also be added dynamically when $\max_{atr} A(atr)$ exceeds a secondary threshold t_{add}. These t_{switch} and t_{add} thresholds are currently implemented as static hyperparameters and must be adjusted based on task performance similarly to learning rate and other hyperparameters. Figure 1. provides an example scenario of the modeled context loss over time for a domain with two alternating contexts that need to be learned.

For some epochs, it is possible that no context is appropriate for use in the current task (i.e. the δ values generated under

every context exceed the threshold). While training under one context, the manipulation of the weights may interfere with the other contexts, resulting in an unexpected increase in context loss. This loss may be significant enough to trigger an unwanted context switch, or even an infinite context switch loop if δ exceeds the threshold in all known contexts. To combat this, sequential context switches are counted. If all contexts are attempted but all exceed the value of t_{switch}, the context with the lowest ΔC value is chosen and the value of $A(atr)$ is re-initialized.

3) Model & Training Regime: The switching mechanisms above necessitate some modifications to how weight updates are handled in the model, and so a custom Keras model is created which we call the `ContextModel`. As context-switches occur at the end of an epoch, it's possible that the weights may be updated multiple times depending on the batch size. In this case, if a context-switch happens to occur, the weights in the network would have been updated based on an inappropriate context, conflicting with previous working memory research. This is currently solved by maintaining a backup of the weights at the beginning of each epoch. If a context switch happens to occur during a training epoch, the weights are restored to prevent learning under the incorrect context in compliance with existing working memory models and the epoch may be repeated under the new context.

Immediately after a context switch, the repetition of the epoch may be desired. In supervised learning problems the model should locate and learn under the best fitting context. Since the context switch is decided at the end of the epoch, all training from that epoch could potentially be reverted. For supervised learning problems, it's important to repeat the epoch if a context switch occurs so the model can learn appropriately without wasting any information. However, in reinforcement learning settings, the repetition of epochs may be unwanted as updates to the model are based on previous states that were potentially evaluated under a different context. To handle these situations, the training loop includes the Boolean parameter, $retry\_fit$, that indicates whether an epoch should be repeated upon a context switch.

For temporally-extended reinforcement learning tasks where an agent is required to pass through non-terminal states (typically along the way to reaching a terminal state, such as a goal state for the task), it is important to distinguish these steps to ensure that the model can correctly track sequential switches in a given episode. During these intermediate steps, the sequential switch counter in the switching mechanisms must be retained to allow handling dynamic context allocation and locating best-fit contexts after all contexts have been attempted. In temporally-extended tasks, the terminal states function as special states in which the reward is absorbed whereas the non-terminal states observe a discounted reward. Since this behavior is programmed explicitly in the task, the $absorb$ parameter has been included in the framework's training loop to indicate to the model which states are terminal/non-terminal.

It is also important to provide a reliable means of initializing the values of $A(atr)$. During the model's first experience with a context, the value of $A(atr)$ for that context is uninitialized, and thus the switching mechanisms are inactive. By default, the value of $A(atr)$ for a particular context is initialized to the value of ΔC of the first training epoch. However, for some tasks, initializing to the first-observed ΔC value may be inaccurate as there are other potential input combinations/values that could produce a higher calculated loss. To deal with these situations, the framework includes the ability to provide user-specified initial $A(atr)$ values, the ability to modify the observed ΔC, or by temporarily disabling the switching mechanisms for a number of epochs to allow the model to first gain experience before attempting to recognize changes in the task.

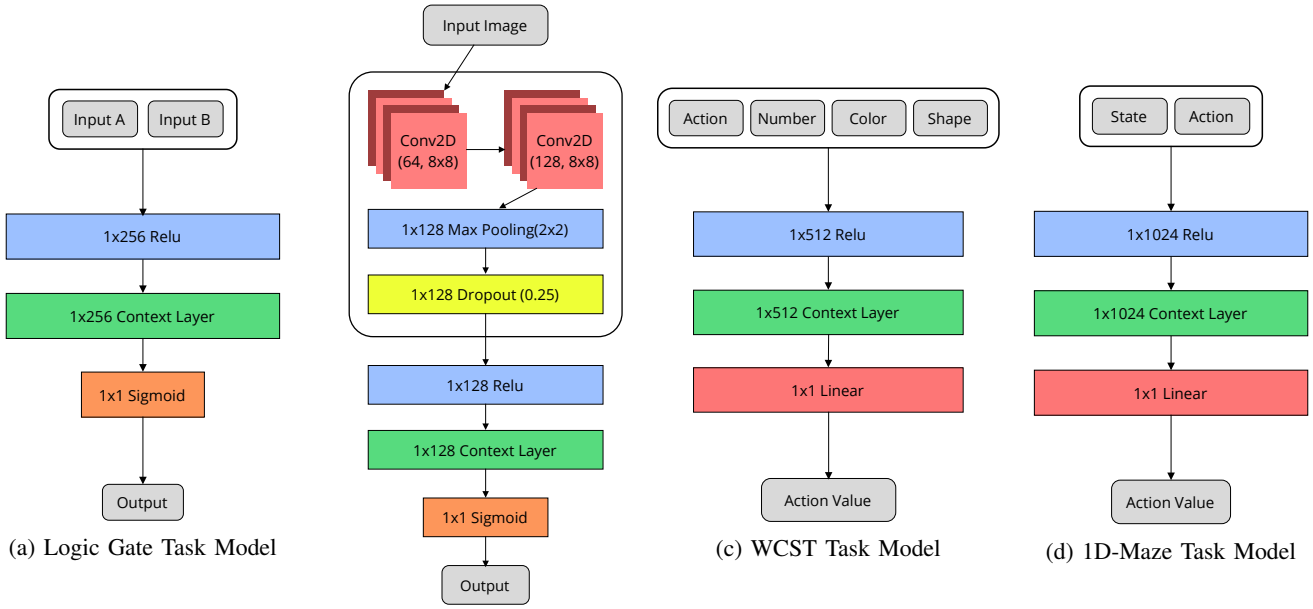
Lastly, for supervised learning problems, the framework supports training a model over multiple different contextual datasets uniformly. The framework trains the model for a specified number of epochs on each of the datasets. The complete training period over all of the provided datasets is considered a **training cycle**. In addition to specifying the number of training epochs, the number of training cycles may also be specified, and the order of the datasets may be shuffled at the beginning of each new cycle.

B. Experimentation & Evaluation

We evaluate this framework over a variety of tasks consisting of both supervised learning and reinforcement learning settings. These tasks serve not only to provide support for successful learning with the framework, but also demonstrate the overall capability and generalizability across the variety of different tasks. While the framework supports dynamic allocation of contexts, the evaluation presented below focuses on static context allocation.

It is also important to note that due to the inclusion of error-monitoring mechanisms within this framework, it is difficult to make any fair comparisons with many other models/frameworks. While our controls focus on varying the number of available ATRs, we cite the controls provided by Jovanovich and Phillips [19] as they are solving the same types of problems described here, including the *Wisconsin card sorting test*. LSTM's were chosen as the state-of-the-art model for one of their controls. They were able to demonstrate and conclude that LSTM's were unable to learn these types of tasks. This result was not surprising as error-monitoring mechanisms are not present in standard deep-learning approaches but are critical components of task-switching models in the literature.

1) Logic Gate Task: This supervised-learning task requires a single model to autonomously learn and predict multiple different logic gates without any provided contextual-information regarding the active logic gate. The model will be presented a training batch containing two binary inputs and the expected output for a particular logic gate. After some time, the logic gate will be swapped for a different gate, modifying the model's expected output. The model should recognize this and switch contexts to avoid overwriting it's prior knowledge.



(a) Logic Gate Task Model

(b) MNIST Divisibility Task Model

(c) WCST Task Model

(d) 1D-Maze Task Model

Fig. 2: The neural network model architectures for each of the tasks. Since the models were implemented using Tensorflow Keras, layers such as Relu, Sigmoid, and Linear are implemented as dense layers with the corresponding activation function.

This experiment utilizes the neural network architecture described in Figure 2a to model a total of eight different logic gates. These gates are comprised of six common gates: *AND, OR, NAND, NOR, XOR, XNOR*; along with two additional custom gates that simply forward one of the given input values. Each logic gate is separated into its own dataset containing all possible pairs of bipolar inputs (-1 is used in-place of 0) along with the expected output value. The order of the logic gate datasets is randomized during each training cycle, and the active dataset is shuffled during each training epoch. The model is trained with a batch size of 1 and updated via stochastic gradient descent (SGD) and binary-crossentropy loss. The hyperparameters for this experiment are described in Table I.

TABLE I:
Logic Gate Task Training Parameters

Name	Value	Description
k	4	Number of training cycles
n	50	Number of training epochs per cycle
α	0.1	Learning rate for optimizer
α_A	0.5	Learning rate for $A(atr)$
t_{switch}	-0.02	Context-switch threshold

2) MNIST Divisibility Task: This next task aims to provide a proof-of-concept, demonstrating the compatibility of the framework with CNNs by classifying images from the MNIST dataset. The MNIST dataset provides single handwritten digits in the form of 28x28 greyscale images and comes pre-split with 60,000 images for training and 10,000 images for testing [20]. Given a single image, the model must recognize the digit in the image and determine if it satisfies the current context: even, odd, or divisible by three.

For this task, we construct a CNN integrated with a context layer as described in Figure 2b. The model takes a single image as input and outputs a single binary value indicating if the given digit fits the current rule. The images in the dataset are normalized such that each greyscale value is between 0 and 1, and the datasets/images are shuffled in the same manner as described in the logic gate task. The model is trained on the first 5,000 images from the training dataset and tested against the first 1,000 images from the testing dataset using a batch size of 32. Weight updates are applied using the Adam optimizer and binary-crossentropy loss. The hyperparameters are listed in Table II.

TABLE II:
MNIST Divisibility Task Training Parameters

Name	Value	Description
k	3	Number of training cycles
n	5	Number of training epochs/cycle
α	0.001	Learning rate for optimizer
α_A	0.5	Learning rate for $A(atr)$
t_{switch}	-0.003	Context-switch threshold

3) Wisconsin Card Sorting Test: The Wisconsin Card Sorting Test (WCST) is a common working memory task that requires a participant to learn and rapidly switch between various rules dictating how to sort a presented card based purely on correct/incorrect feedback. This task consists of a number of rounds and trials. During a trial, the participant is presented a stimulus card containing three features drawn from three dimensions (number, color, shape), each dimension having three possible feature values. For example, a card may consist of two red circles, three green squares, one blue

triangle, etc.. The participant is then required to match the card based strictly on a particular dimension governed at the beginning of each round, of which they are not informed. After several trials have been attempted, the round ends and a new round begins with a brand new sorting rule.

This task is modeled with a 1-step Q-learning approach using an ϵ-greedy policy with the architecture shown in Figure 2c. During an episode, a random stimulus card is generated consisting of 1 feature in each of 3 possible sorting dimensions. The model accepts as input each of these features as one-hot encoded vectors along with the action choice. Using this input, the model can predict the expected reward for the provided action given the current card and context. In this particular implementation of the WCST, the generated stimulus card is guaranteed to be free of any ambiguity that could come from feedback. After every n_{trials}, a new rule is selected at random. For models using statically-allocated ATRs, this experiment implementation guarantees that every rule will have been presented before a rule is ever repeated to ensure that one rule is not learned under two different ATRs. For models using dynamically-allocated ATRs, this constraint is not necessary. Lastly, the employed reward schedule provides the agent with $+1$ for correct sorting and -1 for incorrect sorting. The model was trained using the SGD optimizer and MSE loss function. The resulting hyperparameters are described in Table III.

TABLE III:
WCST Task Training Parameters

Name	Value	Description
n	20	Total number of rounds
n_{trials}	100	Number of trials per round
ϵ	0.1	Probability of non-greedy action
α_O	0.1	Learning rate for optimizer
α_Q	1.0	Learning rate for Q-learning
α_A	0.05	Learning rate for $A(atr)$
$A_{initial}$	$2e-4$	Initial loss value of $A(atr)$
t_{switch}	$-8e-6$	Context-switch threshold

4) 1D-Maze task: This final experiment is a temporally-extended reinforcement learning task requiring an agent to solve a 1-dimensional maze with varying goal states (positions). In this task, the agent is positioned in a random starting state at the beginning of each episode and must traverse the maze moving either left or right to locate the active goal state. The environment is partially observable in that the agent can observe its current state, but all other information is kept hidden. The maze is also periodic such that if the agent moves beyond one end of the maze it will land on the opposite side of the maze. After a number of episodes, the goal state will change positions and the agent must switch contexts to properly accommodate the new goal. In the end, the agent should learn the optimal path to each of the goal positions under their own contexts.

This task is modeled with the *n*-step Q-learning algorithm using the model architecture shown Figure 2d. This experiment consists of a maze with 10 states and 3 different possible terminal/goal states located at 0, 1, and 5. At the beginning of each episode, the agent is initialized in a random non-terminal state. The value of each action is evaluated by providing the network with the one-hot encoded state and action pair, and selected via an ϵ-greedy policy. During the episode, the agent observes a reward of $+1$ upon reaching the goal state, and -1 for any other non-terminal state. After n_{switch} episodes, a new goal position is chosen at random. Like the previous experiments, the task implementation guarantees that models with statically-allocated contexts will experience each of the possible goals before a goal re-occurs. Finally, the values of ΔC in this task during early training can be unpredictable, requiring tuning of the optional context loss initialization hyperparameters. While modifying the initial values for $A(atr)$ works well, for this experiment we utilize the alternative approach of delaying the switching mechanisms for a model's first A_{delay} episodes under a new context. Weight updates are applied using the SGD optimizer with the MSE loss function. The finalized hyperparameters are specified in Table IV.

TABLE IV:
1D-maze Task Training Parameters

Name	Value	Description
n	3000	Total number of episodes
n_{switch}	500	Goal switch frequency
ϵ	0.3	Probability of non-greedy action
α_O	0.01	Learning rate for optimizer
α_Q	1.0	Learning rate for Q-learning
α_A	0.003	Learning rate for $A(atr)$
A_{delay}	250	New-context switch delay
t_{switch}	-0.06	Context-switch threshold

IV. Results

We first examine a benchmark of the logic gate task over 100 runs. This benchmark includes control experiments with a multi-layer neural network (essentially a contextual network limited to using one ATR) along with additional contextual neural networks allocated with ATRs fewer than and exceeding the number of actual gates for comparison. As shown in Figure 3, after 8 full training cycles, the models where the number of allocated ATRs are fewer than the actual number of contexts cannot fully learn the tasks. The standard neural network (1 ATR) converged on the expected theoretical 50% accuracy. As the number of ATRs approaches the actual number of contexts, the accuracy increases and approaches 100%. It's not until the number of ATRs is equal to the number of actual contexts before the accuracy converges on 100%. After the full 8 training cycles, all 100 models consisting of 8 ATRs or higher were able to obtain 100% classification accuracy.

Next, the results of the MNIST divisibility task are examined. As mentioned previously, the role of this task is to test the framework's compatibility with CNNs, rather than provide state-of-the-art benchmarks. Analyzing Figure 4, it can be noted that the generated context deltas are very stable and distinct, making task switches easily detectable. From our experimentation, larger training datasets like MNIST provide great stability and predictability to the generated context deltas, making these tasks highly likely to succeed. By the end

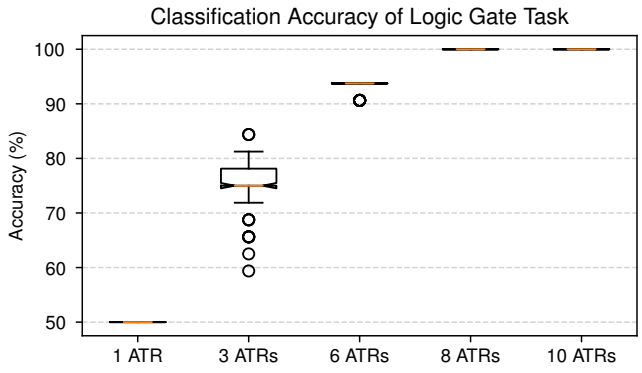


Fig. 3: Classification accuracy distributions for the logic gate task across a range of statically-allocated ATRs. For each boxplot, 100 independent models were trained, each with random initial weights and ATRs. Each model was set to use the corresponding number of ATRs, run for a training regime of 8 cycles, 50 epochs each, and the average classification accuracy across all gate functions was recorded at the end of training. The orange line indicates the median accuracy value across the 100 models and the notches above and below the median represent a 95% confidence interval. Additional hyperparameter values are specified in Table I.

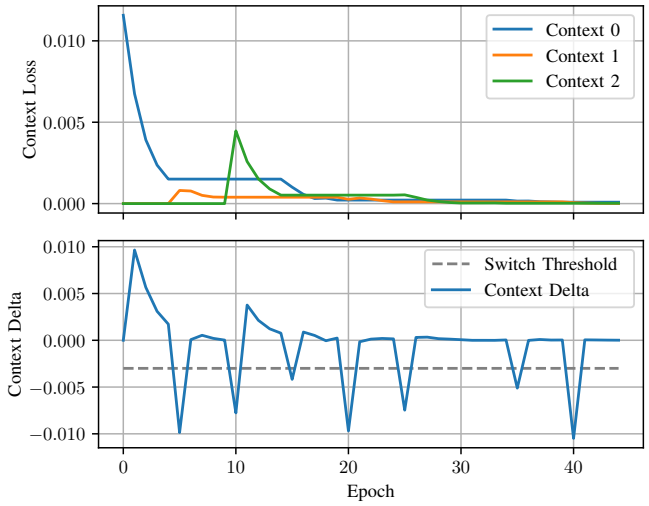


Fig. 4: The resulting context loss and context delta traces for a single model over the training period. The top graph plots the values of $A(atr)$ for each context at each epoch. The bottom graph plots the computed value of δ from each epoch. Context switches can be identified by the δ trace falling below the switch threshold value.

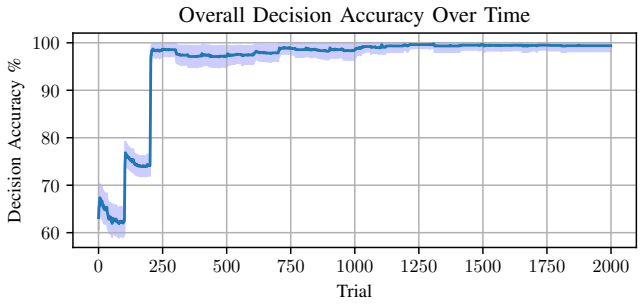


Fig. 5: The average decision accuracy of the Wisconsin Card Sorting Test over time across 100 different models. The shaded area represents the 95% confidence interval. For each model, rule is mapped to its best fitting-context where all possible cards are evaluated. The average prediction accuracy of each model is then averaged and plotted above.

of training, the model achieved an overall testing accuracy of approximately 99.07%. With this small experiment, this result successfully demonstrates the framework's compatibility with CNNs as the model is still able to achieve high classification accuracy on never-before-seen images for three different tasks.

Moving to the reinforcement learning problems, the results of the WCST are examined via a benchmark employing 100 different models utilizing statically-allocated contexts with the task. Figure 5 presents the mean of the overall classification accuracy of each model model after each trial along with a 95% confidence interval (± 1.96 standard errors). As an alternative to the typical means of evaluating a reinforcement learning problem, each model's classification accuracy was determined by applying each rule to the best-fitting context and evaluating the action choices of all possible stimulus cards. The accuracy of each context in each of the models was averaged to determine that model's overall accuracy. In the end, 99/100 models were able to achieve an overall decision accuracy of 100% across all tasks.

Finally, the 1D-maze task is analyzed by performing a benchmark of 100 different models, varying the number of available ATRs. This task also played another important role as the working memory models mentioned earlier were more limited when it came to temporally-extended tasks. This task was evaluated by examining the overall accuracy in optimal move selection for each model under each of the possible goals as plotted in Figure 6. Of these models, 97/100 were able to learn the optimal policy for each of the goal positions under their own contexts. For each of the models, each goal position in the maze is matched with the best-fitting context (i.e. the context generating the least loss) and the number of optimal moves is computed to determine the overall decision accuracy. In the end, these results not only provide a successful demonstration of the framework in a reinforcement learning setting, but also show that it is also capable of performing temporally-extended tasks.

V. Conclusions

The literature suggested that working memory is a critical feature to our decision making process for context-switching domains. Here we presented a new deep learning framework inspired by working memory models to allow traditional neural networks to autonomously learn under a varying

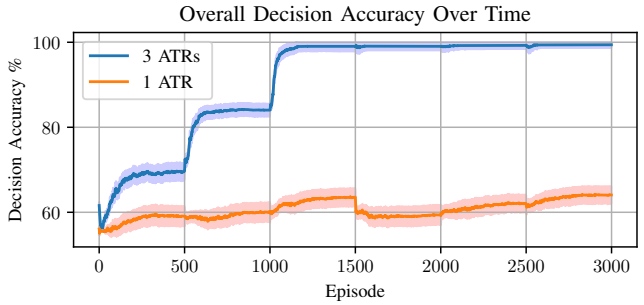


Fig. 6: The average accuracy for optimal move selection in the 1D-maze task over time for 100 different models. The shaded area represents the 95% confidence interval. For each model, each goal is mapped to its best-fitting context and the policy is evaluated for its corresponding goal position by calculating the average accuracy of optimal move selection across all possible states.

number of contexts. The implementation of this framework into Keras/Tensorflow allows for easy integration of context learning into existing neural networks with minimal required modifications to the architecture.

Along with the design of the framework, we provided four different experiments that test and demonstrate its robustness and compatibility across a variety of different network architectures and problem types. Most notably the framework showed great success when working with temporally-extended tasks where previous models were limited, and direct compatibility with CNNs. Furthermore, since the few existing models that include these contextual learning mechanisms have been designed with reinforcement learning in mind, to our knowledge our framework is the first that is also compatible with supervised learning problems.

Lastly, it is worth emphasizing that the experiments were performed with static switching thresholds. Even with these static limitations, we were able to achieve consistent results across all experiments. Future research into new mechanisms to allow more dynamic thresholds could greatly benefit more highly stochastic tasks and avoid unnecessary context switching.

References

[1] R. C. O'Reilly, D. C. Noelle, T. S. Braver, and J. D. Cohen, "Prefrontal cortex and dynamic categorization tasks: Representational organization and neuromodulatory control," *Cerebral cortex (New York, N.Y. : 1991)*, vol. 12, no. 3, pp. 246–257, Mar. 2002, ISSN: 1047-3211. DOI: 10.1093/cercor/12.3.246.

[2] R. C. O'Reilly, "Biologically based computational models of high-level cognition," *Science*, vol. 314, no. 5796, pp. 91–94, 2006, ISSN: 0036-8075. DOI: 10.1126/science.1127242.

[3] R. C. O'Reilly and M. J. Frank, "Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia," *Neural Comput.*, vol. 18, no. 2, pp. 283–328, Feb. 2006, ISSN: 0899-7667. DOI: 10.1162/089976606775093909.

[4] M. J. Frank, B. Loughry, and R. C. O'Reilly, "Interactions between frontal cortex and basal ganglia in working memory: A computational model," *Cognitive, Affective, & Behavioral Neuroscience*, vol. 1, no. 2, pp. 137–160, Jun. 2001, ISSN: 1531-135X. DOI: 10.3758/CABN.1.2.137.

[5] T. Kriete and D. C. Noelle, "Generalisation benefits of output gating in a model of prefrontal cortex," *Connection Science*, vol. 23, no. 2, pp. 119–129, 2011. DOI: 10.1080/09540091.2011.569881.

[6] T. Kriete, D. C. Noelle, J. D. Cohen, and R. C. O'Reilly, "Indirection and symbol-like processing in the prefrontal cortex and basal ganglia," *Proceedings of the National Academy of Sciences*, vol. 110, no. 41, pp. 16 390–16 395, 2013, ISSN: 0027-8424. DOI: 10.1073/pnas.1303547110.

[7] Y. Niv, R. Daniel, A. Geana, S. J. Gershman, Y. C. Leong, A. Radulescu, and R. C. Wilson, "Reinforcement learning in multidimensional environments relies on attention mechanisms," *Journal of Neuroscience*, vol. 35, no. 21, pp. 8145–8157, 2015, ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.2978-14.2015.

[8] N. P. Rougier, D. C. Noelle, T. S. Braver, J. D. Cohen, and R. C. O'Reilly, "Prefrontal cortex and flexible cognitive control: Rules without symbols," *Proceedings of the National Academy of Sciences*, vol. 102, no. 20, pp. 7338–7343, 2005, ISSN: 0027-8424. DOI: 10.1073/pnas.0502455102.

[9] J. L. Phillips and D. C. Noelle, "A biologically inspired working memory framework for robots," in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, 2005, pp. 599–604.

[10] ——, "Working memory for robots: Inspirations from computational neuroscience," in *Proceedings of the 5th International Conference on Development and Learning*, Jan. 2006.

[11] A. Gupta, L. Vig, and D. C. Noelle, "A Cognitive Model for Generalization during Sequential Learning," *Journal of Robotics*, vol. 2011, pp. 1–12, 2011, ISSN: 1687-9600. DOI: 10.1155/2011/617613.

[12] C. H. Chatham and D. Badre, "Multiple gates on working memory," eng, *Current opinion in behavioral sciences*, vol. 1, pp. 23–31, Feb. 2015, PMC4692183[pmcid], ISSN: 2352-1546. DOI: 10.1016/j.cobeha.2014.08.001.

[13] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, no. 5306, pp. 1593–1599, Mar. 1997.

[14] C. H. Chatham, M. J. Frank, and D. Badre, "Corticostriatal output gating during selection from working memory," eng, *Neuron*, vol. 81, no. 4, pp. 930–942, Feb. 2014, S0896-6273(14)00006-3[PII], ISSN: 1097-4199. DOI: 10.1016/j.neuron.2014.01.002.

[15] G. M. DuBois and J. L. Phillips, "Working memory concept encoding using holographic reduced representations," in *Proceedings of the 28th Modern Artificial Intelligence and Cognitive Science Conference*, 2017.

[16] A. S. Williams and J. L. Phillips, "Multilayer context reasoning in a neurobiologically inspired working memory model for cognitive robots," in *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, 2018.

[17] ——, "Transfer reinforcement learning using output-gated working memory," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence, New York, NY.*, 2020.

[18] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[19] M. Jovanovich and J. L. Phillips, "N-task learning: Solving multiple or unknown numbers of reinforcement learning problems," in *In Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, 2018.

[20] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.