Comparing Inherent Learning Capabilities of Recurrent Neural Networks

Jingfeng Zhang (jz44@illinois.edu)
Department of Psychology, 603 E Daniel St
Champaign, IL 61820 USA

Jon A. Willits (jwillits@illinois.edu)
Department of Psychology, 603 E Daniel St
Champaign, IL 61820 USA

Abstract

Recurrent neural networks (RNNs) remain central to cognitive modeling, despite the rise of transformers, because their recurrent dynamics are relatively cognitively plausible. But many questions remain about the strengths and weaknesses of RNNs as learning systems and cognitive models. We compare two architectures: Elman networks with hidden-layer recurrence and Jordan networks with output-layer recurrence. Using carefully constructed artificial languages, we show that Elman networks excel at learning long-range dependencies, especially when intervening elements provide no predictive cues. Jordan networks perform equally well when adjacent elements correlate with target outputs or when sequence-level error signals are available. However, Jordan networks falter in particular with item-level error feedback (as opposed to sequence-level error feedback), particularly for nonlinear sequential relations such as XOR. These results clarify how recurrence placement shapes sequence learning, suggest links to cognitive and neural processing, and highlight the representational advantages of hidden-layer recurrence over simple output chaining.

Keywords: computational modeling, neural networks, statistical learning, natural language processing

Introduction

Recurrent neural networks (RNNs) are a foundational cognitive model for representing the structure of sequences of events. RNNs have been used to model how humans learn linguistic structure and process sentences (Elman, 1990, 1991; MacDonald & Christiansen, 2002; McClelland, St John, & Taraban, 1989), learn semantic information (Elman, 1991; Huebner & Willits, 2018), segment and recognize speech (Christiansen, Allen, & Seidenberg, 1998; Elman, 1990; Magnuson & Luthra, 2024; McClelland & Elman, 1986; Xu, Sjuls, Kalashnikova, & Magnuson, 2024), learn action sequences (Cleeremans & McClelland, 1991), and represent serial order in working memory (Botvinick & Plaut, 2006). In recent years, RNNs have become less popular, both as cognitive models and in practical applications, because of the success of Transformer-style neural networks. This is true despite RNNs having a stronger claim to neural plausibility (owing to the widespread recurrent connectivity in the brain), and RNNs having greater cognitive plausibility (due to Transformers requiring extremely large 'working memories' to encode large context sequences).

Because of this mismatch between the neural plausibility advantage of RNNs over those same transformers on the one hand, and RNN performance limitations compared to Transformers on the other, it would be of great value to better understand the basic strengths and weaknesses of RNNs. The goal of this research is to use carefully constructed artificial languages to better understand some of the fundamental limitations of different architectures of RNNs. Specifically, we will show that two properties of RNNs interact to dramatically affect their performance: 1) The architecture of its recurrency (hidden layer vs. output layer recurrency), and 2) the window size over which learning operates (individual words versus whole sequences).

Recurrent Hidden vs. Recurrent Output Layer

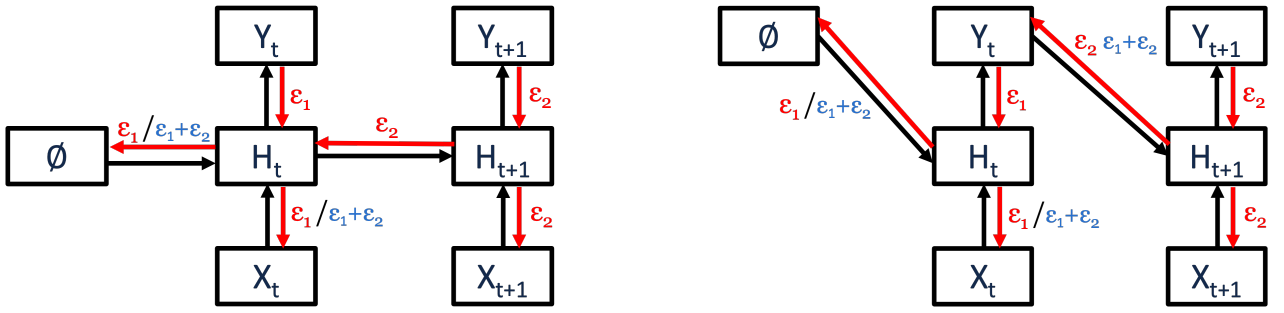
There are many varieties of recurrent neural networks. In this research, we are interested in two broad classes of RNNs: those with recurrent connections from their output layer (sometimes called "Jordan" networks, (Jordan, 1986)), and those with recurrent connections from their hidden layers, sometimes called "Elman" networks". The schematics of the two networks are shown in Figure 1.

The architectural differences between Elman and Jordan networks reflect distinct strategies for integrating past information and provide useful parallels to neurobiological and psychological processes. Elman networks, with their feedback from the hidden layer, model a dynamic internal workspace akin to working memory systems in the brain, where representations of past inputs are continually updated and transformed. This aligns with tasks requiring complex, non-linear dependencies, such as language comprehension, where hierarchical structures must be extracted and maintained. In contrast, Jordan networks, which rely on feedback from the output layer, mirror situations where behavioral outputs or immediate environmental feedback guide future processing, such as in motor control or auto-regressive prediction tasks. This architecture parallels psychological theories of reinforcement-driven behavior, where recent actions directly influence subsequent decisions. These distinctions suggest that the brain may employ different neural circuits or mechanisms (analogous to Elman or Jordan architectures) depending on whether the task requires flexible abstraction or simple reliance on output-driven feedback loops.

Item vs. Sequence Level Learning

We are also interested in whether differences in the learning regime affect model performance as much as architectural differences. Also see figure 1. Under **item-level con-**

In G. Boleda, I. Dautriche, M. Maldonado, C. Saldana and T. Torrent (Eds.), *Proceedings of the 48th Annual Conference of the Cognitive Science Society*



(a) Elman network **(b)** Jordan network

Figure 1: Different recurrent architectures of Elman networks (hidden-layer activations at time $t-1$ are fed back as input to the hidden layer at time t), and Jordan networks (output-layer activations at time $t-1$ are fed back to the hidden layer at time t). Red arrows indicate error backpropagation pathways, with ε denoting error signals; red arrows correspond to item-level learning, and blue arrows correspond to sequence-level learning.

tinuous learning, weights are updated based on the immediate next-item prediction error, which strongly favors adjacent cues. Under **sequence-level learning** (backpropagation through time), error signals from later positions are propagated backward to earlier time steps, allowing later outcomes to shape earlier representations and making long-distance dependencies more learnable. These differences parallel distinctions in psycholinguistics between local, incremental prediction and more global, hindsight-driven adjustment, as well as between online sensitivity to surface order and learning dependencies that span intervening material. We therefore expected Jordan and Elman networks to exhibit systematically different learning behavior under each regime.

Little research has compared the performance of Elman and Jordan RNNs with the goal of understanding their relative strengths and weaknesses. Pham and Karaboga (1999) trained Elman and Jordan networks to predict sequences generated by simple linear and nonlinear equations, finding that Elman networks had greater accuracy. Wu, An, Guan, Huang, and Zhou (2019) built prediction models for human brucellosis cases using Elman and Jordan recurrent neural networks, and compared their performance to a traditional seasonal ARIMA model. One limitation of this previous research into RNNs is that they have typically been applied to datasets that were not designed to test specific *a priori* theoretical predictions about how the networks' performance should differ. In the current work, we use a carefully controlled artificial language manipulated specifically to test hypotheses about the differences between the two architectures.

Current Study Hypotheses

We tested three hypotheses about the differences between Elman networks (Elman, 1990) and Jordan networks (Jordan, 1986). The first hypothesis was that Jordan networks must learn non-adjacent dependencies via chaining across a sequence, whereas Elman networks can learn non-adjacent dependencies through chaining or via direct representation of the non-adjacent relations. The second hypothesis was that Elman networks, but not Jordan networks, would be able to learn sequences that involved non-linearly separable relationships. For example, if predicting the third element in a three-element sequence requires learning XOR relations between the first two elements in a sequence, hidden layer recurrency is necessary to learn the sequence. We also hypothesize that sequence-level training would benefit both models because it provides explicit, temporally structured error signals over entire sequences, in contrast to item-level continuous learning, where learning signals are local and incremental.

These hypotheses, if true, would correspond with and help explain the observation that RNNs with recurrent level connectivity are a more powerful architecture, capable of learning in situations where Jordan networks cannot. They might also help explain or predict situations of the reverse, where networks with output level recurrent connectivity would have an advantage, of which there are a few. The first is that in cases where linearly separable chaining is sufficient, Jordan networks will learn more quickly (just as a single-layer network will learn more quickly than a multilayer network). Second, there are conceivably situations where the extra power of Elman networks could lead to overfitting and poorer learning overall, not just slower learning. A third consideration is that, just because an SRN is a more powerful architecture, this does not mean it is a better model of human behavior. For example, there are a number of cases (such as (Cleeremans & McClelland, 1991)) where Jordan networks (or Jordan-Elman hybrid models) seem to fit better to human data.

The AxB Grammar

To test our hypotheses, we employed an artificial language that allowed us to test the effects of chaining and nonlinearly separable relationships on long-distance dependency learning. The specific grammar we used is a variant of the AxB grammar (Endress & Johnson, 2021; Gómez & Maye, 2005; Willits, 2013). The AxB grammar defines a language with sequences containing four elements: A, x, B and ".". In the AxB grammar, A, x, and B are categories, and there are sets of items that make up those categories.

The AxB grammar can instantiate non-adjacent dependencies by making members of A and B belong to subcategories (A1, A2, B1, and B2). We can then include a rule: if the sequence starts with a member of A1, it must have a member of B1, and if it starts with a member of A2, it must have a mem-

Table 1: Legal sequences in the AxB grammar. Each element indicates a subcategory containing n-members (n=4 in the current experiments). In Experiment 1, bold sequences were three times as frequent as unbold sequences. In the other experiments all sequence types were equally frequent.

Exp. 1	Exp. 2	Exp 3
A1 x1 B1 .	A1 x1 B1 .	A1 x1 B1 .
A1 x2 B1 .	A1 x2 B1 .	A1 x2 B2 .
A2 x1 B2 .	A2 x1 B2 .	A2 x1 B2 .
A2 x2 B2 .	A2 x2 B2 .	A2 x2 B1 .

ber of B2. The nonadjacent A-B dependencies (spanning the x-items) can be thought of as a simple model of many such nonadjacent relations in language. This includes phonemic relationships within words, where word onsets predict offsets that span intermediate vowels; morphosyntactic relationships, like between plural agreement between sentence subjects and verbs; and semantic relationships, such as *dogs/cats have paws/fur* vs. *crows/ducks have wings/feathers*.

In the AxB grammar, the intermediate x-items traditionally are made independent of the A-B relationships, so as to provide a situation where adjacent dependencies provide no cues to what the correct B item will be, so that the A-B relationships must be learned. But in our studies, we will manipulate this to test our hypothesis about whether Jordan networks require chaining via adjacent items.

Experiment 1: Correlated Intervening Items

In Experiment 1, we investigated the ability of different neural networks to learn to predict sequences containing nonadjacent dependencies where the distributional structure of the intervening items was partially correlated with the nonadjacent relationship. As we will show, this provides an interesting challenge for sequence-learning models and a good starting comparison between Elman and Jordan networks.

Artificial Language Corpus

We used these items to create a corpus of 216 sequences using the following rules. There were 27 sequences consisting of all possible combinations of A1-x1-B1 sequences, 27 A1-x2-B2 sequences, 27 A2-x1-B2 sequences and 27 A2-x2-B2 sequences. The frequency of the A1-x1-B1 and A2-x2-B2 sequences was then multiplied by three, resulting in (27*3)+(27*1)+(27*1)+(27*3)=216 sequences. The set of legal sequences in Experiment 1 is shown in Table 1.

The resulting corpus had the following properties. The A-element in a sequence came from a set that perfectly predicted the set of B-elements that were legal continuations of that sequence (*i.e.*, sequences with an A1 must have a B1 and sequences with an A2 must have a B2). The x-element in any sequence came from a set that was imperfectly correlated with the set of B-elements that were legal continuations of that sequence. In other words, because A1-x1-B1 sequences were three times more frequent than A1-x2-B1 sequences, the existence of an x1 element meant there was a 0.75 probability of a B1-element, and a 0.25 probability of a B2-element (and vice versa for the existence of an x2 element).

This structure poses an interesting challenge for different classes of models. The models should all be good function at learning the general sequence (A-x-B-.). The critical test of all the models will be their prediction of which B-elements are legal continuations of the sequence. Successful models will predict elements from the correct B subcategories B1 and B2; unsuccessful models will not distinguish between these items.

Model Architectures

In Experiment 1 we compared the performance of three neural network architectures. All three used the same set of 20 input and output units, one unit for each token in the vocabulary, plus one "PAD" unit used to provide an empty context window for the beginning of the corpus. All three models also had a hidden layer consisting of 16 units. The underlying category structure of the grammar involves a small number of binary distinctions (A1 vs. A2, B1 vs. B2, x1 vs. x2), requiring the hidden layer to encode at most a few bits of information to solve the task. Sixteen hidden units provides substantially more representational capacity than this requires. We intentionally chose a hidden layer size that exceeds the minimum needed, so that any failures in learning can be attributed to architectural or algorithmic limitations rather than to insufficient capacity.

For control purposes, we tested a non-recurrent multilayer network. Because of the inter-correlated relationship of the x subcategories with the A-B relationships, this network should be able to achieve 75% accuracy in predicting the correct subcategory of the B elements (B1 vs. B2) by always predicting a B1 after an x1, and always predicting a B2 after an x2. But it should not be able to achieve perfect prediction because it has no way to know, when an x was the input, whether the previous input was an A1 or an A2.

The Elman networks can learn hidden representations that are useful for predicting downstream elements, and can learn to retain information about previous states over relatively long distances (Elman, 1991; Huebner & Willits, 2021). Therefore, when the A-element is the input, they should learn to activate hidden representations that encode the A's subcategory, and use this to predict the more probable x element's subcategory. They should also learn to retain activation of the identity of the A-element via the hidden-hidden recurrent connections, and use this to perfectly predict the correct B subcategory. However, it is possible the Elman network will first learn representations for A items that are useful for predicting the x-items, and because of the imperfect x-AB relationship, get "stuck" on those representations and have trouble discovering the more accurate long-distance relationships. Such a failure would be similar to blocking in classical learning theory (Kamin, 1969) and proactive interference in classical memory research (Keppel & Underwood, 1962).

Jordan networks work differently. When a Jordan network sees an A element, they should learn to predict the correct

x subcategory. Then, when the x element is then the input, its own predictions at the previous time-step are the recurrent input. Because the x subcategories are correlated (albeit imperfectly) with the A-B relationships, the Jordan network should still succeed at predicting the correct B-subcategory. When the current input is an x, and the previous input was an A1, the recurrent input will tend to have higher activations for x1's than x2's. When the previous input was an A2, the x2's will tend to have higher recurrent activations than the x1's. This difference in recurrent activation, if properly weighted, can then be used as the basis for making the correct prediction about B1 vs. B2. In this way, the inter-correlations of the x-elements with the A-B relationships seem to play an important role in the Jordan's network's ability to make the correct predictions. This contrasts with the Elman network, which shouldn't rely on this inter-correlations.

Finally, we varied the learning regime in Experiment 1, but we did not expect it to substantially alter the outcomes. The dependencies tested in this experiment can, in principle, be learned through relatively local chaining mechanisms that are available to both recurrent architectures.

Method

Training Procedure. We created 30 randomly initialized versions of each model architecture. Each model was trained for 200 training epochs. In each epoch, each model saw the entire training corpus, one token at a time. For each token, the corresponding input unit was activated, and activation was fed forward through the network's weighted connections resulting in an output activation. For models with recurrent layers, we used the appropriate previous state(s) as an additional input into the hidden layer. We compared the models' predicted outputs to the correct outputs, and computed error using the cross-entropy loss function using backpropogation. All models were trained using an SGD optimizer function with a learning rate of 0.01.

Results and Discussion

The performance of all three models can be seen in Figure 2, showing the mean accuracy predicting items from the correct B subcategory, averaging across the 30 runs of each model type, over the course of 200 epochs of training. On any given B-item trial (i.e., a trial where an x-item was an input, and a B-item was the correct next item to predict), we looked at the model's most highly activated output unit. The trial was counted as correct if the unit was from the correct B subcategory. For example, if the trial was one where x1_1 was the input, and A1_1 had been the input on the trial before, then if any member of B1 (B1_1, B1_2, or B1_3) was the most activated unit, the trial was counted correct.

As hypothesized, the nonrecurrent model achieved an accuracy of 75% predicting items from the correct B subcategory when an x was the input, reflecting the inter-correlation of the x subcategories and the A-B subcategory relationships. In contrast, both the recurrent hidden (Elman) network and the recurrent output (Jordan) network achieved 100% accuracy. The Jordan network was able to use its recurrent activation of its differential predictions of x subcategories to predict items from the correct B subcategories. The Elman network was able to activation a hidden representation of the A input and use that activation as input when x was the input (via the hidden-hidden connection) to predict items from the correct B subcategory. The Elman network's performance did not differ between item-level learning and sequence-level learning, achieving rapid and reliable convergence under both regimes. The Jordan network also reached perfect accuracy under both learning regimes, but learning was slower under item-level learning than under sequence-level learning, achieving perfect performance in about 10 epochs, compared to about 80.

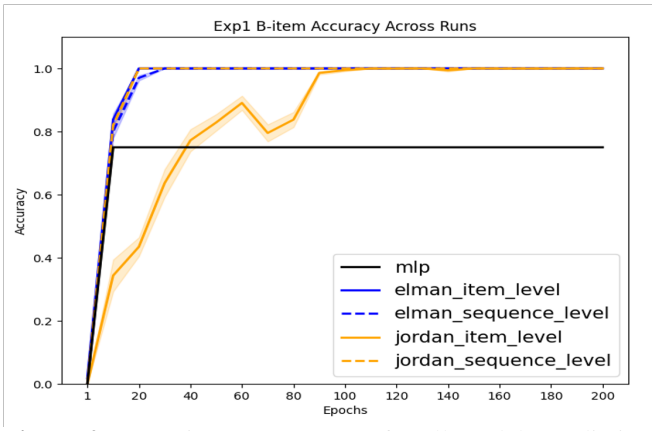


Figure 2: Experiment 1 accuracy for all models predicting items from the correct B subcategory over training when intervening x-items were correlated.

Experiment 1 provides a useful baseline where both Elman and Jordan networks succeed, albeit for different reasons. Their differential reasons for success lead to predictions about conditions under which their performance should vary more substantially, as we will test in Experiments 2 and 3.

Experiment 2

In Experiment 2, we tested the same neural network architectures on their ability to learn A-B nonadjacent relationships when the intervening x-items are orthogonal and completely non-predictive of which B-item should occur. RNNs with hidden layer recurrency can learn this kind of relationship (Willits, 2013; Zhang & Willits, 2024), but it is untested whether RNNs with output layer recurrency can do so.

There is reason to believe that under these conditions, Jordan networks may fail. As described in Experiment 1, Jordan's networks' ability to succeed at predicting items from the correct B-subcategories should depend on whether they are making differential output predictions after A1 inputs and A2 inputs. If they are (as they were in Experiment 1 where A1 imperfectly predicted x1 and A2 imperfectly predicted x2), they can use this differential to perfectly predict which B-subcategory's items will occur. But in the case where x subcategories are orthogonal to the A and B subcategories, there are no systematic differences about what follows A1 and A2 items. As a consequence, Jordan networks' outputs should

be identical following A1 items and A2 items, and therefore their recurrent activations when x items are the inputs should be indistinguishable, leading to an inability to correctly predict which B subcategories items can occur.

With respect to learning regime, we did not have a strong prediction for Experiment 2. Although sequence-level learning can, in principle, support the acquisition of longer-distance dependencies, the critical limitation for Jordan networks in this condition arises from the absence of informative intermediate cues rather than from constraints on temporal credit assignment. As a result, we did not expect differences between different learning regime to change the success or failure of the architectures in this experiment.

Method

Training Corpus. The corpus in Experiment 2 was identical to Experiment 1, except all sequence types (A1-x1-B1, A1-x2-B1, A2-x1-B2, A2-x2-B2) were equally frequent. This removed the correlation between the x subcategories and the A and B subcategories. The A1-B1 and A2-B2 relationship remained perfectly predictive in Experiment 2.

Architectures and Training Procedure. The models and training procedure were identical to those in Experiment 1, with a learning rate of 0.005 for all models.

Results and Discussion

The performance of all models can be seen in Figure 3, showing the mean accuracy predicting items from the correct B subcategory, averaging across 30 runs of each model type, over the course of 500 epochs of training. As expected, non-recurrent multilayer networks showed incapability of predicting items from the correct B subcategory, reflecting limitations of a non-recurrent architecture in handling nonadjacent dependencies in sequence prediction. Similarly as predicted, the Elman network succeeded in learning to predict items from the correct B subcategory when an x-item was the input. Although learning under item-level training was more difficult in this condition, the Elman network nevertheless succeeded in most cases, with 28 out of 30 randomly initialized models achieving perfect performance by the end of training.

Surprisingly, and contrary to our predictions, the Jordan network succeeded in learning to predict items from the correct B subcategories, but only under sequence-level learning. When trained with item-level continuous learning, Jordan networks failed to acquire the nonadjacent A–B relationships. This pattern is notable because there is no systematic relationship between A subcategories and the intervening x-items. To account for the success of the Jordan network under sequence-level learning, we consider how learning unfolds when error signals are propagated through time.

Under backpropagation through time, prediction errors from later B-item predictions are propagated backward to earlier time steps, including those in which A-items are processed. As a result, the error signal from predicting B-items can influence how the network adjusts its predictions of x-items following A1 versus A2 inputs. Although the network

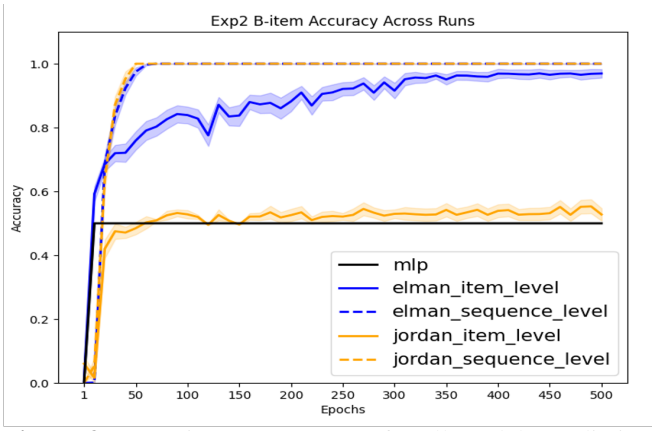


Figure 3: Experiment 2 accuracy for all models predicting items from the correct B subcategory when intervening x-items were orthogonal.

learns that x-items are equally likely after A-items, small idiosyncratic differences in output activations following A1 and A2 inputs, which initially arising from random weight initialization, can be preserved and gradually amplified through learning. These differences, while extremely small, become meaningful when output activations are fed back as recurrent input at the next time step, allowing the network to distinguish which B subcategory is more likely when x is the input. In this way, error signals from predicting both x-items and B-items jointly shape the recurrent context, enabling the Jordan network to succeed under sequence-level learning despite the lack of informative intervening structure.

These results raise interesting questions about Jordan networks and other models in which recurrency is based on output feedback rather than internal state feedback. Surprisingly, such models can succeed in cases where, given their objective function, they should not. This success appears to arise from arbitrary, unextinguished differences produced by suboptimal learning at the previous time step and by training strategies. This finding is interesting for two reasons: it raises questions about the stability of Jordan networks' ability to maintain these differences, and it suggests another case in which noise in the learning process can be beneficial, leading to more generalizable performance.

Experiment 3

In Experiment 3, we tested a more difficult case in which correct prediction of the B subcategory requires computing a sequential XOR relationship over both the preceding A-item and the intervening x-item. In this condition, neither A nor x alone is sufficient to determine the correct B subcategory; instead, the model must jointly represent both elements, and critically, their relationship is non-linearly separable. As shown in table 1, B1 occurs when the sequence contains either A1 followed by x1 or A2 followed by x2, whereas B2 occurs when the sequence contains either A1 followed by x2 or A2 followed by x1. As a result, the individual pairwise relationships among A, x, and B are uninformative: the A subcategory does not by itself predict either the x or B subcat-

Figure 4: Experiment 3 accuracy for all models predicting items from the correct B subcategory when the Ax-B relationship was not linearly separable

egories, nor do the x and B subcategories predict each other.

Following the logic of Experiment 2, one might expect Jordan networks to struggle in this XOR condition. Although Jordan networks can sometimes exploit residual differences in their recurrent input, the XOR structure requires these residual signals not only to be preserved across the intervening item but also to be nonlinearly combined with the current input. Without a mechanism for maintaining and integrating an internal representation of A across the x-item, this presents a particularly challenging learning scenario for Jordan networks even with sequence-level training. Elman networks, by contrast, can maintain such representations in their recurrent hidden state and are therefore expected to succeed.

Method

Training Corpus. The training corpus is shown in Table 1, and described above.

Model Architectures and Training Procedure. The model architectures and training procedure were identical to those used in Experiments 2. For item-level learning, both the Elman and Jordan networks employed a modified initialization for the input-to-hidden connections, in which weights of linear modules corresponding to the *hx* pathway were initialized from a uniform distribution over $[-0.8, 0.8]$.

Results and Discussion

The performance of all models can be seen in Figure 4, averaged across 30 runs of each model type over 800 training epochs. The nonrecurrent model achieved 0.5 mean accuracy.

Elman network successfully learned the sequence under both item-level continuous learning and sequence-level learning. As in previous experiments, learning under item-level training was slower, but the Elman network nevertheless reached perfect performance in both regimes. In contrast, the Jordan network's performance depended strongly on the learning regime. It failed under item-level learning but succeeded under sequence-level learning, albeit more slowly than the Elman network: the Elman network reached perfect accuracy after approximately 240 epochs, whereas the Jordan network required nearly 480 epochs. This contrasts with earlier experiments in which the two architectures exhibited comparable learning speeds under sequence-level training.

General Discussion

This study investigated the learning capabilities of Elman and Jordan recurrent neural networks (RNNs) in acquiring non-adjacent dependencies using a controlled artificial language. The findings demonstrated that Elman networks, with hidden layer recurrency, were effective at learning long-distance dependencies, even when intervening elements provided no predictive cues. In contrast, Jordan networks, rely on recurrent connections from the output layer, and struggled relative to Elman networks. In Experiment 1, when intervening elements carried some predictive value, Jordan networks performed as well as Elman networks when sequence level learning was used, and learned but slightly more slowly when item level learning was used. In Experiment 2, when intervening elements were independent and orthogonal, they still performed well when sequence level learning was used (because representation of the sequence could be used to relate prediction error across the distance), but dropped to chance performance when Item level learning was used. In Experiment 3, when dependencies required non-linear representations such as XOR relationships, Jordan networks were again at chance with Item level learning, and even learned more slowly than Elman networks when sequence level learning was used.

These findings have important implications for understanding the cognitive and neural plausibility of different RNN architectures, particularly in relation to human language learning. The study reaffirms the advantage of Elman networks in modeling tasks that require hierarchical abstraction, such as syntactic parsing and semantic integration, aligning with psychological theories that emphasize working memory-like processes in linguistic cognition. On the other hand, the success of Jordan networks in tasks involving adjacent predictive cues suggests a parallel to reinforcement-driven behavior and associative learning. This dichotomy highlights the potential division of labor in the brain, where some circuits may rely on immediate feedback from prior outputs while others maintain dynamic internal states to support abstract generalization.

More broadly, these results contribute to the ongoing debate about the relevance of RNNs as cognitive models in light of the rise of transformer architectures. Although transformers have outperformed RNNs in many practical applications, this study underscores the continued importance of recurrent architectures to capture sequential dependencies in a neurally and cognitively plausible manner. Understanding the strengths and limitations of RNNs provides insight into how human cognition balances memory constraints with representational flexibility. Future research could explore hybrid models that integrate the advantages of Elman and Jordan-style recurrency, potentially leading to architectures that more accurately reflect the diverse mechanisms underlying human learning and prediction.

References

Botvinick, M. M., & Plaut, D. C. (2006). Short-term memory for serial order: a recurrent neural network model. *Psychological Review*, *113*(2), 201.

Christiansen, M. H., Allen, J., & Seidenberg, M. S. (1998). Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes*, *13*(2-3), 221-268.

Cleeremans, A., & McClelland, J. L. (1991). Learning the structure of event sequences. *Journal of Experimental Psychology: General*, *120*(3), 235.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179-211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, *7*, 195-225.

Endress, A. D., & Johnson, S. P. (2021). When forgetting fosters learning: A neural network model for statistical learning. *Cognition*, *213*, 104621.

Gómez, R., & Maye, J. (2005). The developmental trajectory of nonadjacent dependency learning. *Infancy*, *7*(2), 183-206.

Huebner, P. A., & Willits, J. A. (2018). Structured semantic knowledge can emerge automatically from predicting word sequences in child-directed speech. *Frontiers in Psychology*, *9*, 133.

Huebner, P. A., & Willits, J. A. (2021, November). Scaffolded input promotes atomic organization in the recurrent neural network language model. In A. Bisazza & O. Abend (Eds.), *Proceedings of the 25th conference on computational natural language learning* (pp. 408–422). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.conll-1.32/ doi: 10.18653/v1/2021.conll-1.32

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 8).

Kamin, L. J. (1969). *Selective association and conditioning*.

Keppel, G., & Underwood, B. J. (1962). Proactive inhibition in short-term retention of single items. *Journal of Verbal Learning and Verbal Behavior*, *1*(3), 153-161.

MacDonald, M. C., & Christiansen, M. H. (2002). Reassessing working memory: comment on just and carpenter (1992) and waters and caplan (1996).

Magnuson, J. S., & Luthra, S. (2024). Simple recurrent networks are interactive. *Psychonomic Bulletin & Review*, 1-9.

McClelland, J. L., & Elman, J. L. (1986). The trace model of speech perception. *Cognitive Psychology*, *18*(1), 1-86.

McClelland, J. L., St John, M., & Taraban, R. (1989). Sentence comprehension: A parallel distributed processing approach. *Language and Cognitive Processes*, *4*(3-4).

Pham, D. T., & Karaboga, D. (1999). Training elman and jordan networks for system identification using genetic algorithms. *Artificial Intelligence in Engineering*, *13*(2), 107–117.

Willits, J. (2013). Learning nonadjacent dependencies in thought, language, and action: Not so hard after all. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 35).

Wu, W., An, S.-Y., Guan, P., Huang, D.-S., & Zhou, B.-S. (2019). Time series analysis of human brucellosis in mainland china by using elman and jordan recurrent neural networks. *BMC infectious diseases*, *19*, 1–11.

Xu, Q., Sjuls, G. S., Kalashnikova, M., & Magnuson, J. (2024). Modeling infant cortical tracking of statistical learning in simple recurrent networks. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 46).

Zhang, J., & Willits, J. (2024). Distributional language models and the representation of multiple kinds of semantic relations. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 46).