NEURAL REPRESENTATIONS IN MULTI-TASK LEARNING GUIDED BY TASK-DEPENDENT CONTEXTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The ability to switch between tasks effectively in response to external stimuli is a hallmark of cognitive control. Our brain is able to filter and to integrate external information to accomplish goal-directed behavior. Task switching occurs rapidly and efficiently, allowing us to perform multiple tasks with ease. In a similar way, deep learning models can be tailored to exhibit multi-task capabilities and achieve high performance across domains. Still, understanding how neural networks make predictions is crucial in many real-world applications.

In this study, we delve into neural representations learned by multi-tasking architectures. Concretely, we compare *individual* and *parallel* networks with task-switching networks. Task-switching networks leverage task-dependent contexts to learn disentangled representations without hurting the overall task accuracy. We show that task-switching networks operate in an intermediate regime between *individual* and *parallel* networks. In addition, we show that shared representations are produced by the emergence neurons encoding multiple tasks. Furthermore, we study the role of contexts across network processing and show its role at aligning the task with the relevant features. Finally, we investigate how the magnitude of contexts affects the performance in task-switching networks.

1 INTRODUCTION

Living involves constantly gauging and selecting the optimal task to perform. This decision is the result of the interaction of different elements, such as our current goals, external circumstances, or the stimulus context (Monsell, 2003). In the brain, task switching and cognitive control has been associated principally with the prefrontal cortex (PFC), which provides top-down regulation to other cortical brain areas Johnston et al. (2007). In this sense, PFC controls the activation of multiple neural pathways that are activated or deactivated to ultimately result in task execution. Neural pathways are formed by collections of neurons that cooperate to yield a specific effect. Although single neurons have been historically regarded as centre of processing in the brain, we are now moving towards a framework in which neural networks conform the functional processing unit of the nervous system Yuste (2015). For example in the visual cortex, where subpopulations of neurons are responsible of the processing of different visual field features for correct object recognition DiCarlo et al. (2012).

One possible explanation how the top-regulation exerted by PFC can engage other brain areas is the gating theory, whereby contexts regulate the activation of neural pathways (Miller et al., 2001). The encoding of relationships between stimulus and contexts can be facilitated by tuning neural activity to multiple tasks. In the PFC of monkeys, it has been observed that single neurons exhibit nonlinear responses to multiple stimuli (Rigotti et al., 2013). This behavior, or mixed selectivity, favors high-dimensional representation of neural activation, which allows simple linear readouts generate a vast number of responses, in opposition to low-dimensional representations (Fusi et al., 2016).

Recently, artificial neural networks have been revisited as models of neural computation, many findings suggesting their practicality for assessing brain theories Richards et al. (2019). For example, the internal representations learned by neural networks have been associated with representations in the brain in multi-tasking settings (Ito et al., 2022; Ito & Murray, 2021; Flesch et al., 2022a;b).

In this paper, we investigate the neural representations learned by multi-tasking architectures. Neural networks have been designed to be capable of processing multiple tasks in parallel, which is

beneficial for the network performance (Caruana, 1997). Here, we focus in neural networks using contexts to switch attention between tasks. We use population analysis tools to investigate how neural computations are associated with task stimulus Kriegeskorte et al. (2008); Jazayeri & Ostojic (2021) and show the advantages of learning representations using task-switching networks.

1.1 Related Work

Neural networks with contexts have been used as models to study cognitive control in machines and humans. Mante et al. (2013) reproduced PFC dynamics from monkeys using recurrent neural networks with sensory contexts. Similarly, Ardid & Wang (2013) analyzed how task-switching behavioral effects, such as switch and congruency costs, emerge from network attractor dynamics. In Musslick et al. (2017), authors studied the learning efficiency of neural networks in multi-tasking with contexts and tasks with ranging degrees of overlapping. Flesch et al. (2022a) analyzed the geometry of representations learned by neural networks and humans in a task-switching schedule. Later, Flesch et al. (2022b), modified the stochastic gradient descent to strengthen task-relevant features in a continual learning setting. More recently, Ito et al. (2022) studied the generalization of new tasks by composing old tasks using different rule-based contexts. In Ito & Murray (2021), authors used a neural network to investigate the transformation mapping between visual and motor representations occurring in the brain. They used representational similarity analysis to study the geometry of neural codes in multi-tasking Kriegeskorte et al. (2008); Kriegeskorte & Kievit (2013).

Contexts have been used to alleviate catastrophic forgetting in continual learning. Masse et al. (2018) showed that adding context to parameter stabilization methods improved accuracy when compared to parameter stabilization alone. Serra et al. (2018) used gating by using task-based attention mechanisms to help preserving information of previous tasks without compromising new ones. In Grewal et al. (2021), authors combined synaptic intelligence with active dendrites and sparse activations to reduce catastrophic forgetting. Also, the idea of recruiting multiple subnetworks for continual learning has been explored before in Wortsman et al. (2020).

Li et al. (2016) proposed multi-bias non-linear activations to improve feature extraction in convolutional neural networks. Task-switching networks were introduced by (Sun et al., 2021) for semantic segmentation using task embeddings to promote parameter sharing in the network.

1.2 Our Contributions

The main contributions of this paper are twofold:

1. Firstly, we investigate the representations learned by three different variations of multi-layer perceptrons (MLP). We show that addition of contexts help learning internal representations that are richer in comparison to other multi-task paradigms, in terms of shared representations and output reconstruction (3.1).
2. Secondly, we conduct a series of experiments to evaluate task-switching networks and expand previous work involving multi-task learning with contexts (3.3) and present novel research regarding contexts (sections 3.5, 3.4, 3.4).

2 Methods

2.1 Architectures

We conducted different experiments with three variants of the MLP (Rumelhart et al., 1986), (Goodfellow et al., 2016) in multi-task learning:

1. **Individual Networks**: Each task is learned by an independent network. Multi-tasking is performed by combining the outputs of the networks. An individual network is parameterized as $\mathbf{y_t} = f_t(\mathbf{x}; \theta_t)$, where t denotes the task in the set $T \in \{1, 2, ..., N\}$ of N tasks, θ_t denotes the weights and biases specific for each task, $\mathbf{W_t}$ and $\mathbf{b_t}$ for each layer. Parameters are not shared across tasks.
2. **Parallel Networks**: Tasks are processed simultaneously and each one is decoded in the last layer. The parallel processing regularizes the network and allows overall better per-

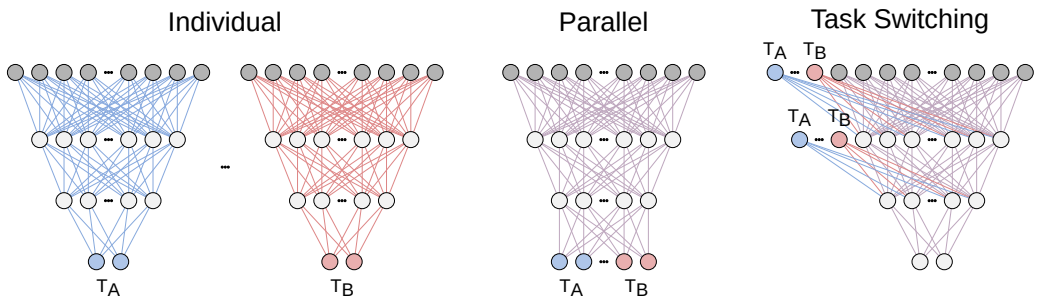

Figure 1: Architectures for multi-task learning addressed in this paper.

formance than the individual networks Caruana (1997). A parallel network is defined as $\mathbf{y} = f_t(\mathbf{x}; \theta)$. Parameters are fully shared across tasks.

3. **Task-switching Networks**: Each task is processed individually, but networks resources are shared and optimized across tasks. The output of a task-switching network is $\mathbf{y_t} = f_t(\mathbf{x}; \mathbf{W}, \mathbf{b_t})$. Weights are co-optimized for all tasks, but biases are exclusive. We define a task-switching network to be a composition of contexts layers, which can include the bias signal for each task, $y = \mathbf{W^T x} + \mathbf{b_t}$ or not $y = \mathbf{W^T x}$. A task-switching network require of at least of one context layer with bias. At running time, we use the bias associated with the task to perform by encoding tasks in a one-hot encoding vector. We train task-switching networks following the interleaved task-switching schedule (Flesch et al., 2018).

Architectures are depicted in figure 2.1. The detailed description of the training conditions are described in A.1.

2.2 Tasks

In this study, we created five binary tasks using the MNIST database (LeCun et al., 1998). The main task pair we use throughout the study correspond to *parity* and *value*. Parity defines the number as either even or odd, whereas small defines a number being smaller or equal to four or larger. The detailed list of tasks and outputs is explained in A.2.

3 Results and Discussion

3.1 Task-switching networks operate in an intermediate regime between individual and parallel networks

The first experiment consisted in analyzing the neural representations of three different multi-task learning networks. Similarly as in (Caruana, 1997), we compared the multi-tasking using individual neural networks (one for each task) and parallel neural networks, where representations are learned simultaneously. In addition, we included task-switching networks, where tasks are modulated according to the current task context.

To analyze the representations learned by each network, we used the representational similarity analysis (RSA) Kriegeskorte et al. (2008). We build the representational dissimilarity matrix (RDM) by grouping inputs and neural activations of each layer. For each number we averaged the neural activity and computed the dissimilarity between different numbers. We repeated this process for different tasks and calculated the dissimilarity between class labels (numbers). Finally, we assembled the RDM. Section A.3 addresses the calculation of the RDM in detail.

For individual and task-switching networks, calculating the dissimilarity between tasks can be done directly. For individual networks, we have the activation of remaining networks, and for task-switching networks, we have the activity produced by changing the task-encoded context. Nonetheless, for parallel networks, since the processing for both tasks is the same, representations are entangled. Then, we construct the RDM by using the same activation averages for each task.

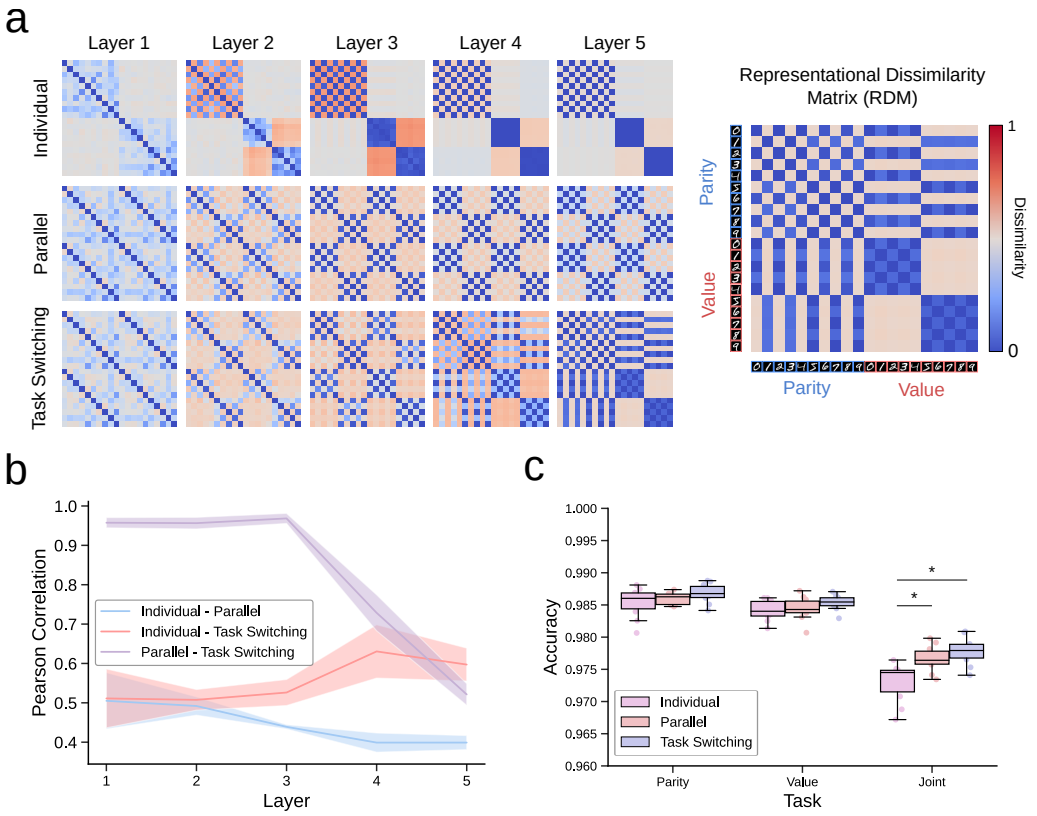

Figure 2: (a) RDM for the *parity/value* task-switching schedule for the three architectures. (b) Pearson correlation coefficient between RDMs (mean of 10 runs with standard deviation). (c) Accuracy of the different models.

Figure 3.1b shows the average RDM for the three models after training with *parity* and *value*. Individual networks do not share representations between tasks (off-diagonal). Parallel networks separate partially the congruent and incongruent numbers to be resolved in the output layer. Task-switching networks, behave similarly as parallel in the first tasks but resolves the task before the output layer, similarly as the individual networks. Importantly, we can reconstruct the task goal from the activity of both individual and task-switching networks. However, we can only reconstruct the task congruency from parallel. Additionally, in comparison to individual networks, task-switching networks share representations between tasks. To quantify the degree of similarity between representations, we calculated the Pearson correlation between RDMs for different runs. Results in figure 3.1c reinforce the previous assessment: task-switching networks behave similarly to parallel networks in the first stage of processing, and ends similar to the individual networks.

Finally, we assessed the performance of networks and analyzed the performance in terms of the accuracy achieved across runs for both tasks. Figure 3.1d shows a similar performance of the three networks for the two tasks, without significant differences. We also evaluated the performance on both tasks simultaneously. We observed significant differences (Wilcoxon Signed Rank Tesk, $p < 0.05$) between individual and parellel on the one hand, and individual and task-switching networks in the other. The co-optimization between tasks for parallel and task-switching networks regularizes the network performance. In the case of the individual, this regularization does not exist. We analyzed same analysis as 3.1 using 5 different tasks. The results are shown in B.

3.2 TASK-SWITCHING NETWORKS PERFORM MULTIPLE TASKS BY EXTRACTING TASK CONGRUENCY INFORMATION

The second experiment consisted in analyzing the representations learned by task-switching networks when contexts are removed after training. Tasks share a common pool of neural activations that is driven towards one task or another depending on the task context. To study this common computation, first we investigate the RSA by calculating the correlation between RDMs of three variants of the task-switching network: only context in the first layer (*first*), context in all layers (*all*), and trained with all contexts but removed at inference time (*removed*). Figure 3.2a shows the RSA between these networks. *removed* and *all* have similar representations in the first layers, only to diverge as we approach the output layer. Contrarily, *first* and *all* start with lower correlation and finish with maximum. The drop of correlation in the first layers is caused by the interference of the only context with the feature extraction of number in the *first* network.

To further analyze the representations, we computed multidimensional scaling on the RDMs (Jazayeri & Ostojic, 2021). For implementations details see (see A.4). Figure 3.2b shows the MDS for the three first layers and the two last. Numbers are represented differently for both tasks in the *first* network, since the context bias has been introduced early in the processing. Here, the network has to resolve the task rapidly before the task signal decays. In layer 3 of the *all* and *removed* we see the projection of numbers into an orthogonal manifold: one dimension represents the incongruent numbers in the *parity/value* task schedule and in the other, the congruent numbers.

In addition to MDS, we used a generalized linear model to decode the activations in the original dimensional space (A.5. Figure 3.2d shows the performance of the classifier when we tested the linearity of number, task and congruency. In high-dimensional space, number is decoded with high accuracy in the first layers. For all three models, the accuracy collapses in the last layers. *removed* and *all* are slightly better than *first*, possibly due to the context interference in the first layer. Task information is already present in the first layer of *first*. *all* shows low accuracy in the first layers, as if the contexts where not acting before the third layer (more about weight importance in 3.4). Finally, to test the hypothesis that task-switching networks extract congruency information, we evaluated the performance on congruency itself. *removed* is capable of decoding perfectly congruency across layers in the high-dimensional space defined by the neural activations. Here, *first* shows the greatest decrease in accuracy when approaching the last layers. This plot confirms the hypothesis that contexts drive neural activity towards the goal of the task. During training, both tasks are being optimized together, reaching some local minima that reduces the loss for both tasks. Congruency is the midpoint between the optimization of both task and determines the amount shared representations. In the extreme case of two identical tasks, the overlapping is maximum and the bias signals would converge to the same, reducing then the multi-tasking problem to a single-task problem, with contexts working as regular activation biases.

3.3 MIXED SELECTIVITY EMERGES FOR RESOLVING TASK EXECUTION

After having inspected the behavior of populations of neurons in single layers, we wanted to investigate the role of single units, specially in the last layer, where the network is ready to predict the correspondent output. We assumed that multiple of neurons have to behave similarly to drive collectively the dynamics towards the task goal. Mixed selectivity has been previously observed in artificial neural networks with a single hidden layer and contexts in the first layer (Flesch et al., 2022a). Here, we focused on activations in the last layer and performed agglomerative hierarchical clustering in to cluster neurons. The details of the method employed are in A.6.

The results of the clustering are shown in figure 4. The plots represent the mean activity of different clusters when presented with one of the numbers. We used two networks from the previous section, *all* and *removed*. The activity was normalized to a maximum of 1 (radius of the plot). We observed the existence of two different clustering patterns. In both, we found a cluster of dead units that is not active for any of the numbers. The first row of the plot represents the first pattern. In this pattern of activation, there is a single cluster that responds to even numbers in parity and values smaller or equal to four in the value task. When contexts are removed the cluster responded only to the even numbers and smaller or equal to four, entailing that context increases the activity for 6 and 8 in parity and 1 and 3 in value, that is the incongruent numbers. With the remaining numbers, the network

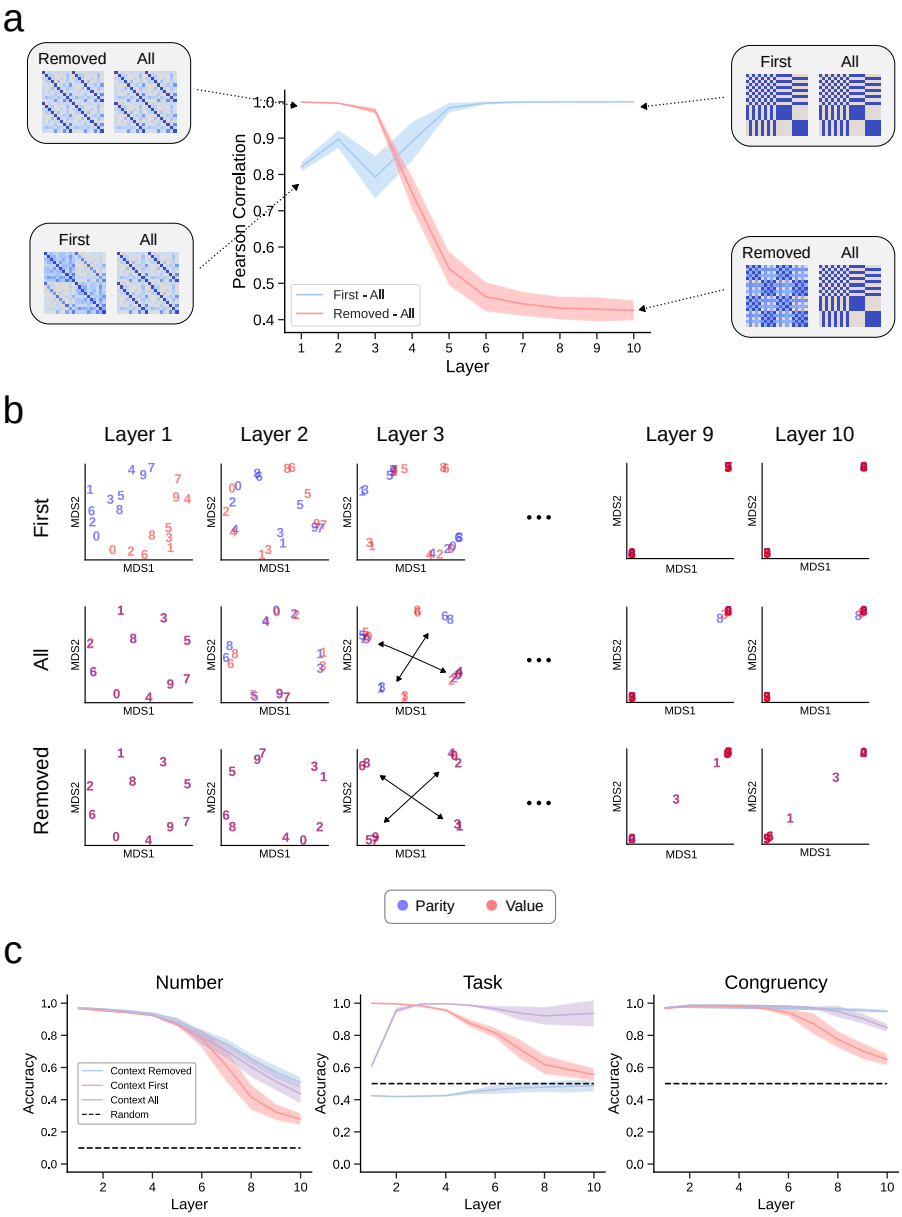

Figure 3: (a) RSA between *first*, *all* and *removed* network. (b) Multidimensional scaling (MDS) of the representational dissimilarity matrix (RDM). (c) Linear decoder of neural activity.

works thresholding neural activity. For example, odd numbers in the parity task show zero neural activity, and thus classified as odd numbers.

The second pattern of activation is showed in the second row of 4. We found two main clusters of activity. The first cluster is responsible to even numbers and values equal or small to four, the second cluster to odd numbers and values larger than four. When contexts are removed, clusters encode for congruent numbers. These patterns of activations show how neurons are tuned to different tasks depending on the current active stimulus. In the brain, neurons that are responsive to multiple tasks are said to exhibit mixed selectivity (Rigotti et al., 2013). We repeated the experiment using five tasks and found that the majority of patterns resembled pattern number two. Results are in figure B in the appendix.

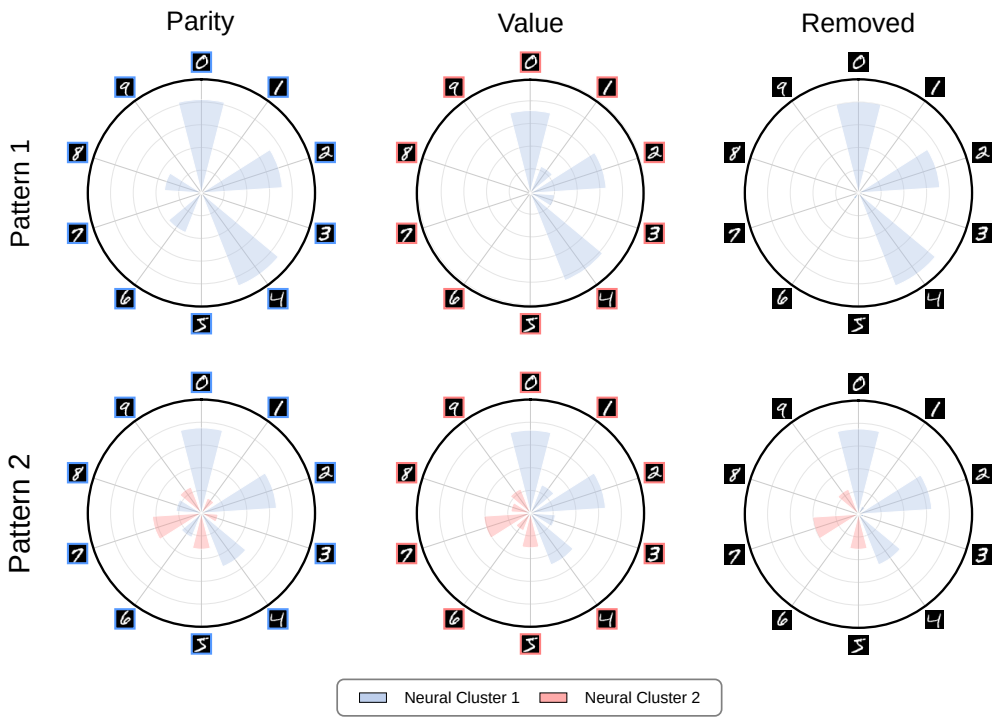

Figure 4: Neural clustering for the *parity/value* task-switching schedule.

3.4 Context strength is maximized at intermediate layers

After observing the impact of context in the first layer (section), we hypothesized that context location could be essential for proper input feature extraction and to drive the activity towards the task goal. Flesch et al. (2018) observed anticorrelation in the contexts of tasks situated in the first layer in networks with one hidden layer. Here, we run a series of experiments to assess the strength of contexts varying their position across the network. Figure 5 shows the results. Each row represents a different context configuration and each column is the index of the layer (we fixed the maximum number of layers at 10). If a cell is colored according to the heatmap, the network has a context at that position. If the cell is gray, the network has a layer but without context. If white, it denotes the absence of the layer, and thus the network in shallower. We tried four different settings adding contexts from the first layer (5a), adding contexts starting from the last layer (5b), adding a single context at different layers (5c), adding contexts and layers at the same time (5d). In the two first heatmaps we show the Frobenius norm of the weights associated with the corresponding task. Then we calculate the correlation between contexts of different tasks at the same location. Finally, we show the average accuracy (and standard error) of the networks in the same row. The dashed line in the accuracy plot represents the mean accuracy achieved by a network of ten layers and ten contexts (benchmark).

Results show that context location is not trivial. Largest context strength is achieved at intermediate layers, where the norm of contexts is higher. At the same time, the anticorrelation between contexts of different tasks is higher. Context have positive correlation if located in the end in networks with multiple contexts. In addition, adding context in the beginning or in end have associated the largest magnitude of context strength. It can be explained by the fact that the whole task resolution is located in a small processing step. When assessing the accuracy, the presence of contexts in intermediate layers achieved overall the same performance the benchmark.

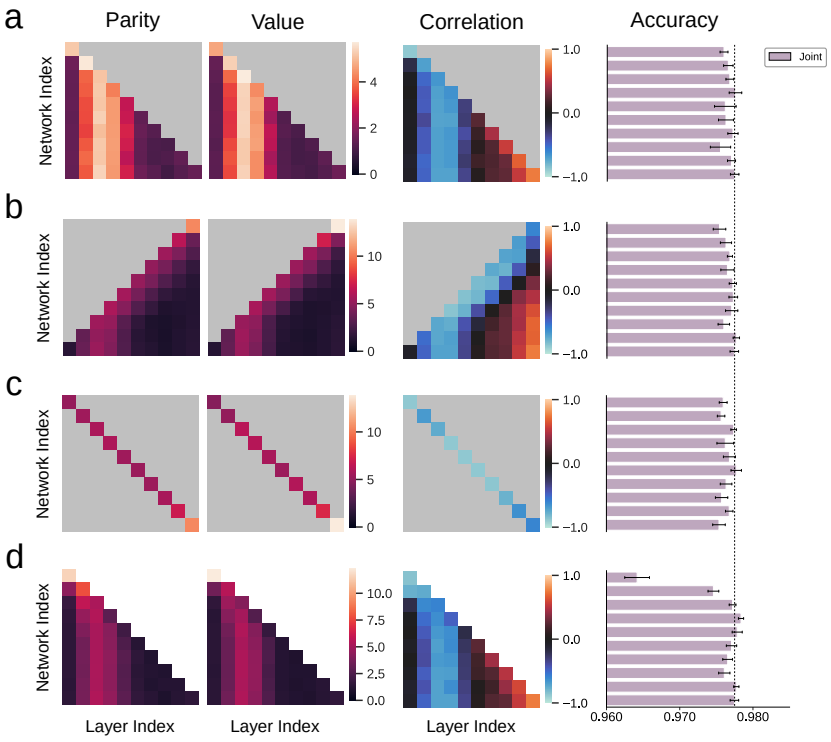

Figure 5: Magnitude and correlation between contexts using different network architectures. (a) Context are added from the first layer. (b) Context are added from the last layer. (c) One context per network. (d) Networks have different number of layers.

3.5 Task-switching networks generalize representations between tasks

The last experiment involved the analysis of the neural representations when the magnitude of contexts was modified from the values of training. For each task, since we use one-hot encoding, the contexts have a default value of 1 if the task is to be executed and 0 otherwise. However, the value of the context can be modified at test time to take any possible value. In addition, many contexts can be activated at the same time. We used a 10 layer network with contexts in all layers. In the last experiment, we analyze the behavior of the task-switching network when modifying contexts beyond training values. Furthermore, we inspect the neural representations to gain insight of how networks perform tasks.

First, we studied the how is the transition from one task to another. To do so, we interpolated the values of contexts between the two tasks, passing by an intermediate value of 0.5. The total sum of the contexts was kept to 1, and the values of one task are complementary to the other. Figure 5 shows the accuracy for the different values of the context. We moved from task *value* to task *parity*. Results show that the transition between tasks is smooth: there is not a hard threshold at 0.5, but a progressive change from one task to other. The neural representations at 0.5 show an intermediate state between tasks. The baseline accuracy is 0.6 since it is the level of congruency between tasks. The accuracy at 0.5 is close to 0.8 that results from adding to the congruency baseline the half of the remaining incongruent numbers due to random chance.

Next, we wanted the strength of opposition between tasks. We incremented the value of both contexts simultaneously from 0 to 1. Figure 5b shows the accuracy values for both tasks. The accuracy of both tasks begins at 0.8 and slightly diverges when reaching a value of 1 for each task. There is a slight preference of the network for the task *parity*. The internal representations however, show that the network operates in an intermediate step between tasks as observed in figure 5a when both tasks had contexts of 0.5.

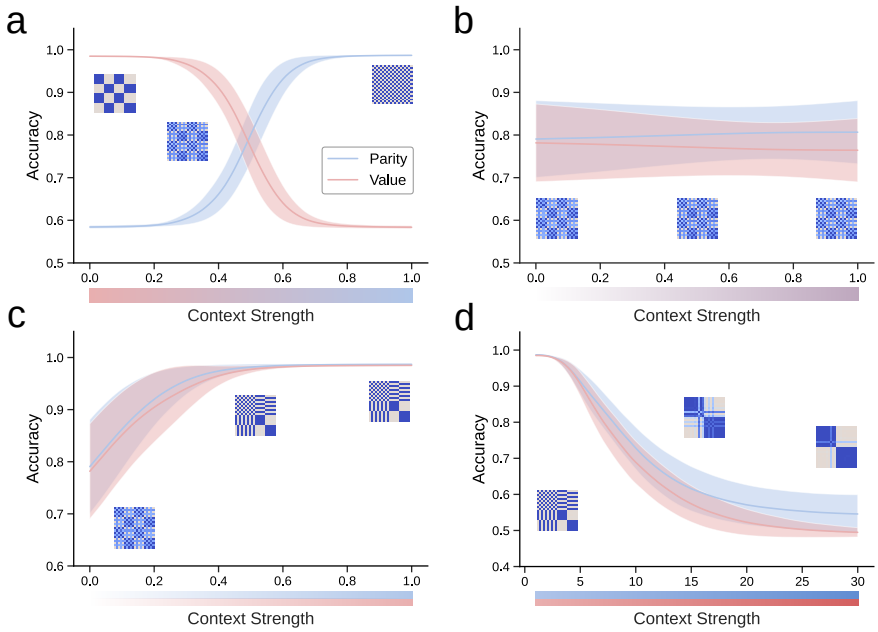

Figure 6: Accuracy of tasks as functions of context intensities. (a) The total context strength is kept fixed between tasks. (b) Both tasks are increased simultaneously. (c, d) Contexts are increased and evaluated independently (no interaction).

Finally, we wanted to analyze how the network performed each single task separately when the values of the contexts were under or higher the the unity. First, we evaluated what occurs when the contexts take values inferior to 1. We swept each task independently from 0 to 1, and represented the values in the same plot (figure 5c). Results indicate that networks do not need to get the same magnitude of context to operate with the same performance. Concretely, we observe that with contexts close to 0.5, the networks achieve the same accuracy. We took each network and created the RDM to compare with the representations learned at context 1. We observed that also, close to 0.5, the network achieves similar representations.

We inspected the generalization for values greater than 1. Figure 5d shows the accuracies for the independent tasks when the values of contexts are 30 fold of those in training. When the context signal increases, there is an asymmetry between number extraction and task information that leads to a decrease of accuracy. The dissimilarity between mean activations becomes close to zero (maximum correlation).

4 Conclusions

In this study, we have conducted a series of experiments with multi-task networks focusing on task-switching networks, which leverage contexts to switch between tasks. We have seen that task-switching networks learn disentangled shared representations that allow to reconstruct the mapping between task and goal. Task-switching networks encode congruency to resolve multi-tasking. Contexts can either improve feature extraction or interfere with the network processing, depending on their position in the network. Finally, we have shown that task-switching networks generalize representations between tasks.

REPRODUCIBILITY STATEMENT

We attached the code to train and generate the figures together with the submission of this paper.

REFERENCES

Salva Ardid and Xiao-Jing Wang. A tweaking principle for executive control: neuronal circuit mechanism for rule-based task switching and conflict resolution. *Journal of Neuroscience*, 33 (50):19504–19517, 2013.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

Timo Flesch, Jan Balaguer, Ronald Dekker, Hamed Nili, and Christopher Summerfield. Comparing continual task learning in minds and machines. *Proceedings of the National Academy of Sciences*, 115(44):E10313–E10322, 2018.

Timo Flesch, Keno Juechems, Tsvetomira Dumbalska, Andrew Saxe, and Christopher Summerfield. Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, 110(7):1258–1270, 2022a.

Timo Flesch, David G Nagy, Andrew Saxe, and Christopher Summerfield. Modelling continual learning in humans with hebbian context gating and exponentially decaying task signals. *arXiv preprint arXiv:2203.11560*, 2022b.

Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current opinion in neurobiology*, 37:66–74, 2016.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323. JMLR Workshop and Conference Proceedings, 2011.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Karan Grewal, Jeremy Forest, Benjamin P Cohen, and Subutai Ahmad. Going beyond the point neuron: Active dendrites and sparse representations for continual learning. *bioRxiv*, 2021.

Takuya Ito and John D Murray. Multi-task representations in human cortex transform along a sensory-to-motor hierarchy. *bioRxiv*, 2021.

Takuya Ito, Tim Klinger, Douglas H Schultz, John D Murray, Michael W Cole, and Mattia Rigotti. Compositional generalization through abstract representations in human and artificial neural networks. *arXiv preprint arXiv:2209.07431*, 2022.

Mehrdad Jazayeri and Srdjan Ostojic. Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Current opinion in neurobiology*, 70:113–120, 2021.

Kevin Johnston, Helen M Levin, Michael J Koval, and Stefan Everling. Top-down control-signal dynamics in anterior cingulate and prefrontal cortex neurons following task switching. *Neuron*, 53(3):453–462, 2007.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikolaus Kriegeskorte and Rogier A Kievit. Representational geometry: integrating cognition, computation, and the brain. *Trends in cognitive sciences*, 17(8):401–412, 2013.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, pp. 4, 2008.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Hongyang Li, Wanli Ouyang, and Xiaogang Wang. Multi-bias non-linear activation in deep neural networks. In *International conference on machine learning*, pp. 221–229. PMLR, 2016.

Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

Earl K Miller, Jonathan D Cohen, et al. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.

Stephen Monsell. Task switching. *Trends in cognitive sciences*, 7(3):134–140, 2003.

Sebastian Musslick, Andrew Saxe, Kayhan Özcimder, Biswadip Dey, Greg Henselman, and Jonathan D Cohen. Multitasking capability versus learning efficiency in neural network architectures. Cognitive Science Society, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

Mattia Rigotti, Omri Barak, Melissa R Warden, Xiao-Jing Wang, Nathaniel D Daw, Earl K Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497 (7451):585–590, 2013.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popović, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, and Luc Van Gool. Task switching network for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8291–8300, 2021.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

Rafael Yuste. From the neuron doctrine to neural networks. *Nature reviews neuroscience*, 16(8): 487–497, 2015.

A APPENDIX

A.1 NETWORK TRAINING

For each architecture, we trained two different models with 5 (figure 3.1) and 10 layers (figure 3.2, 4, 5 and 3.5). Networks were trained for a total number of 50 epochs. The number of hidden units per layer was fixed to 100. We selected ReLU as the activation function Glorot et al. (2011) and initialized weights and biases following a uniform distribution in the range $(-1/\sqrt{N}, 1/\sqrt{N})$, where N is the number of inputs to the layer. We used the Adam optimizer for training (Kingma & Ba, 2014) with learning rate 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and the cross entropy loss function. The models were implemented in PyTorch Paszke et al. (2019). We controlled the random number generator for reproducibility and hypothesis testing.

A.2 BINARY TASKS

We designed five main tasks using MNIST. For all tasks, we split the images into a training set of 50,000 images and a test set of 10,000. The batch size was fixed to 100 images. The following table denotes the names of the task and the expected output of the network (0, first output unit, 1 second output unit):

Task Name	Number									
	0	1	2	3	4	5	6	7	8	9
Parity	1	0	1	0	1	0	1	0	1	0
Value	1	1	1	1	1	0	0	0	0	0
Prime	0	0	1	1	0	1	0	1	0	0
Fibonacci	1	1	1	1	0	1	0	0	1	0
Multiple 3	0	0	0	1	0	0	1	0	0	1

We define congruency as the overlapping between outputs of pair of tasks:

	Parity	Value	Prime	Fibonacci	Multiple 3
Parity	1	0.6	0.3	0.5	0.4
Value	0.6	1	0.5	0.7	0.4
Prime	0.3	0.5	1	0.6	0.5
Fibonacci	0.5	0.7	0.6	1	0.3
Multiple 3	0.4	0.4	0.5	0.3	1

A.3 REPRESENTATIONAL DISSIMILARITY MATRIX (RDM)

For each layer, we construct an activation matrix consisting of $N_T \times N_{test}$ rows and N_{hidden} columns, where N_T is the number of tasks, N_{test} is the test set size, 10,000 and N_{hidden} the number of hidden units of a layer. For each label of MNIST, we will calculate the average activation for the hidden units in the layer. For each task, we create a matrix of dimension $10 \times N_{hidden}$. This matrix represent the average unit response to each of the possible stimulus of the dataset. Finally, we calculate the dissimilarity between row vectors, denoting the dissimilarity between the average of two numbers for two different tasks. We calculate the dissimilarity between vectors as $1 - r/2$, where r is the Pearson correlation coefficient.

A.4 MULTIDIMENSIONAL SCALING (MDS)

To construct the multidimensional scaling embedding we create first the RDM of a layer. The RDM is a distance matrix where a position ij denotes the distance between i and j. We project this high-dimensional points into a two-dimensional embedding using MDS. We used the scikit-learn module to implement MDS Pedregosa et al. (2011). The maximum iteration was set to 1000 to guarantee convergence. The tolerance was set to 10^{-5}.

A.5 LINEAR DECODER

We used multiclass logistic regression as linear decoder of neural activations across layers. We used the scikit-learn module to implement the logistic regression model Pedregosa et al. (2011). The maximum iteration was set to 5000 to guarantee convergence. The tolerance was set to 10^{-3}. We used L2 regularization and cross entropy as loss function.

We constructed an activation matrix consisting of $N_T \times N_{test}$ rows and N_{hidden} columns, where N_T is the number of tasks, N_{test} is the test set size, 10,000 and N_{hidden} the number of hidden units of a layer to decode. We divided the rows into a train set and test set following a 90/10 split. We used the labels of numbers to create the task and congruency labels. We assembled the activation matrix for each layer of a network and averaged the results of the 10 different runs.

A.6 AGGLOMERATIVE HIERARCHICAL CLUSTERING

Similarly as in the assembling of the RDM, we average the neural activity for the ten numbers, having a matrix of size $10 \times N_{hidden}$. Here we gather the averages of activity for the different tasks to construct a matrix of size $N_{hidden} \times 10 \cdot N_T$. Each independent row conforms the averages of activations across tasks for a single hidden unit. We normalize the activity to be maximum of 1. We apply hierarchical clustering to create the dendrogram of hidden units using the *centroid* method in SciPy (Virtanen et al., 2020). The we find the cluster of each hidden unit. Finally, we calculate the average activity of each cluster.

B SUPPLEMENTARY FIGURES

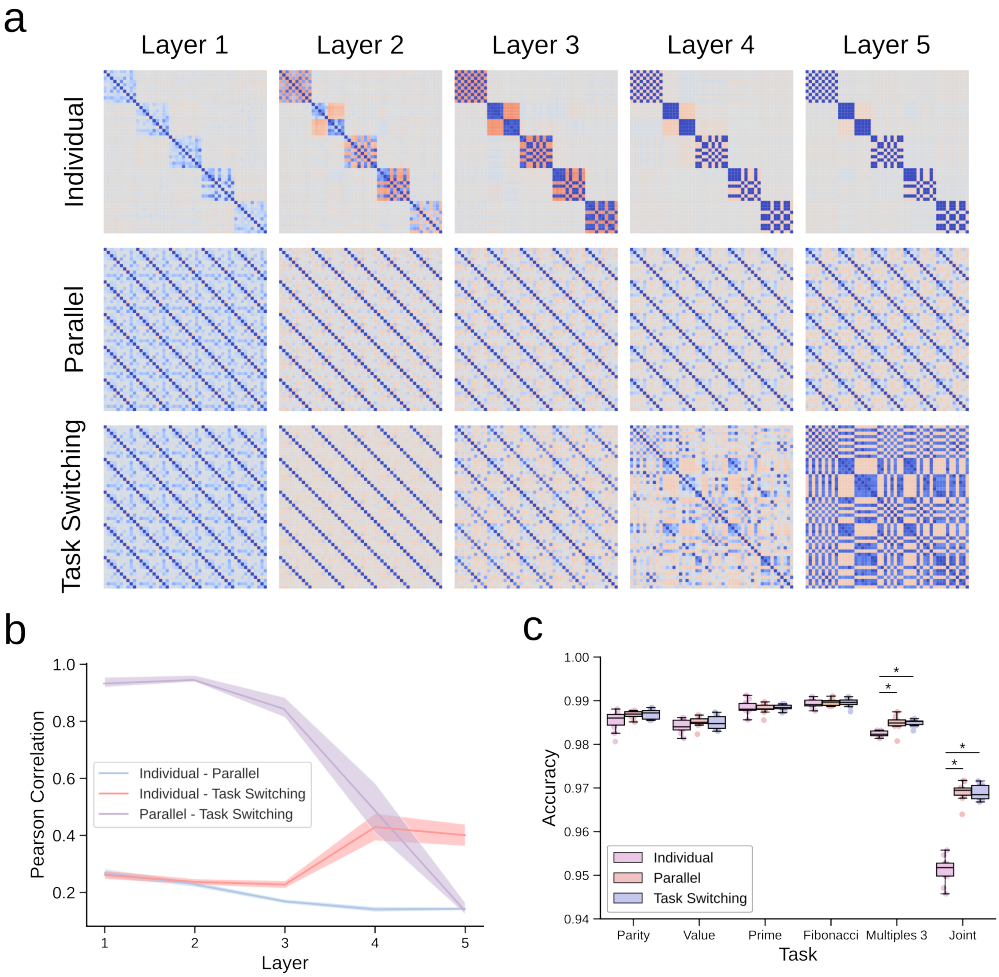

Figure 7: (a) RDM for the *parity/value/prime/fibonacci/multiple3* task-switching schedule for the three architectures. (b) Pearson correlation coefficient between RDM (mean of 10 runs with standard deviation). (c) Accuracy of the different models.

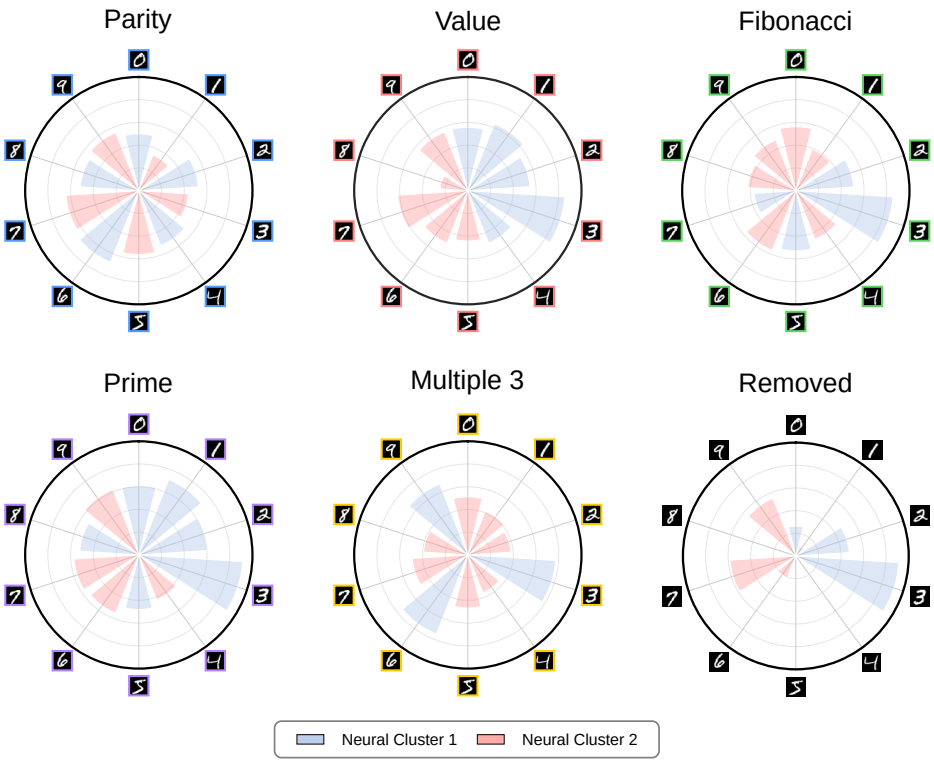

Figure 8: Neural clustering for the *parity/value/prime/fibonacci/multiple3* task-switching schedule.