Transformer Mechanisms Mimic Frontostriatal Gating Operations When Trained on Human Working Memory Tasks

Aneri Soni,[*34] Aaron Traylor,[*1] Jack Merullo,[1] Michael J. Frank,[†23] & Ellie Pavlick[†1]
∗ Co-first author
† Co-Senior Author, Equal Contribution
[1] Department of Computer Science: Brown University, Providence, Rhode Island
[2] Department of Cognitive and Psychological Sciences: Brown University, Providence, Rhod
[3] Carney Institute for Brain Science: Brown University, Providence, Rhode Island
[4] Department of Neuroscience: Brown University, Providence, Rhode Island
{aneri_soni, aaron_traylor, jack_merullo, michael_frank, ellie_pavlick}@brown

April 15, 2026

Abstract

Working memory (WM) is thought to rely on sophisticated frontostriatal mechanisms for selective gating, supporting updating and readout of information to and from distinct memory "addresses" (neural populations). According to computational models, capacity limitations in WM arise due to challenges in "role addressability": learning to correctly bind items in memory to their respective roles and assigning credit to the corresponding gating operations. However, these conclusions are based on particular assumptions about biological neural networks and it is unclear whether the principles generalize to other architectures. To address this question, we examine whether similar gating mechanisms emerge in Transformer neural network architectures, which have demonstrated success on tasks requiring executive function — the ability to represent, coordinate, and manage multiple subtasks — yet lack intentionally built-in gating mechanisms. We analyze the mechanisms that emerge within Transformers trained on human WM tasks explicitly designed to place demands on gating. We find that the Transformer's attention mechanism develops role-addressable input and output gating, but only when trained on task distributions that benefit from frontostriatal-like gating mechanisms. Moreover, these gating strategies support enhanced generalization and variable binding, and increase the models' effective capacity to store and access multiple items in memory, resembling the constraints found in frontostriatal models. These results suggest that gating mechanisms serve a fundamental computational role in managing role addressability and binding, and highlight opportunities for future research on computational

similarities between modern AI architectures and models of the human brain.

1 Introduction

Working memory (WM) involves the ability to store, maintain and manipulate multiple items such that they can be readily used for ongoing task processing and to constrain behavior. Working memory capacity is, however, limited (Cowan, 2008). Theoretical accounts differ in the nature of this limitation (e.g., whether it reflects a limited number of discrete slots, or a common pool of resources to store multiple items, thereby limiting their precision as WM load grows; Ma, Husain, and Bays (2014)). Various hybrid models have also been proposed, supported by evidence (Nassar, Helmers, & Frank, 2018; Swan & Wyble, 2014; van den Berg, Awh, & Ma, 2014). Another line of work suggests that capacity limits are further limited by difficulties in WM *management*: the need to filter what information should be maintained, to bind individual items to distinct "roles" so that they can be accessed appropriately, and to prioritize information already in memory for readout (Baier et al., 2010; McNab & Klingberg, 2008; Oberauer, 2013; Oberauer & Lin, 2017; Rose et al., 2016; Soni & Frank, 2024; Todd, Niv, & Cohen, 2009).

In biologically-inspired computational models of frontostriatal WM circuitry, these operations are supported by selective *gating mechanisms*. Clusters of prefrontal neurons (or *stripes*) represent distinct addresses in memory that can be selective accessed via gating actions triggered by basal ganglia and thalamus (Calderon, Verguts, & Frank, 2022; Dayan, 2012; Frank & Badre, 2012; Kriete, Noelle, Cohen, & O'Reilly, 2013; O'Reilly & Frank, 2006; Soni & Frank, 2024). Gating mechanisms are responsible for multiple distinct aspects of WM management, including determining which items to store and retrieve, when, and from where (e.g., how to bind items to distinct roles that can be used for later retrieval). These gating mechanisms contain at least three important components (Frank & Badre, 2012; O'Reilly & Frank, 2006; Soni & Frank, 2024; Todd et al., 2009) supported by experimental evidence (Badre & Frank, 2012; Baier et al., 2010; Bhandari & Badre, 2018; Chatham, Frank, & Badre, 2014; McNab & Klingberg, 2008; Rac-Lubashevsky & Frank, 2021; Rac-Lubashevsky & Kessler, 2016; Rose et al., 2016). First, *input gating* controls whether or not given information is stored in memory, and if stored, determines the "address" (population of neurons) to which it should be written. Second, *output gating* determines when and what information to prioritize, or read out of memory, to inform a subsequent decision, such as to produce a response to a task (Chatham et al., 2014; Frank & Badre, 2012; Rose et al., 2016). Finally, working memory is *role addressable*, meaning that items can be bound to a learned task-dependent context (i.e. *role*) when stored and accessed in working memory. For example, after being introduced to a few people, one might want to retrieve the name associated with a particular face when planning to address them. In a lab setting, this might require a participant to report the color that was associated

with a particular orientation Nassar et al. (2018); van den Berg et al. (2014). When learning effective gating policies, these models afford functions such as variable binding and indirection that support rapid generalization to new situations (Bhandari & Badre, 2018; Collins & Frank, 2013; Frank & Badre, 2012; Kriete et al., 2013; O'Reilly & Frank, 2006). They also account for a variety of data implicating an network linking frontal cortex with basal ganglia and thalamus in WM gating.

Capacity limits arise in such models because as the number of items to maintain increases, it becomes increasingly difficult to learn which gating operations are most effective, leading to interference and difficulties in credit assignment Soni and Frank (2024); Todd et al. (2009). These limitations can be partially mitigated by learning to chunk multiple items in WM; such that networks with chunking abilities performed better than those allocated a larger number of "slots" with which they could conceivably store information (Soni & Frank, 2024). These simulation experiments suggested that a key factor limiting WM capacity lies in the difficulties in learning to manage WM access through filtering and binding (Soni & Frank, 2024). An implication of this conclusion is that part of WM challenges lie in management rather than maintenance *per se*, and thus that even computational architectures without any explicit limitations on maintenance may also require managing effective gating policies, or analogous processes thereof.

2 Studying WM in transformer neural networks

The Transformer architecture Vaswani et al. (2017) has recently become the dominant neural network model in artificial intelligence. Unlike some earlier AI architectures, which were inspired (albeit loosely) from human processing of language (Hochreiter, 1997) or vision (LeCun, Bengio, & Hinton, 2015), Transformers have no mechanisms designed overtly to resemble the human brain. It thus remains an open question what, if any, similarities exist, and whether there are opportunities for theory or insights from AI and neuroscience to mutually inform one another (McGrath, Russin, Pavlick, & Feiman, 2024; Russin, Pavlick, & Frank, 2025).

Recent work has explored how transformer key-value computations relate to memory systems in the brain, with a particular focus on episodic memory (Gershman, Fiete, & Irie, 2025; Whittington, Warren, & Behrens, 2021). Working memory, however, places distinct demands on a system: rather than encoding and retrieving experiences with high fidelity, it requires active, selective gating of a small set of currently task-relevant items, updated in a Markovian fashion and protected from proactive interference by earlier overlapping content (see Discussion for elaboration).

In this work, we focus specifically on WM management via gating mechanisms to meet these specific demands. Although some Transformer variants have additional built-in structure for memory (Burtsev, Kuratov, Peganov, & Sapunov, 2020; Dai et al., 2019; Y.-S. Wang, Lee, & Chen, 2019), the architec-

tures which currently dominate modern AI systems are "vanilla" (Brown et al., 2020; Touvron et al., 2023), lacking specialized components to support this type of control.

We thus investigate whether, and under what conditions, such structure can emerge as a result of training. We train vanilla Transformer models on a task from cognitive neuroscience that was designed to investigate selective gating and working memory in humans (O'Reilly & Frank, 2006; Rac-Lubashevsky & Frank, 2021). We use recent techniques from *mechanistic interpretability* (Nanda & Bloom, 2022; Olah, 2022) to expose the mechanism that the Transformer uses to perform the task. We find that, as a result of training, the self-attention mechanism specializes in a way that resembles existing models of input-output gating (§4.1), but that these mechanisms only arise when the training task places specific demands on gating that mimics biological networks (§4.2). We further find that when such mechanisms do arise, they are predictive of better task performance and of generalization to changes in the input distribution and task demands (§4.3), improving effective working memory capacity (§4.4). Our findings highlight the importance of considering the emergent mechanisms that result from training in addition to the innate architectural mechanisms when drawing comparisons between AI systems and human cognitive processes. In principle, Transformers are good candidates for learning such gating behavior. Transformers' native self-attention mechanism consists of attention heads which are arguably functionally analogous to frontostriatal stripes. The decomposition of these heads into distinct keys, queries, and values (see Appendix A.1) means that the Transformer can in principle learn to differentiate reading and writing operations in a role- and context-dependent way across its multiple heads. However, whether Transformers will use their self-attention to implement such a mechanism is an open question, especially in cases when it is possible to fit the training data using more heuristic and less generalizable solutions.

The core theoretical claim we test is that computational constraints on learning role-addressable memory management are sufficient to give rise to gating-like mechanisms, and that these constraints manifest similarly across biological and artificial systems. Human studies have shown that working memory capacity is not limited by the number of items one can maintain but rather by their effective gating strategies in frontostriatal circuits (Baier et al., 2010; McNab & Klingberg, 2008; Vogel, McCollough, & Machizawa, 2005). Theoretical work has shown that capacity limits in these circuits do not stem from a limitation in the number of available neural populations, but rather result from a credit assignment problem that arises when learning to manage multiple items in memory (Soni & Frank, 2024; Todd et al., 2009). Moreover, these limitations can be partially mitigated by learning to reuse effective gating policies (Soni & Frank, 2024).

Note that an implication of this view is that artificial systems facing these management demands should be pressured toward discovering analogous gating solutions even if they have no demands on memory maintenance or capacity. We test this prediction in vanilla Transformers, which have no explicit gating machinery and no meaningful memory capacity constraints (since all informa-

tion is available in the context window), making them an ideal test case for isolating management and learning demands from storage demands. We derive three specific predictions. First, if gating emerges from management demands rather than architectural bias, it should only arise when the training task requires role-addressable storage and retrieval (and conversely, it should fail to emerge under control task conditions that remove these demands). Second, if the learned mechanisms are genuinely gating-like (rather than other heuristic solutions), they should support generalization to out-of-distribution inputs and transfer to tasks with higher working memory demands. Third, echoing findings from frontostriatal models, increasing the number of items to be managed should produce capacity-like limitations even in a system with no storage constraints, and these limitations should be partially mitigated by prior learning of effective gating policies.

The results support all three predictions, suggesting that gating is a computational response to the demands of role-addressable working memory management.

3 Experimental Design

3.1 Tasks

Reference-Back 2 Task: We use a variant of a task from cognitive neuroscience known as the "reference-back 2" task (Rac-Lubashevsky & Frank, 2021). This is one among a number of task designs inspired by frontostriatal neural networks (O'Reilly & Frank, 2006; Soni & Frank, 2024) which requires selective updating and accessing of information in a role-addressable manner. In the reference-back paradigm, symbols are viewed one at a time with associated roles, and the participant must determine whether the current symbol is the same or different as that stored in memory for a given role. For example, a sequence might contain letters (role 1) and numbers (role 2), each of which occurs along side an update instruction which is either `Store` or `Ignore`. For each symbol in the sequence, the participant must do two things: 1) make a same/different judgment based on whether the current symbol matches the previously-stored symbol for that role, and 2) if the update instruction is `Store`, update the symbol associated with the associated role. See Figure 1 for an example. We create a modified text-based version of the reference-back 2 task suitable for the transformer. In our design, roles are denoted explicitly using special tokens (i.e., either `Reg0` or `Reg1` for the two-role version) indicating the role (or "register") to which the symbol should be bound. See Appendix A.2 for more details about our task implementation.

Split-Set Control: The computational advantages of role-addressable gating in frontostriatal networks (relative to other recurrent neural networks) are particularly evident when any symbol can be assigned to any register, so that the networks have to learn to assign them separable addresses (O'Reilly & Frank,

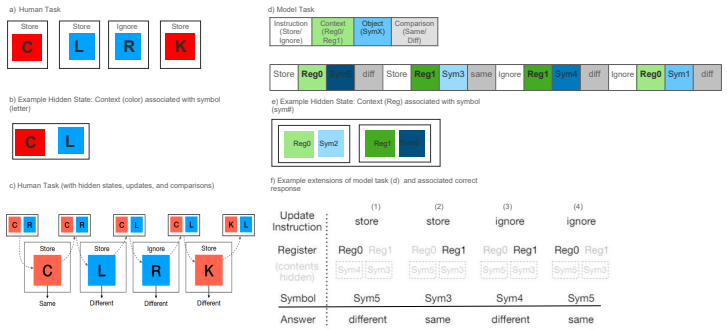

Figure 1: **Task** The reference-back-2 task requires making same vs. different judgments for each symbol in a sequence by comparing against a previously-shown symbol. See text for description of the task. a) Four example trials in the human task. In this example, the colors blue and red serve as contexts (roles). The letters are the associated information to be stored and retrieved from working memory. An additional instruction indicates whether the content of memory should be stored (updated) or ignored. b) The hidden state maintains each context and associated letter. c) The arrows show how the letters are updated and compared with the hidden states. Note that even in an "ignore" trial, the task requires comparing the current item to that in memory for correct responding. d) The model task is presented with the instruction, context (register), associated symbol and the comparison response. e) Corresponding hidden state that the model should represent to associate a register (context/role) to symbol. f) Four example trials where model has to compare hidden state symbols to incoming stimuli and update memory based on instruction. Model correctness is determined by if model response (same/different) matches expected value.

2006). To test the hypothesis that gating mechanisms only emerge in response to such task demands, we created a "split set" control condition in which the registers are associated with disjoint sets of symbols. In this task, it is possible to succeed without role-addressable gating, since symbols never need to be decoupled from their associated registers. This condition also serves to demonstrate that attentional mechanisms within transformer models do not by definition implement selective gating, but only do so when incentivized by task demands.

Ignore-Integrated and Ignore-Separated Split-Set Controls: In the same vein, we include a further simplified task variant which is designed to require selective input gating but not output gating. This condition, along with the split-set control, have been previously used to show that the advantage of frontostriatal gating networks relative to other recurrent networks is largely reduced in such scenarios (O'Reilly & Frank, 2006).

Specifically, in the *ignore separated* condition (referred to as split-set control

earlier), each register sees a disjoint set of symbols. This separation also applies to the `Ignore` update instruction, such that the model could in principle learn to Ignore each of the symbols paired with the `Ignore` instruction and does not need to learn to differentiate `Store` and `Ignore` in a meaningful way. In contrast, in the *ignore integrated* condition, symbols under an `Ignore` instruction will be in distribution with the chosen register and hence it will sometimes have to attend to those symbols (when they are paired with `Store`). Thus in this case, the model must distinguish between a `Store` and `Ignore` instruction and thus learn input gating, but does not have to learn an output gating policy because both registers have mutually exclusive symbols.

3.2 Models

Our focus was to understand and interpret attentional mechanisms of the models and therefore we used a minimal model that was able to learn the task. We train small, attention-only, decoder-only Transformer models from scratch on our task. Our models contain two decoder-only layers, each with two heads, and no multilayer perceptrons or layer normalization, followed by a linear "unembed" layer. The models are trained on 100k training data points for 60 epochs. See Appendix A.3 for additional details.

3.3 Metrics

Performance on Refback2 Task: We evaluated how well the model performs on the task on which it was trained using standard accuracy on the same vs. different prediction for each symbol in the sequence.

Input and Output Gating: The refback2 task is assumed to benefit from gating mechanisms, but success on the task is not in and of itself diagnostic of having learned the gating mechanisms. To develop an intrinsic measure of the input and output gating mechanisms, we use path-patching (Goldowsky-Dill, MacLeod, Sato, & Arora, 2023; K. Wang, Variengien, Conmy, Shlegeris, & Steinhardt, 2022), a generalization of causal mediation analysis (Pearl, 2001) that allows us to determine which components of a neural model (e.g., attention heads) work together in order to produce observed behavior on a task. Path patching involves designing a minimal pair of inputs, a "clean" input and a "corrupted" and then finding specific components of a model which fully account for the difference in the model's output between the two cases. By "patching in" only these components, the model can be made to behave as though it is seeing the corrupted input even when it is in fact seeing the clean input. See Appendix A.4 for more details.

We use path patching as a measure of input gating in the following way. For input gating, we design a minimal pair of inputs in which the corrupted copy includes `Ignore` in a place where the clean copy contained a `Store`, or vice-versa. We then use path patching to identify an incisive edit that can be made to the activations of the model in order to prevent the model from storing ("gating

in") a given symbol. We report the accuracy of the input gating mechanism as the percentage of inputs on which this edit to the weights changes the models behavior in the expected way.

Analogously for output gating, we design minimal pairs that we expect to yield differences in an output gating mechanism, assuming that one exists and is functioning correctly. Specifically, our clean and corrupted sequences differ in the register associated with one of the symbols, which should trigger a difference in what information is read ("gated out") in order to make a final same vs. difference judgment. Again, we report the accuracy of the output gating mechanism as the percentage of cases on which it is possible to make such an edit and produce the desired effect.

Our description of results in Section 4 elaborates on both input and output gating metrics and their interpretation.

4 Results

4.1 Key and Query Vectors Specialize for Input and Output Gating

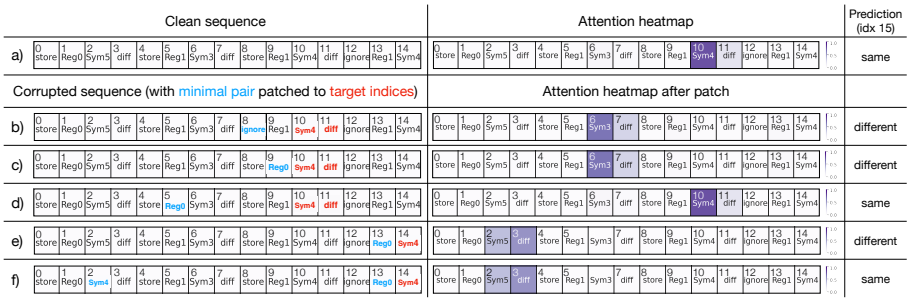

Figure 2: **Path Patching Examples.** Model behavior across different path patching conditions. Attention is visualized as a shade of purple, with deeper shade corresponding to higher attention to that token. We create "corrupted" minimal pairs in which changing a token in the input (light blue) either changes the correct label at index 15 (examples b, c, e) or does not (d, f). We make small path-patching edits with the minimal pair to targeted network components (layer 1 keys for b, c, d, f; queries for e,f, see text). In all test examples, making the small patch successfully alters the model's prediction to align with the "corrupted" example, as expected.

We find that Transformers implement a role-addressable gating mechanism whereby key vectors control input gating and query vectors control output gating (see Appendix A.1 for summary of key, query, value attention, and Discussion for elaboration on how key and query vectors work together in the service of input and output gating). Specifically, after training, the key for Sym_i (e.g., at

tokens 2, 6, 10, and 14 in Fig. 2a) contains information about the combination of the update instruction, register, and symbol for position i. A query's ability to address this position depends on whether the represented tuple contains a `Store` or an `Ignore`. Indeed, key vectors representing an `Ignore` tuple receive very little attention (0.4% of layer 1 attention averaged over test set), whereas those representing a `Store` tuple receive the bulk of the attention (86.8%). We demonstrate the above narrative using path patching, described in Section 3.3), and further below. Figure 2 shows a summary of the path patching experiments and results. Our task is simple and does not contain noise, so in all cases, the intervention (i.e., patching to the keys or queries) results in a 100% change in the model prediction in the expected direction. Thus, for compactness, Figure 2 depicts the conditions but does not include quantitative results.

First, to investigate input gating, we create clean sequences sampled from our test set, and then corrupt these sequences by switching a `Store` within tuple i to an `Ignore`. We path-patch only the key vectors of i. We expect, if the key controls input gating, that patching these key vectors to those associated with an `Ignore` instruction should "block" attention to all of tuple i. An example attention pattern is in Fig. 2, panels a and b. We find that the model's attention shifts away from the tuple accordingly in 100% of patched instances. The presence of an `Ignore` or a `Store` within a tuple controls whether the key construction acts as an open input gate or a closed input gate.

Key construction also depends on the *role* of the represented content; within our task, the role corresponds to whether the Reg_i is Register 0 or Register 1. When making a same/different prediction, key vectors representing a tuple that matches the target register receive most of the model's attention (92.5% of total attention), whereas those that do not match are not attended to (3.3% of total attention). Again, we use path-patching to causally determine whether key construction encodes roles, this time perturbing the target register rather than the update instruction (see Fig. 2, row c). We hypothesized that this perturbation would be akin to altering the address in which an item is input gated. Indeed, the model's attention shifts away accordingly across every example in the test set. For example, in panel c), when Reg 1 is corrupted to Reg 0 at position 9, the model's attention switches to the earlier symbol ($Sym3$ at position 6) because it is now the most recent item associated with Reg1 in the final query, leading to a "different" response. Note that the stored tuple must be modified; if the same corruption is made earlier (as in row d), attention does not shift (because Reg 1 is later updated anyway). This behavior shows that the gating within self-attention is *role-addressable*; the registers within the task function as roles, and are embedded within the key vectors as part of the representation.

Given that key vectors serve the role of addresses, query vectors in turn control which key vectors are accessed, through the final Q*K dot product in attention. Query construction thus performs the role of output gating within Transformers. The query composition controls which addressable $Symboli$ representations are attended to based on the identity of the target register. We again determine this through a set of path-patching experiments in which we perturb the target register (Fig. 2, row e).

We find that patching to the query vector in such cases indeed causes the attention to shift from the original stored tuple (74.1% of attention) to the stored tuple that matches the edited register, resulting in a corresponding change in the final same/different judgment. Editing aspects of the target tuple other than target register has minimal effect on the query construction. No edits to the query cause the model to attend to an `Ignore` tuple, further evidencing of output gating behavior–only content that has been made "addressable" can be accessed for a response. Furthermore, we find that the target instruction and symbol do not factor into the query composition– changing them through path-patching to the query does not affect attention. This is notable because the model could employ other strategies for determining which tuples are eligible to be the stored tuple; e.g. attending to all symbols to match if any of them are the same as the target symbol.

4.2 Emergence of Gating Policies Depends on Task Demands

Under which conditions to gating computations arise? One possibility is that they are a trivial consequence of the Transformer's architecture. After all, gating is sometimes expressed as a simple multiplicative operation, as in LSTMs (Hochreiter, 1997). And indeed, the key, query, and value vectors underlying the attention heads are combined via matrix multiplications. However, this alone does not ensure that the networks learn effective, role-addressable gating policies. We thus investigate whether the training task demands induce such gating policies by running several control experiments (§3.1) which simplify the task, removing the role addressability of gating requirements. We thus hypothesized that since this task is much easier to learn, the Transformer will learn an overly memorized, brittle solution and will not develop effective input and output gating strategies.

As shown in Figure 3, networks trained on the split set control task do not perform well at either the input or the output gating subtasks (see §3.3 for gating metrics). Moreover, models trained on the ignore-integrated task perform well at the Store vs Ignore input gating tasks, but not the output gating task, as expected, while models trained on the ignore-separated task perform poorly on both input and output gating (3b). These results strongly support the intuition that while Transformers have good inductive biases for learning role-addressable gating, it does not come "for free", and emerges only when demanded by the training task – the same contrast in demands that demonstrated advantages of frontostriatal gating (O'Reilly & Frank, 2006).

4.3 Emergence of Gating Predicts Task Performance

Is learning this gating policy useful for succeeding on the task? To answer this question, we first compare models with the same hyperparameters across different random seeds. We train 20 new models, each with a different random initialization, and measure both training loss and test set accuracy. 5 of the

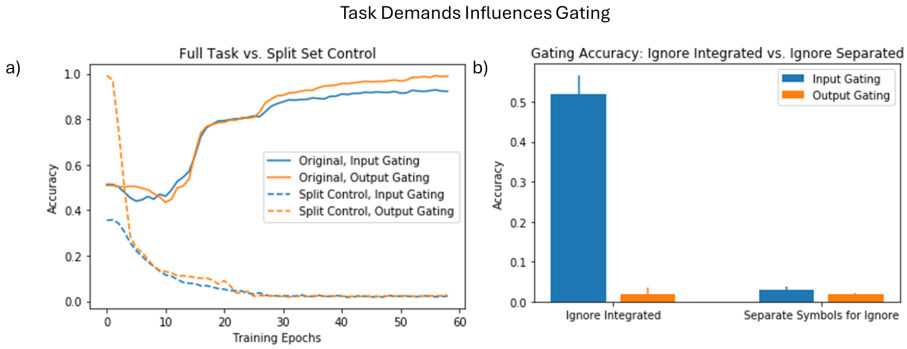

Figure 3: **Split Set Control** The split set control (a.k.a ignore segregated) uses a disjoint set of symbols for each register and a disjoint set of symbols for the ignore condition. This is the most simplified version of the task as the network does not need to distinguish between store vs. ignore instructions and thus does not require input gating (it just needs to learn to always ignore some symbols which is easily accomplished with standard weight learning mechanisms). It also does not require output gating because it never has to assign the same symbol to different registers, so it does not have to flexibly perform variable binding. In the ignore integrated condition, the registers have disjoint symbol sets, but the ignore condition pulls from any of the register symbol sets. This task should require input gating to distinguish store from ignore instructions, but not necessarily how to differentially read out items from the registers (output gating). a) Compared to the full task where both input and output gating are learned through training, there is no development of either gating strategy in the split set control (ignore separated). b) When ignore symbols are integrated with store symbols, the models must learn to respond appropriately to store/ignore instructions (input gating). Indeed, the network does learn the input gating subtask for the ignore integrated but not ignore separated conditions. Neither condition shows any learning of output gating since each register has its own mutually exclusive symbol set.

models succeed 100% of the time, and the other 15 models succeed between 94%-99.99% of the time, with a mean of 97.72% and a standard deviation of 2.03. We measure the intrinsic quality of the input and output gating subtasks using the path patching metrics described in Section 3.3.

Figure 4 shows the 5 runs that reach 100% accuracy on the test data as well as 5 randomly selected runs that do not. Two trends stand out. First, models which score a perfect test accuracy appear to succeed at the gating subtasks more readily than models which do not. Of the former, 3 of 5 models reach 100% accuracy on both subtasks readily, plateauing less than halfway through training. In contrast, models that make errors in the test set also do not reach such immediate success at the subtasks (including the 10 not pictured in this graph); in fact, many categorically fail, scoring as low as 49%

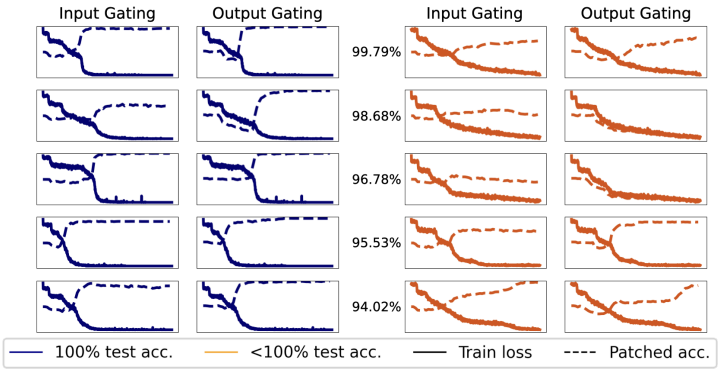

Figure 4: **Model Performance** over training on patching subtasks. Each graph contains an individual model's training loss (solid line) and subtask accuracy (dashed line, between 0 and 1) over time; the line's color corresponds to whether the model reaches 100% accuracy on the general test set.

accuracy. These results do not indicate that this class of models' representations are useless for the task– they all score between 94% and 99.99%, well above chance performance.

The second trend is that many models across both classes have a sharp decline in training loss, which correlates with a similarly steep increase in accuracy on both subtasks. We interpret this phase transition as suddenly learning a gating mechanism. Models that do not exhibit phase transitions to the same degree take longer to fit the task, and do not reach high subtask accuracy.

While the above results suggest it is possible for a model to achieve fine accuracy in distribution without learning a gating policy, we hypothesize that models which do learn a more general gating strategy will better generalize to out of distribution examples. To assess this, we held out a subset of symbols from each register (randomized which symbol would be held out from which register). We tested these models on a challenge dataset which included examples of in-distribution pairings (register-symbol pairings that were seen in training) and out-of-distribution pairings (register-symbol pairings that were not seen during training). When holding out 5% of the symbols, the models learn a robust input and output gating strategy, and notably, perform on average at 99.7% for out-of-distribution symbol-register pairings. In contrast, the models trained on the split set controls do not learn robust gating strategies (despite performing perfectly at the trained task) and accordingly perform more poorly on the out of distribution examples: 77.2% (ignore integrated) and 88.6% (ignore separated).[1]

[1]One concern is that in the 5% held out, each register is trained on 49 symbols while in the split set (ignore integrated), each register is trained on 25 symbols. While all other parameters are held constant, this difference might be big enough to account for the large difference in generalization. To account for this, we ran another set of simulations holding out 60% of the symbols (each register is trained on 35 symbols). These models robustly learn a gating policy. Even with a drastic increase in the number of held out symbols from 5% to 60%, the

Experiment	Out of Distribution	In Distribution
5% Held Out	99.7%	99.8%
60% Held Out	95.3%	99.3%
Split Set (Ignore Integrated)	77.2%	94.6%
Split Set (Ignore Separated)	88.6%	80.8%

Table 1: **Accuracy for In Distribution and Out of Distribution Symbol-Register Pairings in Challenge Dataset** Experiments and their associated performance on the challenge data set. The challenge dataset includes a mix of in-distribution and out-of-distribution symbol-register pairings. This table breaks down the accuracy based on if the symbol-register pairing was in the training (In-Distribution) or not seen during training (out-of-Distribution).

Note that the Split Set (Ignore Separated) task depends least on learning an effective gating strategy. These models learn the task very quickly and show no ability to perform on the gating subtask, and instead likely learn heuristic solutions. Interestingly, when tested on the challenge dataset (which includes examples of both in distribution and out of distribution register-symbol pairs), these models show a large disruption in their ability to perform on in-distribution sequences (80.8%). This insinuates that the strategy learned by these models is highly dependent on memorization and by adding in new elements, the strategy fails. The out-of- distribution accuracy is slightly higher than the in-distribution accuracy and future work could try to better understand the effects of perturbing a brittle model.

4.4 Gating Policy Transfers to Increased Task Demands

Thus far, the experiments have focused on tasks with two registers, mimicking that used in the reference back-2 task (Rac-Lubashevsky & Frank, 2021). However, human working memory has a capacity of about 3-4 items (Cowan, 2008), albeit with vigorous debates questioning whether this limit is discrete or continuous (Luck & Vogel, 2013; Wei, Wang, & Wang, 2012; Zhang & Luck, 2008). More recent models and data suggest a "chunking" hybrid between the two, whereby multiple memoranda can compete for shared continuous resources within discrete slots (Nassar et al., 2018; Soni & Frank, 2024). Chunking increases *effective* capacity, allowing more items to be remembered at the cost of precision of some of the items. Notably, in frontostriatal gating models, when the number of registers to manage was larger than two, networks given limited memory allocation but with chunking capabilities performed better than those that were allocated as many PFC populations as items to store (Soni & Frank, 2024). The reason for this seeming paradox is *credit assignment*: as the number of PFC populations (stripes) increases, the gating management problem becomes more challenging – the network has to learn to route each item to

out-of-distribution accuracy is still very high compared to either of the split set controls.

distinct populations and to also learn to read out from the corresponding population for a given probe. Moreover, these learning problems are interdependent.

These limitations are computational rather than anatomical and stem from the learning process. We hypothesize that they would also manifest in Transformers, even though Transformers have no inherent memory demands (since all information is available in the context window). Specifically because of the flexibility and expressivity of Transformers, we predicted that by increasing task demand, the network would learn the task, but struggle to do so. We further predict that models that learn mechanistic solutions will generalize better to new tasks.

To test this hypothesis, we increased the number of registers from 2 to 3. First, we confirmed that asymptotic accuracy dropped to 95.4%. This result is qualitatively the same even when training for twice the number of epochs. This is non-trivial given that we are just adding one register. We predicted that the degree to which the transformer solves the task is related to heuristics and memorization rather than gating in these cases. We predicted that if we first pretrained the model to effectively manage two registers, the network will be able to scaffold the learned gating strategy to learn the three register task more robustly. We further predicted that this pretraining would only be useful if pretraining encouraged gating strategies. Our results in Figure 5 support both of these conclusions. Not only do pretrained networks on the original reference back-2 task exhibit higher gating sub-task accuracy 5, but networks that were pretrained with the original two register task showed very rapid learning in the three register task in the first few epochs. Moreover, this pretraining was far less effective when it did not encourage gating (the split symbol control from above).

5 Summary and Discussion

In this work, we investigate Transformer models for emergence of a learned *gating mechanism*; a network component performing role-addressable gating, similar to that in working memory of humans. We observe that the model learns a gating policy and find that task performance is correlated with gating ability. Our results show how learning gating mechanisms is one way Transformers can excel at tasks that require executive function.

The Transformer models are capable of making use of key composition for input gating and query composition for output gating on the task. We find that making precise corruptions to specific architectural elements of the network causes the model's prediction to change from *Same* to *Different* or vice versa, indicating that those components are causally responsible for the gating mechanism. The architectural biases of attention within the vanilla Transformer model lend themselves well to representing role-addressable content. The learnable nature of keys, queries, and values allows the model to learn to create internal representations. These representations can be learned to signify roles and addresses, mimicking the variable binding and input / output gating mech-

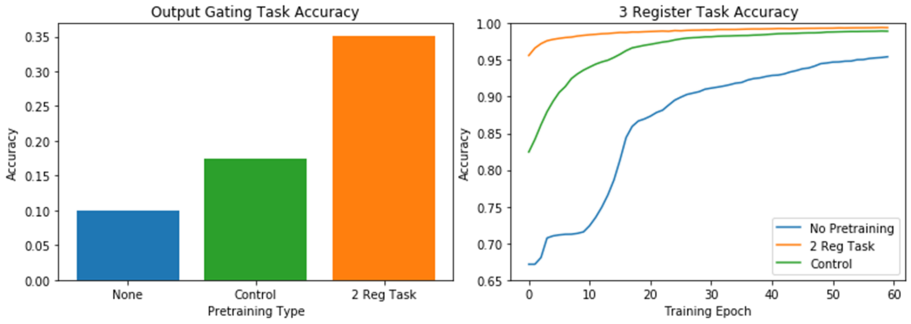

Figure 5: **3 Register Task and Pretraining** a)At the end of training on the 3 Register Task, models with 2 Register Task pretraining perform the best on output gating tasks, insinuating learning of a mechanistic solution. b) Accuracy curves through training (after the pretraining) show a stark difference between pretraining on the normal 2 register task and the control task. Models pretrained on the 2 register task, generalize quickly (first few epochs) and with high accuracy (above 95%) on this new task. At the end of training, no pretraining models are below 95% accuracy - which is below the accuracy that 2 register task pretraining models showed at the beginning of training. Control pretraining models show some generalization (due to learning of basic elements e.g. symbols) but do not generalize to the same degree. Models pretrained with the 2 register tasks, which should encourage a mechanistic solution, perform well at the new 3 register task, showing a superior ability to generalize. 22 random seeds were run for each experimental condition and the results here an average of those models. See Appendix A.5) for comparison on input and output gating task accuracies.

anisms in biological neural networks (Collins & Frank, 2013; Frank & Badre, 2012; Kriete et al., 2013; O'Reilly & Frank, 2006).

Our finding that input and output gating were mediated by keys and queries might seem surprising, given that the two vectors always work in tandem in a Transformer. And indeed, the key-query dot product is required for attention to operate, and in that sense both are always involved. But this is actually analogous to the relationship between input and output gating in biological systems: you cannot output gate something that was never input gated in the first place. The two operations are jointly necessary yet functionally dissociable — and our path-patching experiments directly support this dissociation. When we patch only the key vectors of a tuple, attention shifts away from that tuple, showing that keys control addressability. When we patch only the query vectors (switching the target register), attention shifts to a different stored tuple, showing that queries control which address is accessed. These are distinct interventions with distinct functional consequences, even within the same attention head.

When we trained more models on the task, we found that the models which perform best on the task correlate with the markers of gating we observed in

our circuit analysis, and that the learning trajectory shows a steep decrease in training loss and a steep rise in patched subtask accuracy simultaneously, suggesting that the model has learned a gating policy at that time. Both findings are analogous to those of Frank and Badre (2012), in which they find that networks which learned a hierarchical gating policy performed better at a hierarchical learning task, and humans that learn this policy also show a sharp decrease in loss when they discover it. There is still more work to be done to better understand the models that don't learn the gating policy: what types of solutions do they learn? How do we push models towards a mechanistic solution by hyperparameter tuning? Understanding grokking phase planes in a similar manner as Liu et al. could be informative.

Nevertheless, we show that a critical factor controlling the learned gating policy is the task demands. We found that experimental conditions in which biological networks exhibit gating advantages were similarly needed to give rise to learned gating policies in Transformers. Conversely, when the task places less demands on role-addressable gating (our split symbol control conditions), the models 1) did not learn gating policies, 2) were less able to generalize to out-of-distribution pairs, and 3) were less able to rapidly acquire tasks with higher gating demands (three registers).

We show that models that learn and solve the tasks using gating mechanisms are better at generalization. Further we show how learning brittle and memorized solutions causes some models to falter at even in-distribution pairings when they are mixed with out-of-distribution examples. These results can be used to inform larger models that are needed for better and faster generalization. Future work can characterize what kinds of mistakes are common within the mechanistic and heuristic models to further characterize these models and promote better generalizability.

Ultimately, finding connections between emergent behavior of Transformer models and human working memory serves to benefit both computational cognitive neuroscience and artificial intelligence. Although Transformer models themselves are limited in their biological plausibility, in this setting they learned behavior mimicking the functionality of working memory, and their application within computational models of the brain should be further explored. From the perspective of artificial intelligence, understanding the strengths and limitations of Transformer models on executive function tasks may inform model analysis across the many diverse settings in which these models are applied.

5.1 Related work

Our findings complement recent work relating transformer key-value computations to hippocampal episodic memory (Gershman et al., 2025; Whittington et al., 2021). While episodic memory is characterized by high capacity, it is vulnerable to proactive interference that is overcome by working memory (e.g., Kane and Engle (2000)). Working memory, by contrast, operates under a fundamentally different set of demands. It has famously limited capacity, but is specifically adapted for tasks requiring flexible, active management of currently

task-relevant information — deciding what to store, when, and where, and crucially, how to access it without suffering interference from earlier items that overlap in content or role. In working memory, the system must learn to update stored content in a Markovian fashion: what matters is the current state of each register, not the full history of what has been stored there. This places particular demands on selective gating — both for filtering what enters memory (input gating) and for selectively retrieving the right item at the right time (output gating). The emergence of gating policies only under task conditions that demand role-addressable storage and retrieval — and their failure to emerge under simplified conditions — speaks to challenges specific to working memory management, beyond what a standard key-value retrieval mechanism needs to solve. Together, these lines of work suggest that transformer attention may provide a general substrate for multiple forms of memory, with the specific mechanisms that emerge shaped by the demands of the task on which the system is trained.

Memory-augmented neural networks such as Neural Turing Machines (Graves, Wayne, & Danihelka, 2014) and Differentiable Neural Computers (Graves et al., 2016) feature explicitly designed read/write gating mechanisms, and like the hippocampal key-value memory frameworks discussed above, their large external memory matrices and content-based addressing make them more analogous to episodic memory systems than to working memory — optimized for high-capacity storage and retrieval rather than for flexible Markovian updating of a small set of actively maintained items. Our work addresses a complementary and distinct question: can gating emerge in a system with no such machinery, purely from the demands of the task? Vanilla Transformers are well suited for this test precisely because they lack explicit gating and, for the sequence lengths used here, face no meaningful memory capacity constraints. This allows us to isolate the contribution of management and credit assignment demands from memory storage demands per se — directly testing the theoretical claim from the PBWM literature that capacity limits in working memory arise not from storage limitations but from the difficulty of learning role-addressable gating policies (Soni & Frank, 2024; Todd et al., 2009). The emergence of gating-like mechanisms in Transformers only under task conditions that demand role-addressable storage and retrieval — and their failure to emerge otherwise — suggests that such mechanisms reflect a fundamental computational solution to the challenges of working memory management rather than merely a useful engineering choice.

6 Limitations

While this paper draws parallels between the Transformer architecture and the brain, it is important to emphasize that there are some significant differences between how Transformers and humans solve tasks. In particular, because Transformers can attend to any part of the sequence when creating a representation, they are not limited by memory constraints. Transformers can solve tasks that would push the limits of human working memory (but it should be noted that Transformers require disproportionately large amounts of training data to do

so). Nevertheless, we hypothesized that Transformers might still learn effective gating policies that mimic those in frontostriatal networks. Moreover, as briefly reviewed in the introduction, working memory capacity in humans and biological computational models is not limited primarily by the memory demand *per se*, but rather by the difficulty of the credit assignment problem for learning how to manage role-addressable storage and access of multiple items in memory. Thus it is nevertheless significant that we find that Transformers also had to confront a credit assignment problem, in that it was inefficient at learning the 3 register task unless it had been pre-trained to learn a gating policy within the 2 register task – despite an effectively unlimited memory capacity.

The present work focuses on a single task — the reference-back-2 paradigm — chosen because it was specifically designed to place demands on role-addressable gating motivated by similar work in PBWM, and has well-characterized behavioral and neural correlates in humans. Whether similar gating mechanisms would emerge in Transformers trained on other working memory paradigms, such as n-back, delayed match-to-sample, or AX-CPT variants, remains an open question. We note that the original PBWM work demonstrated advantages of gating mechanisms across a family of such tasks (O'Reilly & Frank, 2006), and we expect the same principles would generalize, but systematic demonstration of this is beyond the scope of the current paper. More broadly, we suspect that the most robust and flexible gating policies would emerge from training on a curriculum spanning a diverse range of tasks that collectively demand working memory management — a direction we leave for future work (but see previous work on how such curriculum learning can induce generalizable strategies that mimic aspects of human behavior that likely relate to WM (Russin et al., 2025)). Future work could also assess how other factors such as sequence length and task structure affect the emergence of gating in networks. While we did test these models on novel symbol-register pairings, future work could work to test the models on explicitly novel symbols. The present findings are intended as a proof of concept: establishing that gating can emerge in a system with no explicit gating machinery, that its emergence depends critically on task demands, and that when it does emerge it supports better generalization and transfer to novel task conditions.

References

Badre, D., & Frank, M. J. (2012). Mechanisms of hierarchical reinforcement learning in cortico–striatal circuits 2: Evidence from fmri. *Cerebral cortex*, *22*(3), 527–536. doi: https://doi.org/10.1093/cercor/bhr117

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. doi: https://doi.org/10.48550/arXiv.1409.0473

Baier, B., Karnath, H. O., Dieterich, M., Birklein, F., Heinze, C., & Müller, N. G. (2010). Keeping memory clear and stable - the contribution of

human basal ganglia and prefrontal cortex to working memory. *Journal of Neuroscience*, *30*. doi: 10.1523/JNEUROSCI.1513-10.2010

Bhandari, A., & Badre, D. (2018). Learning and transfer of working memory gating policies. *Cognition*, *172*, 89–100. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0010027717303037` doi: https://doi.org/10.1016/j.cognition.2017.12.001

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... others (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901. doi: https://doi.org/10.48550/arXiv.2005.14165

Burtsev, M. S., Kuratov, Y., Peganov, A., & Sapunov, G. V. (2020). Memory transformer. *arXiv preprint arXiv:2006.11527*. doi: https://doi.org/10.48550/arXiv.2006.11527

Calderon, C. B., Verguts, T., & Frank, M. J. (2022). Thunderstruck: The acdc model of flexible sequences and rhythms in recurrent neural circuits. *PLoS Computational Biology*, *18*(2), e1009854. doi: https://doi.org/10.1371/journal.pcbi.1009854

Chatham, C. H., Frank, M. J., & Badre, D. (2014). Corticostriatal output gating during selection from working memory. *Neuron*, *81*(4), 930–942. doi: https://doi.org/10.1016/j.neuron.2014.01.002

Collins, A. G., & Frank, M. J. (2013). Cognitive control over learning: creating, clustering, and generalizing task-set structure. *Psychological review*, *120*(1), 190. doi: https://doi.org/10.1037/a0030852

Cowan, N. (2008). *Chapter 20 what are the differences between long-term, short-term, and working memory?* (Vol. 169). doi: 10.1016/S0079-6123(07)00020-9

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*. doi: https://doi.org/10.18653/v1/P19-1285

Dayan, P. (2012, December). How to set the switches on this thing. *Current Opinion in Neurobiology*, *22*(6), 1068–1074. Retrieved from `http://dx.doi.org/10.1016/j.conb.2012.05.011` doi: 10.1016/j.conb.2012.05.011

Frank, M. J., & Badre, D. (2012). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: computational analysis. *Cerebral cortex*, *22*(3), 509–526. doi: https://doi.org/10.1093/cercor/bhr114

Gershman, S. J., Fiete, I., & Irie, K. (2025). Key-value memory in the brain. *Neuron*, *113*. doi: https://doi.org/10.1016/j.neuron.2025.02.029

Goldowsky-Dill, N., MacLeod, C., Sato, L., & Arora, A. (2023). Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*. doi: https://doi.org/10.48550/arXiv.2304.05969

Graves, A., Wayne, G., & Danihelka, I. (2014). *Neural turing machines.* arXiv. Retrieved from `https://arxiv.org/abs/1410.5401` doi: 10.48550/ARXIV.1410.5401

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-

Barwińska, A., ... Hassabis, D. (2016, October). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471–476. Retrieved from http://dx.doi.org/10.1038/nature20101 doi: 10.1038/nature20101

Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*. doi: https://doi.org/10.1162/neco.1997.9.8.1735

Kane, M. J., & Engle, R. W. (2000). Working-memory capacity, proactive interference, and divided attention: Limits on long-term memory retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *26*(2), 336–358. Retrieved from http://dx.doi.org/10.1037/0278-7393.26.2.336 doi: 10.1037/0278-7393.26.2.336

Kriete, T., Noelle, D. C., Cohen, J. D., & O'Reilly, R. C. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, *110*(41), 16390–16395. doi: https://doi.org/10.1073/pnas.1303547110

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444. doi: https://doi.org/10.1038/nature14539

Liu, Z., Kitouni, O., Nolte, N., Michaud, E. J., Tegmark, M., & Williams, M. (2022). Towards understanding grokking: Aneffective theory of representation learning. *NeurIPs*. doi: https://doi.org/10.48550/arXiv.2209.11143

Luck, S. J., & Vogel, E. K. (2013). *Visual working memory capacity: From psychophysics and neurobiology to individual differences* (Vol. 17). doi: 10.1016/j.tics.2013.06.006

Ma, W. J., Husain, M., & Bays, P. M. (2014). *Changing concepts of working memory* (Vol. 17). doi: 10.1038/nn.3655

McGrath, S. W., Russin, J., Pavlick, E., & Feiman, R. (2024). How can deep neural networks inform theory in psychological science? *Current Directions in Psychological Science*, 09637214241268098. doi: https://doi.org/10.1177/09637214241268098

McNab, F., & Klingberg, T. (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience*, *11*. doi: 10.1038/nn2024

Nanda, N., & Bloom, J. (2022). *Transformerlens*. https://github.com/neelnanda-io/TransformerLens.

Nassar, M. R., Helmers, J. C., & Frank, M. J. (2018). Chunking as a rational strategy for lossy data compression in visual working memory. *Psychological Review*, *125*. doi: 10.1037/rev0000101

Oberauer, K. (2013). The focus of attention in working memory—from metaphors to mechanisms. *Frontiers in Human Neuroscience*. doi: https://doi.org/10.3389/fnhum.2013.00673

Oberauer, K., & Lin, H.-Y. (2017). An interference model of visual working memory. *Psychological Review*. doi: https://doi.org/10.1037/rev0000044

Olah, C. (2022). *Mechanistic interpretability, variables, and the importance of interpretable bases*. https://www.transformer-circuits.pub/2022/mech-interp-essay.

O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, *18*(2), 283–328. doi: https://doi.org/10.1162/089976606775093909

Pearl, J. (2001). Direct and indirect effects. In *Proceedings of the seventeenth conference on uncertainty in artificial intelligence* (p. 411–420). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rac-Lubashevsky, R., & Frank, M. J. (2021). Analogous computations in working memory input, output and motor gating: Electrophysiological and computational modeling evidence. *PLoS computational biology*, *17*(6), e1008971. doi: https://doi.org/10.1371/journal.pcbi.1008971

Rac-Lubashevsky, R., & Kessler, Y. (2016). Dissociating working memory updating and automatic updating: The reference-back paradigm. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *42*(6), 951. doi: https://doi.org/10.1037/xlm0000223

Rose, N. S., LaRocque, J. J., Riggall, A. C., Gosseries, O., Starrett, M. J., Meyering, E. E., & Postle, B. R. (2016, December). Reactivation of latent working memories with transcranial magnetic stimulation. *Science*, *354*(6316), 1136–1139. Retrieved from `http://dx.doi.org/10.1126/science.aah7011` doi: 10.1126/science.aah7011

Russin, J., Pavlick, E., & Frank, M. J. (2025, August). Parallel trade-offs in human cognition and neural networks: The dynamic interplay between in-context and in-weight learning. *Proceedings of the National Academy of Sciences*, *122*(35). Retrieved from `http://dx.doi.org/10.1073/pnas.2510270122` doi: 10.1073/pnas.2510270122

Soni, A., & Frank, M. J. (2024). Adaptive chunking improves effective working memory capacity in a prefrontal cortex and basal ganglia circuit. *eLife*, *13*, RP97894. Retrieved from `https://elifesciences.org/reviewed-preprints/97894` doi: https://doi.org/10.7554/eLife.97894

Swan, G., & Wyble, B. (2014). The binding pool: A model of shared neural resources for distinct items in visual working memory. *Attention, Perception, and Psychophysics*, *76*. doi: 10.3758/s13414-014-0633-3

Todd, M. T., Niv, Y., & Cohen, J. D. (2009). Learning to use working memory in partially observable environments through dopaminergic reinforcement. In *Advances in neural information processing systems 21 - proceedings of the 2008 conference.*

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... others (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. doi: https://doi.org/10.48550/arXiv.2302.13971

van den Berg, R., Awh, E., & Ma, W. J. (2014). Factorial comparison of working memory models. *Psychological Review*, *121*. doi: 10.1037/a0035234

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. doi: https://doi.org/10.48550/arXiv.1706.03762

Vogel, E. K., McCollough, A. W., & Machizawa, M. G. (2005). Neural measures reveal individual differences in controlling access to working memory. *Nature*, *438*. doi: 10.1038/nature04171

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., & Steinhardt, J. (2022). Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*. doi: https://doi.org/10.48550/arXiv.2211.00593

Wang, Y.-S., Lee, H.-Y., & Chen, Y.-N. (2019). Tree transformer: Integrating tree structures into self-attention. *arXiv preprint arXiv:1909.06639*. doi: https://doi.org/10.48550/arXiv.1909.06639

Wei, Z., Wang, X. J., & Wang, D. H. (2012). From distributed resources to limited slots in multiple-item working memory: A spiking network model with normalization. *Journal of Neuroscience*, *32*. doi: 10.1523/JNEUROSCI.0735-12.2012

Whittington, J. C., Warren, J., & Behrens, T. E. (2021). Relating transformers to models and neural representations of the hippocampal formation. *arXiv*. doi: https://doi.org/10.48550/arXiv.2112.04035

Zhang, W., & Luck, S. J. (2008). Discrete fixed-resolution representations in visual working memory. *Nature*, *453*. doi: 10.1038/nature06860

A Appendix

A.1 Self-Attention in Transformers

Transformers are powerful language models which create contextualized representations of the input sequences. They learn to predict the next token one at a time using an "attention" mechanism that scans other tokens in the sequence for relevant information (Bahdanau, Cho, & Bengio, 2014). A common practice (that is used here) is to mask any future tokens and so predictions and representations must be made by the current token and any previously seen tokens. These models are able to learn and represent complex sequence modelling tasks.

For a given prediction, Transformer attention generates three separate vectors for each token in the sequence: a query, key, and value (q, k, v). The key vector is a set of tokens that the model has learned are most relevant to the token at hand. The query vector scans the tokens in the key vectors and calculates how much the current prediction should "attend" to each of those tokens. Then, the value vectors at those positions are multiplied by the corresponding weights, summed up, and added to the next representation: for token i at layer j, the contextual representation is $\sum_k q_i^j \cdot k_k^j * v_k^j$. Thus, the next token prediction includes earlier sequential information by combining the value vectors from previous tokens. In other words, Transformer attention can be viewed as a read/write mechanism: for a given token, the queries and keys dictate which tokens to read from, the values are the content that is read (proportional to the attention calculated by the keys and queries), and the summed content is written to a new representation at the given token. As we shall see below, the

comparison to role-addressable input and output gating operations is evident. The key vectors form addresses analogous to the PFC "stripes" (neural populations that support variable binding in memory). The learned key construction determines the address to store an item and is thus analogous to input gating. Conversely, the query vectors determine which addresses are accessed, and are thus analogous to output gating.

A.2 Textual Reference-Back-2 Task

The *textual reference-back task* requires making same/different judgments between incoming symbols assigned to a particular "register" in memory, with respect to those seen previously and linked to those same registers. Like the original tasks, the textual reference-back task is sequential, and requires independent updating and maintenance of two memory registers, each containing one of S arbitrary *symbols* at a time. At the beginning of each sequence, each register is initialized individually to one $s \in S$ (the pool of symbols is shared between registers, which was shown to more substantively tax gating mechanisms in (**?**).) Each sequence is composed of L tuples, each containing register address Reg_i, symbol Sym_i, same/different label Ans_i, and update instruction Ins_i. For a tuple $i \in L$, the **answer** Ans_i is a binary value that is either Same if symbol Sym_i is currently stored in the register with address Reg_i, or Different otherwise. The **update instruction** Ins_i also takes one of two values (`ignore` or `store`), evenly distributed. If the instruction is `ignore`, then the model still needs to make the same/different determination with respect to the stored reference, but the new symbol should not update the register content (i.e., the reference remains unperturbed). If it is `store`, then from that point on in the sequence, Sym_i is stored in the register with address Reg_i until otherwise updated. An example is shown in Fig. 1.

We implement each reference-back task example in our data as a single sequence, and measure models' ability to predict Same versus Different for each Ans_i. Each sequence has 10 same/different answers, and we generate 100,000 train, 1,000 validation, and 1,000 held-out test sequences.

The class balance of `same` to `different` answer labels in the train/test datasets is roughly 1:2, making a "maximum class" heuristic solution 0.66 accuracy, 0.33 precision, and 0.5 recall. We test several other heuristics, the strongest of which is predicting `same` if another tuple including `Store` and the target register and target symbol exists in the sequence, which scores 0.80 accuracy, 0.82 precision, and 0.85 recall.

A.3 Models and Training

We train small, attention-only Transformer models from scratch on our task. Our models contain two decoder-only layers, each with two heads, and no multilayer perceptrons or layer normalization, followed by a linear "unembed" layer to project the output of the last decoder into the space of the entire vocabulary at each timestep In practice, only 'same' and 'different' are ever predicted. Our

network uses absolute positional embeddings (Vaswani et al., 2017). The vocabulary contains all possible tokens, represented individually with embedding size E. Models are trained to predict the next token with the language modelling objective: if the model is predicting Ans_c, it will have access to all (Ins_i, Reg_i, Sym_i, Ans_i) tuples where $i < c$, as well as Ins_c, Reg_c, and Sym_c. However, the models only receive loss at positions where a same/different token must be predicted (this is similar to the reward function applied in frontostriatal gating networks; (O'Reilly & Frank, 2006; Soni & Frank, 2024)). Furthermore, each layer gets a causal attention mask– when constructing each token representation, it cannot look ahead at tokens further down the sequence.

The models are trained over 60 epochs of the 100k training data points, learning from 6 million examples in total. Models are evaluated on their accuracy (whether the correct Ans_i is predicted for each tuple i), measured in precision and recall, as well as the `same` versus `different` token logit difference.

A.4 Path Patching

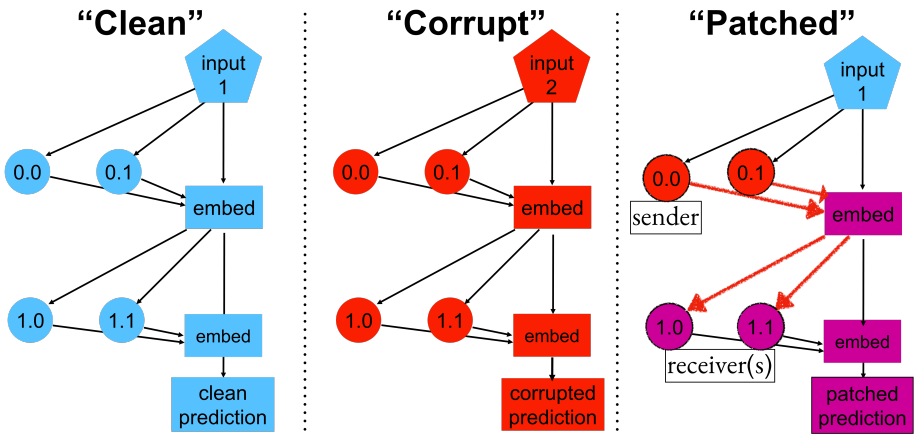

Figure 6: **Graphical Path Patching** Graphical diagram of the path-patching process. Attention heads are represented as circles (layer,head index), and contextual representations of each token (as well as the next token prediction) are represented as rectangles.

Path-patching involves making a incisive edit to the representations of a trained model and observing how the model's behavior is affected, allowing one to infer the computations implemented within individual attentional heads (see Fig. 6). Path-patching requires a minimal pair of examples: the "clean" example and the "corrupted" example, in which one token from the clean example is changed, as well as the correct label. Given representations from the model for both the clean and the corrupted examples (the blue and orange components in the figure), we can chose a specific component anywhere in the model (referred

to as the "sender"), and replace the clean representation with the corrupted one at that specific component. These embeddings will be received by the next layer ("receiver"), thereby "patching" the path. From there, the patched model will compute the new prediction.

In the figure, we send from layer 0, head 0 and 1 to layer 1, both heads 0 and 1. All clean representations that are not along this path are not modified and are unaffected by the patch. The model then recomputes all representations after the receiver (the "patched" representations), and arrives at a new prediction. If the model output matches the corrupt prediction rather than the clean one, that prediction is causally dependent on the path from sender to receiver. See (K. Wang et al., 2022) and (Goldowsky-Dill et al., 2023) for a more comprehensive review of path-patching methods.

We perform a small hyperparameter search and select a model that reaches 100% accuracy on the held-out test data for further analysis. We determine the circuit that the model uses through an array of path-patching experiments with a simple minimal pairs paradigm. Our "sender" within path-patching is always both attention heads at layer 0, and our "receiver" is always both attention heads at layer 1.

We first establish that a Transformer model is able to succeed on the reference-back-2 task.

At layer 0, the model learns to condense the task-critical information from each tuple into one embedding, at the position for Sym_i[2]. At this layer, the model pays 85.8% of total attention to the task-critical information to that tuple, and just 14.2% of attention to other tuples.

At layer 1, the attention heads learn to attend to the Sym_i key vector representing the tuple where information was last stored in the target register. The heads pay 70.2% of total attention to this tuple (the "stored" tuple), and only 29.8% of attention to all other tokens. This behavior is tied to the target register matching the register in the stored tuple, which is analogous to gating of the relevant role-addressable PFC stripe (O'Reilly & Frank, 2006; Soni & Frank, 2024). We focus our analysis on the Layer 1 representations which exhibit this learned gating policy, shown in Fig. 2.

A.5 Input vs. Ouput Gating Accuracies, Pretraining

[2]Redundantly, the model does the same at the position for Ans_i. Through additional experimentation, we determine that this is a quirk of Transformer learning, and does not impact our analysis.

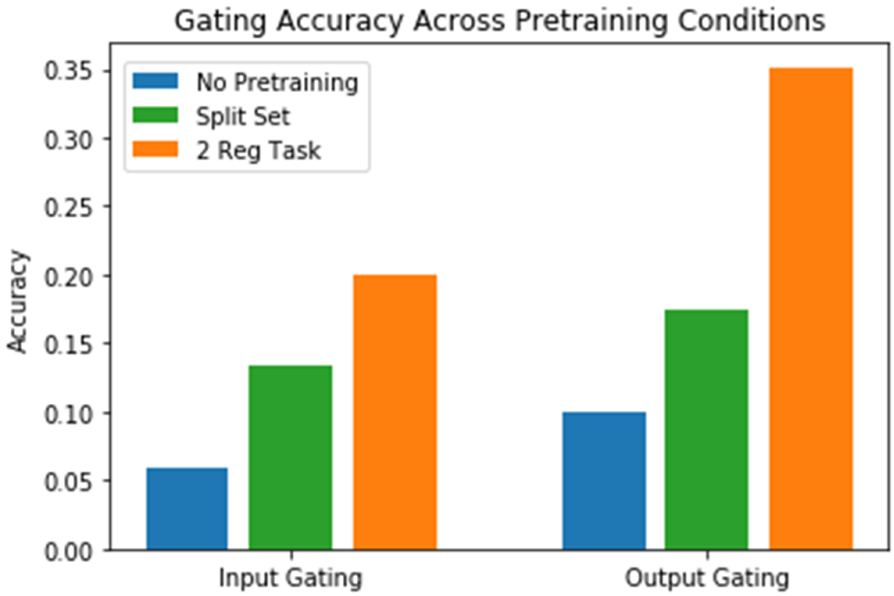

Figure 7: **Input and Output Gating Task Accuracy** Pretraining on the 2 register task leads to the highest accuracy on input and output gating subtasks - insinuating that these models learn the most mechansitic solutions. In general the input gating accuracy is lower, indidcating a differential role for each gating. There is human experimental evidence to suggest that output gating is harder to learn and is more crucial for better performance