VSA-based Positional Encoding Can Replace Recurrent Networks in Emergent Symbol Binding

Francesco S. Carzaniga[1,2,*], Michael Hersche[1,3], Kaspar Schindler[2] and Abbas Rahimi[1]

[1]IBM Research-Zurich, Rüschlikon, Switzerland

[2]Department of Neurology, Inselspital, Sleep-Wake-Epilepsy-Center, Bern University Hospital, Bern University, Bern, Switzerland

[3]ETH Zürich, Zürich, Switzerland

Abstract

Variable binding is an open problem in both neuroscience and machine learning relating to how neural circuits combine multiple features into a single entity. Emergent Symbols through Binding in External Memory is a recent development tackling variable binding with a compelling solution. An emergent symbolic binding network (ESBN) is able to infer abstract rules through indirection using a dual-stack setup—whereby one stack contains variables and the other contains the associated keys—by autonomously learning a relationship between the two. New keys are generated from previous ones by maintaining a strict time-ordering through the usage of recurrent networks, in particular LSTMs. It is then a natural question whether such an expensive requirement could be replaced by a more economical alternative. In this work, we explore the viability of replacing LSTMs with simpler multi-layer perceptrons (MLPs) by exploiting the properties of high-dimensional spaces through a bundling-based positional encoding. We show how a combination of vector symbolic architectures and appropriate activation functions can achieve and surpass the results reported in the ESBN work, highlighting the role that imbuing the latent space with an explicit structure can play for these unconventional symbolic models.

Keywords

Emergent symbolic binding network, vector symbolic architectures, symbolic reasoning, siren, sparse distributed memory, variable binding, recurrent neural network

1. Introduction

In the fields of neuroscience and philosophy, the *binding problem* [1] refers to the ability of the human brain to form a cohesive experience out of the myriad of inputs it receives from both the external environment as well as the continuous feedback signals which are generated internally. Visual sensation has been perhaps most studied in this regard [2, 3]. Particularly relevant nowadays is the ability of the human brain to process and decompose sentences (and more in general language [4]) into their constituent components [5].

Neural binding can be subdivided further with each sub-problem being of great interest in its own right. However, here we will focus exclusively on the *variable-binding* aspect. The inverse of the binding problem is the unbinding problem, or the *best match problem*, which deals with how to effectively separate the neural correlates into their foundational components.

NeSy 2023: 17th International Workshop on Neural-Symbolic Learning and Reasoning

*Corresponding author.

frc@zurich.ibm.com (F. S. Carzaniga)

Both binding and unbinding play a critical role in the human brain's ability to produce abstract concepts, elaborate them, and more in general to reason symbolically. For a neural network to exhibit the latter behaviours, it is therefore necessary for it to solve the former problems as well [6, 7].

Neuro-symbolic artificial intelligence (NSAI) [8, 9, 10, 11] with its many incarnations combines a symbolic and structured representation with the end-to-end learning capabilities of neural networks. This enables competitive, and even superior, performance with the current state of the art in visual abstract reasoning tasks [12]. By leveraging the capabilities of NSAI, emergent symbolic binding networks (ESBNs) [13] enable a form of variable-binding and indirection. Its central component is a hetero-associative memory, or *external* memory, which self-optimises the relationship between keys and values through pure back-propagation. This external memory bears a resemblance with Kanerva's sparse distributed memory (SDM) [14] as a flexible model that can be both hetero-associative, where the key and value are different, and auto-associative, where the key and value are the same. The keys in SDM are carefully selected random vectors to maintain minimal destructive interference (thanks to the properties of high-dimensional spaces we detail in Section 2.1). Recently, SDM has been identified as a close analogue of Transformer attention [15].

It has been shown in many NSAI models that imbuing the latent space with implicit structure (e.g. creating a semantic hierarchy among concepts) significantly improves performance in reasoning tasks. In this work, we establish that this effect is also relevant in ESBNs. We exploit the properties of vector symbolic architectures (VSAs) [16, 17, 18, 19] in order to modify interactions within the key space, without altering the value space. This is made possible by VSA's great flexibility and allows us to keep the original indirection mechanism intact and strengthen our conclusions.

We present a simpler alternative to cumbersome recurring networks in ESBNs. By combining a multi-layer perceptron (MLP) with appropriate activation functions and positional binding based on VSA, we can successfully replace an LSTM and still solve visual abstract reasoning tasks. Moreover, we characterise the regularising effect of LSTMs in ESBNs, and show that while they effectively constrains over-parametrisation, this also leads to under-expressiveness in some situations. Taken together, our findings show that MLPs are capable of the same time-ordering sensitivity as LSTMs and clarify that the resulting increase in expressiveness is beneficial.

2. Background

2.1. Vector symbolic architectures

To address the variable-binding problem we have opted here to use an approach based on VSAs [16, 17, 18, 19], which explicitly define functions to bind keys to values. This is in contrast to ESBNs [13] which learn how to do indirection on their own. VSAs are computational paradigms which exploit the properties of high dimensional spaces to represent and manipulate symbols. A detailed overview can be found in [20] and [21].

There are a variety of possible representations residing under the term VSA ([22]) and, while it is not necessary here to present all of them, it is nonetheless of interest to remark that not one solution can be found to fit all problems. It is indeed often the case that some tasks perform

better with one or another. The element which binds all these representations together is their ability to imbue a high dimensional space with some structure. We focus precisely on this common aspect to better highlight the strengths of VSA.

In contrast with the well-known *curse of dimensionality*, high dimensional spaces also exhibit extremely beneficial properties, not the least of which is the concentration of measure [23]. This plays a role in the generation of the positional code-book, i.e. the set of vectors used to distinguish one time step from another. For example, it is possible to randomly draw an arbitrary number of vectors while still guaranteeing that all of the vectors remain quasi-orthogonal and easily distinguishable from one another. This allows for greater flexibility in the number of time steps that can be emulated, with the understanding that LSTMs and other recurrent neural networks are much more limited in how long they can be run continuously before encountering gradient issues.

In VSAs, two operations, bundling and binding, form the bedrock of any model. Binding, as the name suggests, performs key-value binding, while bundling creates sets of symbols. Within the space, bundling preserves the similarity of the inputs (i.e. the cosine similarity of the output with each of the inputs is high), while binding does not. We have given an abstract definition which outlines the necessary properties these two operators must possess; however, the specific instantiation is left to the implementer.

A definite realisation of binding and bundling, together with the accompanying space, identify a particular member of the VSA family (for more information see [22]). As binding is taken care of by the ESBN, we need only focus on bundling and we do so by choosing the simplest and most general form: component-wise addition. It must be noted that this kind of bundling is also performed (with a specific code-book) by Transformers [24] under the name of positional encoding. A detailed description of our VSA implementation is presented in Section 3.1.

Now that we have explored the bundling aspect of the architecture we can focus on binding, which follows the ESBN model.

2.2. ESBN

The ESBN architecture has been introduced in [13] as a possible solution to the variable binding problem. ESBNs have been developed to solve reasoning tasks in the form of extrapolating relationships from sets of images. As we believe this is a relevant setting for assessing human-like capabilities, we do the same. Therefore, the input to the network are black and white images, and the output a variable length binary vector encoding such relationship. For more information refer to Section 2.2.2. Here, we provide a brief overview of the fundamentals details for the function of the model, for a more in-depth treatment of the model please refer to the paper above.

2.2.1. Architecture

Symbolic binding networks are composed of two mostly independent components, which we explicitly term the value pathway and the key pathway (cfr. Figure 1, top panel) to make further analysis easier. We present now an overview of both pathways, exposing their machinery and describing their underlying components.

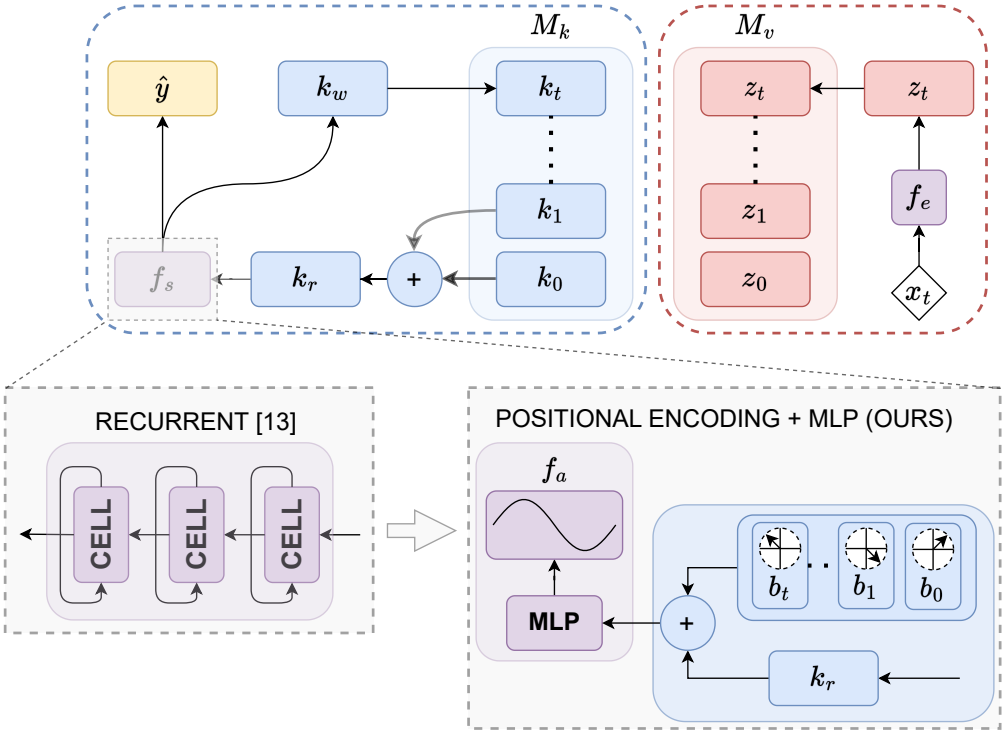


Figure 1: Schematic of the original ESBN architecture with a focus on our modifications. We replace the LSTM with a triple consisting of our positional encoding b, an MLP, and an activation function f_a. Highlighted in blue is the key pathway, in red the value pathway. Our work reduces the number of parameters by $\sim 2.4\times$.

At each step, an image sample x_t passes through the value pathway (cfr. Listing 1), where an encoder f_e generates a feature vector z_t and appends it to the value stack M_v. At the same time, z_t is compared to each element in M_v to form a vector of weights w_t. This mechanism closely resembles attention in that the weight of a given memory item is proportional to its similarity with z_t. In the key pathway (cfr. Listing 2), the similarity vector w_t is used to combine the keys in M_k in a weighted sum (or superposition) to generate k_r. The newly generated k_r is then fed into an LSTM which outputs a prediction and a new key k_w. Finally, k_w is appended to M_k, not k_r.

Listing 1: One step in the value pathway.

$$z_t = f_e(x_t)$$
$$w_t = \sigma(M_v \cdot z_t)$$
$$\mathrm{p}(z_t, M_v)$$

Listing 2: One step in the key pathway.

$$y_t, g_t, k_{w_t} = f_s(k_{r_{t-1}})$$
$$c_t = \mathrm{S}(M_k \cdot w_t)$$
$$k_{r_t} = g_t \sum_i (w_t)_i (M_k, c_t)_i$$
$$\mathrm{p}(k_{w_t}, M_k)$$

Listings 1 and 2: σ is the softmax function. S is the sigmoid function. (a, b) refers to concatenation. p pushes an element onto the stack.

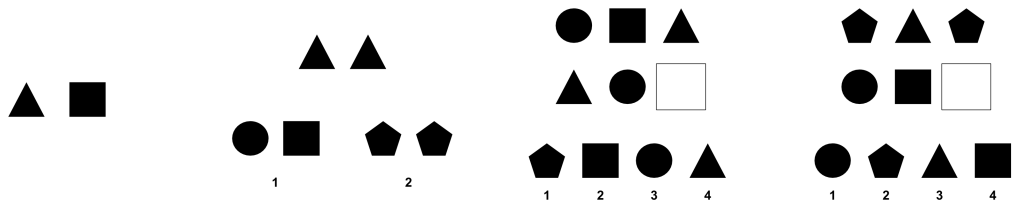
While the two pathways never interact explicitly, their computational graphs are connected through w_t such that back-propagation is possible. Each image in the task, including both question and answer panels, is fed through the network. The final output $\hat{y}$, be it a binary or one-hot-encoded vector, is used to determine the answer in a task-dependent manner.

A key contributor to the performance of ESBNs in [13] is the introduction of a temporal context norm (TCN). This scheme acts as a regulariser by normalising over temporal windows similarly to how batch norm does over batches. It has been shown that TCN is fundamental for ESBNs in the original paper, and also improves performance for non-ESBN architecture in the tasks outlined above. To provide as representative a comparison as possible, we also include TCN in our architectures. However, we further show in Section 4.1.2 that its inclusion may not be as essential.

To highlight our intended comparison between LSTM and VSA-enhanced MLPs, we focus here only on the key pathway. We manipulate it as follows: we replace (cfr. Figure 1, bottom panel) the key encoder f_s with a simpler feed-forward neural network instead of an LSTM, and we add a positional encoding component to its input. This allows us to emulate the time progression of an LSTM with a much more economical and well-behaved alternative. In Section 3, we analyse in more detail how these modifications are implemented and how the behaviour of the model changes.

2.2.2. Tasks

The tasks (cfr. Figure 2) chosen in the original work serve to showcase the ability of the ESBN to infer abstract rules from visual cues in varying degrees of difficulty. Generally, visual reasoning datasets like CLEVR [25] and RAVEN [26] represent the litmus test for architectures — be they neurosymbolic or not — which purport to approach human-like cognition capabilities. The same tasks are, therefore, particularly suited for evaluating the performance of our architecture as well.



(a) Same/different task (b) Relational match-to-sample task (c) Distribution of three (d) Identity rules

Figure 2: Summary of the tasks. Figure adapted from [13].

It has been shown [27] that conventional neural networks are unable to solve even extremely simple visual reasoning tasks like the same/different task, in which the model is asked to determine whether two shapes are equal. An ESBN is not only capable of successfully determining this straightforward relationship, but can also extend it to novel image pairs such as in the relational match-to-sample task. Finally, a subset of RAVEN progressive matrices [28] can be used to test cognition more at a human level with the distribution of three and identity rules

tasks. These two final tasks put some strain on both the ESBN and our architecture, which further serves to highlight the difference in performance.

For each scenario a certain number of samples is held out for testing, with percentages varying from 0% to 98%. Naturally, training with fewer samples proves more difficult and also emphasises the sample-efficiency of the different architectures.

3. Approach

Given the minimalism of the value pathway, we focus our efforts on the key pathway. As explained in Section 2.2, keys are generated by feeding the attention weighted superposition of previous keys to the LSTM. This process intrinsically keeps tabs on the order in which it receives its inputs. We hypothesise this ordered structure to be crucial to the success of the model such that a straightforward replacement with a non-recurrent architecture would not be suitable. To preserve the properties outlined above, we introduce a new model which combines a more amenable representation of time as a discrete set of vectors (positional encoding through bundling) with an efficient feedforward network which makes use of state-of-the-art activation functions such as SIREN [29].

To better elucidate the key elements of our model we provide a 'in the footsteps' view of a key (cfr. k_r in Figure 1) that has been newly generated by superposition. First, is is bundled with the corresponding time-representing vector, in order to relocate it into our structured space. Then it is fed through an MLP which extracts the relevant features. Finally, it is passed to the activation function which favourably transforms these features to be added back onto the key stack.

3.1. Bundling

As discussed in Section 2.1, the bundling operation of our choice is component-wise addition. The goal of this step is to embed k_r with information about its position in the sequence, in such a way that a non-recurrent network should accurately infer the corresponding time-step from the key alone. We hypothesise that an appropriate VSA model is able to structure the space so that this embedding is successful. On its own, bundling is not sufficient to completely identify a VSA implementation. For this reason we also need to define the code-book (b in Figure 1). The code-book does not easily yield itself to a generalised representation, so we try three variants to avoid possible missteps.

The first and easiest setup amounts to choosing random vectors (*Rand* strategy) in a high dimensional space, which *a priori* have almost no similarity to each other. This approach allows us to maximally separate the time-steps in latent space. The second retraces the positional encoding of the Transformer (*TF* strategy). For more information on the benefit of this choice refer to [24]. The third and final code-book is created using fractional power encoding (*FPE* strategy) [16]. This method can generate any number of vectors with a controlled distance to each other. This specific property, when used to create equally-spaced codes, embeds a sequential structure in the space which more closely follows the concept of time in RNNs.

Once the key vector has been bundled (i.e. summed) with the code vector, it contains information both about the relationship with previous keys and its position in the sequence. It

is then fully equipped to be correctly interpreted by a feedforward network and does necessitate any recurrent structure. The dimensionality of each code vector needs to be equal to the key, which in our case is 256. As control the same experiments are performed with no code-book at all, meaning the position in the sequence is unknown to the MLP and can at most be inferred by correlation with the other keys.

3.2. MLP

As a drop-in replacement to LSTMs for feature extraction we chose an MLP, which performs well on unstructured data and is also efficient parameter-wise. Unstructured here means that, in contrast to the original image samples, keys do not necessarily possess any visual patterns which models such as CNN are better suited to pick up. The model must be, however, powerful enough to understand the bundled structure and decompose it in order to correctly process the key. This choice gives us maximum flexibility and parameter efficiency, while also affording us optimal performance during testing. The only parameters we vary in the MLP are its hidden and output activation functions, which we explore in the following section.

3.3. Activation function

Given the abundance of activation functions in the MLP space, we chose to test three variants. The two most common choices for many models are *tanh* and *ReLU*. ReLU is a biologically inspired activation function which squashes negative values while being unbounded on the positive semi-axis. By contrast, the hyperbolic tangent is a well-behaved sigmoid function which has the benefit of being bounded, and as such tends to lessen the effect of gradient issues. It was chosen as control given ReLU is used in the original ESBN.

Recently there has been renewed interested in sinusoidal activation functions, for example SIREN [29]. It has also been shown that these functions are particularly useful in reconstructing complex spaces by preserving some underlying mathematical structures such as first and second derivatives. This adaptability to time-dependent signals made SIREN an important comparator in our testing strategy.

As output of the MLP we now have a vector that is analogue to the output of the LSTM, and can therefore be used to obtain predictions by following the rest of the key pathway laid out in an ESBN. To evaluate the different architectural choices, and trade blows with the state-of-the art, we use the tasks outlined in Section 2.2.2.

4. Results

Table 1 reports a summary of all results on the maximum holdout for each task and architecture. Fully detailed Tables are presented in the Appendix.

Given the popularity of Transformers and their *a priori* aptness for such attentional focused tasks, we sought to reproduce the results of [13] for a Transformer encoder architecture (TFEnc). Our results were consistent with previous reports and further confirmed that Transformers do not represent a compelling alternative to ESBNs.

Table 1
Accuracy (%) with standard deviation based on ten runs of each architecture on the maximum holdout per task. LSTM refers to the original ESBN, MLP to our architectures. We also provide a comparison with a purely attentional model — in the form of a Transformer encoder — referred to as TFEnc. In green the best performing architecture based on the average across all tasks and holdouts (see Appendix for details). † indicates the original architectures from [13].

Model	Activation	Positional encoding	same_diff	rmts	dist3	identity_rules
LSTM	†	†	100.0 (0.0)	95.9 (1.4)	99.5 (0.4)	99.4 (0.6)
	None	None	100.0 (0.0)	100.0 (0.0)	97.9 (1.0)	99.8 (0.1)
MLP	SIREN	FPE	100.0 (0.0)	100.0 (0.0)	99.4 (0.5)	98.8 (0.6)
	SIREN	TF	100.0 (0.0)	100.0 (0.0)	98.7 (1.2)	99.3 (0.5)
	SIREN	None	100.0 (0.0)	100.0 (0.1)	98.9 (0.8)	66.3 (0.4)
	SIREN	Rand	100.0 (0.0)	100.0 (0.0)	99.7 (0.3)	99.1 (0.9)
	ReLU	FPE	100.0 (0.0)	100.0 (0.0)	94.5 (2.1)	90.8 (2.2)
	ReLU	None	100.0 (0.0)	100.0 (0.0)	95.8 (2.0)	63.2 (0.7)
	Tanh	FPE	100.0 (0.0)	100.0 (0.0)	96.2 (7.6)	98.6 (0.9)
TFEnc	†	†	58.7 (12.0)	79.1 (7.5)	25.0 (0.3)	32.8 (3.6)

Our architectures proved fully competitive with the original ESBN, with the sinusoidal activation function being clearly superior to the alternatives across the board. Initially we hypothesised this to be due to the unbounded nature of ReLU, but after observing the same behaviour on the bounded Tanh, we believed it to be the case that a periodic activation produces better latent neural representations than a monotone one. This phenomenon corroborates the findings of [29].

All positional encoding schemes behaved similarly, with *Rand* outperforming the alternatives on the maximum holdout. A more detailed analysis revealed that this observation does not hold in general, and in fact fractional power encoding has an edge when considering all holdouts (see Table 4). For this reason, we only test ReLU and Tanh activations with FPE encoding. Both SIREN and ReLU MLP architectures without positional encoding appeared to be unable to solve the *identity_rules* task, which validated our hypothesis that notion of position is key in replacing the LSTM

One must note that in the original ESBN, the attention vector w_t was computed by means of the dot product. While in principle a valid choice, dot product is not robust as it can overshoot and lead to issues in both gradient computations and consistency between different samples. Normalisation is a common choice in these cases, and here we specifically chose to use the cosine similarity, which is bounded between -1 and 1. To ensure that such a modification does not significantly hinder performance, we reproduced the results of [13] both with dot product and cosine similarity. We found that cosine similarity significantly improves performance on the *rmts* task bringing it up to 100%. All other models were tested using cosine similarity. The learning rate is kept constant at 5^{-5}.

Interestingly SIREN outperformed every other variant, with FPE and Transformer encoding

performing especially well and often surpassing the original model. By keeping the same hyperparameters across the original ESBN and our models, we are able to report a $\sim 2.4\times$ reduction in size[1]. This also improved results across the board by a straightforward implementation of VSA bundling.

4.1. Ablations

We investigate the limits of model size on performance and the relevance of temporal context normalisation for our architectures.

4.1.1. Model size

Given we achieved a $\sim 2.4\times$ size reduction without loss of performance while keeping the same hyperparameters, we next explored how much we could shrink both the original ESBN and our MLP model while keeping approximately the same performance. Moreover, we investigated whether instead increasing the size of the MLP to match the LSTM could yield further performance improvements at the expense of training speed. This appears unlikely as accuracy is already saturated in the current state.

Table 2 indicates that the two models do not scale the same. Our model can be reduced to a smaller size maintaining almost perfect accuracy, while the LSTM starts underperforming significantly, especially on the *same_diff* task. Increasing the parameter space did not yield increased performance, as predicted.

Table 2
Accuracy (%) with standard deviation based on ten runs of each architecture on the maximum holdout per task. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported.

Model	Parameters	same_diff	rmts	dist3	identity_rules
LSTM	300k	79.9 (33.2)	92.3 (1.2)	96.3 (7.4)	95.3 (2.5)
MLP	300k	100.0 (0.0)	100.0 (0.0)	99.2 (0.6)	99.0 (1.0)
	1.9M	100.0 (0.0)	100.0 (0.0)	98.9 (1.4)	99.1 (0.4)

4.1.2. Temporal context norm

In the original work of [13], the temporal context norm plays a crucial role in the performance of ESBNs and the other architectures tested. In particular on the maximum holdouts per task it increases accuracy from near-chance to near-perfect level.

Given its importance, we tested our MLP model including its usage for a fair comparison. In addition, we also investigated whether TCN can be done away with in our architecture. To evaluate the regularising effect of TCN, we tried to both increase and decrease model size as in the previous ablation (Table 3).

[1]The original ESBN has ~ 1.9 M parameters, while our SIREN model with fractional power encoding has ~ 800 k.

Table 3
Accuracy (%) with standard deviation based on ten runs of each architecture on the maximum holdout per task. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported. Temporal context normalisation is not used.

Model	Parameters	same_diff	rmts	dist3	identity_rules
LSTM	1.9M	50.1 (0.1)	51.0 (0.5)	62.0 (4.0)	95.2 (0.4)
MLP	300k	41.3 (20.2)	64.5 (19.0)	91.3 (5.6)	88.0 (7.7)
	800k	52.9 (4.9)	90.0 (3.0)	84.4 (10.6)	79.8 (4.3)
	1.9M	51.1 (1.0)	88.0 (5.8)	72.1 (13.3)	77.41 (16.3)

A marked drop in performance was observed, especially in the *same_diff* task. Despite this, however, the accuracy stayed well above chance level and surpassed the original LSTM-based ESBN in both the *rmts* and *dist3* tasks. Increasing model size did not yield increased accuracy. On the contrary, decreasing model size regularises in its own right, and hence increases performance. These results show that while TCN is an effective regulariser, it is not a necessary enabler of ESBN performance. For further analysis, refer to Section B.2.

5. Discussion

In this work, we have shown that enhancing simple MLP with VSA yields an effective and efficient alternative to LSTM when applied to ESBNs by simplifying the architecture and enhancing performance. Furthermore, we have demonstrated that a structured representation of time-ordering is entirely capable of replacing the complex machinery of an LSTM without loss of generalisation capability. We also observed a clear saturation of results, whereby the more efficient models have displayed near-perfect results on all tasks and have made further distinguishing and improvement unfeasible. It would be, therefore, of interest to test ESBNs and our SIREN alternative on more challenging datasets such as the full RAVEN, and possibly expand to other non strictly reasoning tasks.

We have identified in the LSTM a key regularising effect which reduces the degrees of freedom of the vanilla ESBN, thus mitigating its marked over-parametrisation. At the same time we have demonstrated that it is feasible to straightforwardly reduce the model size through VSA-enhanced MLP, without losing performance. Following the same principle, we have also identified in the temporal context norm another regulariser, which proves fundamental for an LSTM but not so for MLPs. This finding leads us to hypothesise that any sufficiently powerful regularising technique can be employed in place of the TCN, but further investigation would be needed for confirmation.

The recently found similarities between attentional models and symbolic memories such as SDM might shed more light towards the replacement schema we have presented here, and future work should focus on more complex tasks and architecture. Specifically, it would be interesting to explore to what extent embedding VSA into other symbolic framework might enhance performance and yield results similar or better to other more conventional architectures.

Acknowledgments

This work is supported by the Swiss National Science foundation (SNF), grant no. 200800.

References

[1] J. Feldman, The neural binding problem(s), Cognitive Neurodynamics 7 (2012) 1–11. doi:10/gf7sk7.

[2] E. H. Land, J. J. McCann, Lightness and retinex theory, Journal of the Optical Society of America 61 (1971) 1. doi:10/dq75zw.

[3] E. Adelson, A. Pentland, The perception of shading and reflectance, in: Perception as Bayesian Inference, Cambridge University Press, 1996, pp. 409–424. doi:10/gmx7vx.

[4] N. Chomsky, On certain formal properties of grammars, Information and Control 2 (1959) 137–167. doi:10/fw3f8r.

[5] R. Jackendoff, Foundations of Language, Oxford University PressOxford, 2002. doi:10/fc4r6s.

[6] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artificial Intelligence 46 (1990) 159–216. doi:10/bvnjss.

[7] K. Greff, S. van Steenkiste, J. Schmidhuber, On the binding problem in artificial neural networks, CoRR abs/2012.05208 (2020). arXiv:2012.05208.

[8] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, J. Tenenbaum, Neural-symbolic vqa: Disentangling reasoning from vision and language understanding, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018.

[9] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, J. Wu, The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision, in: International Conference on Learning Representations, 2019.

[10] G. Karunaratne, M. Schmuck, M. L. Gallo, G. Cherubini, L. Benini, A. Sebastian, A. Rahimi, Robust high-dimensional memory-augmented neural networks, Nature Communications 12 (2021). doi:10/gk3vvk.

[11] M. K. Sarker, L. Zhou, A. Eberhart, P. Hitzler, Neuro-symbolic artificial intelligence, AI Communications 34 (2022) 197–209. doi:10/grxtkd.

[12] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, A. Rahimi, A neuro-vector-symbolic architecture for solving raven's progressive matrices, Nature Machine Intelligence 5 (2023) 363–375. doi:10.1038/s42256-023-00630-8.

[13] T. W. Webb, I. Sinha, J. Cohen, Emergent symbols through binding in external memory, in: International Conference on Learning Representations, 2021.

[14] P. Kanerva, Sparse distributed memory, MIT press, 1988.

[15] T. Bricken, C. Pehlevan, Attention approximates sparse distributed memory, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, volume 34, Curran Associates, Inc., 2021, pp. 15301–15315.

[16] T. A. Plate, Holographic recurrent networks, in: S. Hanson, J. Cowan, C. Giles (Eds.), Advances in Neural Information Processing Systems, volume 5, Morgan-Kaufmann, 1992.

[17] T. A. Plate, Holographic Reduced Representation: Distributed Representation for Cognitive Structures, Center for the Study of Language and Information, 2003.
[18] R. W. Gayler, Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience (2004). doi:10/grxtkg.
[19] P. Kanerva, Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors, Cognitive Computation 1 (2009) 139–159. doi:10/dfm9nw.
[20] D. Kleyko, D. A. Rachkovskij, E. Osipov, A. Rahimi, A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations, ACM Computing Surveys 55 (2022) 1–40. doi:10/grktgq.
[21] D. Kleyko, D. Rachkovskij, E. Osipov, A. Rahimi, A survey on hyperdimensional computing aka vector symbolic architectures, part II: Applications, cognitive models, and challenges, ACM Computing Surveys 55 (2023) 1–52. doi:10/grtc8c.
[22] M. A. Kelly, D. Blostein, D. J. K. Mewhort, Encoding structure in holographic reduced representations., Canadian Journal of Experimental Psychology / Revue canadienne de psychologie expérimentale 67 (2013) 79–93. doi:10/f42jpw.
[23] M. Talagrand, A new look at independence, The Annals of Probability 24 (1996). doi:10/c22dpm.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. doi:10/gpnmtv.
[25] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, R. Girshick, Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, in: CVPR, 2017.
[26] C. Zhang, F. Gao, B. Jia, Y. Zhu, S.-C. Zhu, Raven: A dataset for relational and analogical visual reasoning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[27] J. Kim, M. Ricci, T. Serre, Not-so-CLEVR: learning same–different relations strains feedforward neural networks, Interface Focus 8 (2018) 20180011. doi:10/gf6mw7.
[28] J. C. Raven, Raven standard progressive matrices, 1936. doi:10/grxtkf.
[29] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, 2020. doi:10/grxtkc.

6. Appendices

A. Supplementary results

Table 4 shows the average performance across all holdouts. While there is clear saturation towards perfect accuracy, SIREN activation and FPE positional encoding prove to be the best combination. LSTM with cosine similarity outperforms its sibling with dot product in all but one task, *same_diff*, where it drastically underperforms on the $m = 95$ holdout (cfr. Table 5). This behaviour has been validated on 100 runs and cannot be reproduced on other architectures.

Table 4
Average performance (%) of each architecture for each task on all holdouts. Final full average is presented in the last column. In green the best architecture based on the overall result, which is bolded. † indicates the ESBN from [13] with dot product similarity.

Model	Activation	Positional encoding	same_diff	rmts	dist3	identity_rules	Overall
LSTM	†	†	100	99.0	99.2	99.7	99.5
	None	None	94.0	100.0	99.5	99.9	98.3
MLP	SIREN	FPE	100.0	100.0	99.8	99.6	**99.8**
	SIREN	TF	99.0	100.0	99.5	99.7	99.6
	SIREN	None	100.0	100.0	99.7	66.5	91.5
	SIREN	Rand	99.0	100.0	99.8	99.7	99.6
	ReLU	FPE	100.0	100.0	98.2	96.9	98.8
	ReLU	None	100.0	100.0	98.6	65.5	91.0
	Tanh	FPE	100.0	100.0	98.9	99.5	99.6

On the *identity_rules* task (cfr. Tables 11,12) all variants perform relatively similarly, with SIREN and tanh matching the vanilla model in all instances. ReLU is competitive, but on the hardest 95% holdout performance drops by about 9% w.r.t. to SIREN. All variants without positional encoding do not solve the task successfully, validating our hypothesis that a structured space significantly improves results and is key in replacing the LSTM.

On *dist3* (cfr. Tables 9,10) SIREN surpasses LSTM, with ReLU and tanh trailing behind by about 4 percentage points in the maximum holdout. Interestingly here a lack of positional encoding does not seem to harm performance.

Tables 7,8 and 5,6 show that all tested architectures find the *same_diff* and *RMTS* tasks particularly easy, and all perform very well (except LSTM with cosine similarity as mentioned above).

Table 5: Performance comparison on *same_diff* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. The average across all holdouts is also reported, with the best values bolded. † indicates the ESBN from [13] with dot product similarity.
No architecture, except the LSTM with cosine similarity, has any issue on this task.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95	m = 98	Average
LSTM	†	†	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	None	None	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	69.98 (24.51)	100.00 (0.00)	94.00
MLP	SIREN	FPE	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	SIREN	TF	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	95.04 (14.89)	100.00 (0.00)	99.01
	SIREN	None	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	SIREN	Rand	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	95.00 (15.00)	100.00 (0.00)	99.00
	ReLU	FPE	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	ReLU	None	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	Tanh	FPE	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**

Table 6: Performance comparison on *same_diff* task. For each variant and holdout percentage the loss is reported together with the standard deviation in parentheses. In each case 10 runs are completed. † indicates the ESBN from [13] with dot product similarity.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95	m = 98
LSTM	†	†	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.02 (0.00)	0.15 (0.01)
	None	None	0.00 (0.00)	0.00 (0.00)	0.37 (0.03)	0.68 (0.00)	0.68 (0.00)
MLP	SIREN	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.07 (0.01)	0.07 (0.02)
	SIREN	TF	0.00 (0.00)	0.00 (0.00)	0.04 (0.01)	0.35 (0.06)	0.31 (0.06)
	SIREN	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.01 (0.00)	0.07 (0.01)
	SIREN	Rand	0.00 (0.00)	0.00 (0.00)	0.03 (0.00)	0.34 (0.08)	0.38 (0.17)
	ReLU	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.16 (0.03)	0.41 (0.03)
	ReLU	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.29 (0.04)	0.53 (0.01)
	Tanh	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.08 (0.03)	0.13 (0.07)

Table 7: Performance comparison on *rmts* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. The average across all holdouts is also reported, with the best values bolded. † indicates the ESBN from [13] with dot product similarity.
No architecture has any issue on this task.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95	Average
LSTM	†	†	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	95.95 (1.38)	98.99
	None	None	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
MLP	SIREN	FPE	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	SIREN	TF	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	SIREN	None	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.98 (0.06)	99.99
	SIREN	Rand	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	ReLU	FPE	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	ReLU	None	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**
	Tanh	FPE	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	**100.00**

Table 8
Performance comparison on *rmts* task. For each variant and holdout percentage the loss is reported together with the standard deviation in parentheses. In each case 10 runs are completed. † indicates the ESBN from [13] with dot product similarity.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95
LSTM	†	†	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.22 (0.12)
	None	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
MLP	SIREN	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	SIREN	TF	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	SIREN	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	SIREN	Rand	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	ReLU	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	ReLU	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Tanh	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)

Table 9: Performance comparison on *dist3* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. The average across all holdouts is also reported, with the best values bolded. † indicates the ESBN from [13] with dot product similarity.
SIREN architectures and the original ESBN achieve results similar to each other and outperform the others.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95	Average
LSTM	†	†	98.91 (1.03)	99.13 (0.68)	99.07 (1.13)	99.52 (0.40)	99.16
	None	None	100.00 (0.00)	99.99 (0.01)	99.94 (0.06)	97.90 (0.97)	**99.46**
MLP	SIREN	FPE	100.00 (0.00)	99.95 (0.05)	99.90 (0.08)	99.41 (0.49)	99.81
	SIREN	TF	100.00 (0.00)	99.92 (0.17)	99.57 (0.28)	98.71 (1.17)	99.55
	SIREN	None	100.00 (0.00)	99.95 (0.10)	99.85 (0.09)	98.89 (0.79)	99.67
	SIREN	Rand	100.00 (0.00)	99.98 (0.03)	99.67 (0.41)	99.68 (0.28)	**99.83**
	ReLU	FPE	99.95 (0.03)	99.72 (0.14)	98.84 (0.32)	94.48 (2.14)	98.25
	ReLU	None	99.97 (0.02)	99.82 (0.08)	98.95 (0.52)	95.78 (1.99)	98.63
	Tanh	FPE	99.97 (0.03)	99.89 (0.11)	99.43 (0.53)	96.17 (7.63)	98.86

Table 10
Performance comparison on *dist3* task. For each variant and holdout percentage the loss is reported together with the standard deviation in parentheses. In each case 10 runs are completed. † indicates the ESBN from [13] with dot product similarity.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95
LSTM	†	†	0.04 (0.03)	0.03 (0.02)	0.04 (0.05)	0.06 (0.05)
	None	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.07 (0.03)
MLP	SIREN	FPE	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.02 (0.02)
	SIREN	TF	0.00 (0.00)	0.00 (0.01)	0.02 (0.02)	0.05 (0.04)
	SIREN	None	0.00 (0.00)	0.00 (0.01)	0.01 (0.00)	0.04 (0.03)
	SIREN	Rand	0.00 (0.00)	0.00 (0.00)	0.02 (0.03)	0.01 (0.01)
	ReLU	FPE	0.00 (0.00)	0.01 (0.01)	0.07 (0.03)	0.25 (0.13)
	ReLU	None	0.00 (0.00)	0.01 (0.00)	0.06 (0.04)	0.17 (0.10)
	Tanh	FPE	0.00 (0.00)	0.01 (0.01)	0.03 (0.03)	0.10 (0.18)

Table 11: Performance comparison on *identity_rules* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. The average across all holdouts is also reported, with the best values bolded. Highlighted in grey are the architectures without positional encoding, which perform significantly worse than the rest. † indicates the ESBN from [13] with dot product similarity.
All architectures perform similarly, except when no positional encoding is provided. This clearly demonstrates that some bundling scheme is necessary to preserve time-ordering.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95	Average
LSTM	†	†	99.78 (0.32)	99.84 (0.25)	99.81 (0.18)	99.42 (0.63)	99.71
	None	None	100.00 (0.00)	100.00 (0.00)	99.94 (0.07)	99.85 (0.13)	**99.95**
MLP	SIREN	FPE	99.99 (0.01)	99.92 (0.06)	99.60 (0.23)	98.81 (0.61)	99.58
	SIREN	TF	99.98 (0.02)	99.91 (0.10)	99.52 (0.29)	99.32 (0.54)	99.68
	SIREN	None	66.72 (0.60)	66.58 (0.31)	66.35 (0.22)	66.28 (0.41)	66.48
	SIREN	Rand	99.97 (0.03)	99.95 (0.03)	99.75 (0.22)	99.14 (0.93)	**99.70**
	ReLU	FPE	99.69 (0.18)	99.39 (0.18)	97.77 (0.62)	90.83 (2.16)	96.92
	ReLU	None	66.69 (0.45)	66.43 (0.27)	65.81 (0.50)	63.19 (0.73)	65.53
	Tanh	FPE	99.99 (0.01)	99.92 (0.03)	99.49 (0.31)	98.59 (0.94)	99.50

Table 12
Performance comparison on *identity_rules* task. For each variant and holdout percentage the loss is reported together with the standard deviation in parentheses. In each case 10 runs are completed. Highlighted in grey are the architectures without positional encoding, which perform significantly worse than the rest. † indicates the ESBN from [13] with dot product similarity.

	Activation	Positional encoding	m = 0	m = 50	m = 85	m = 95
LSTM	†	†	0.01 (0.01)	0.01 (0.01)	0.03 (0.03)	0.08 (0.09)
	None	None	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.01 (0.01)
MLP	SIREN	FPE	0.00 (0.00)	0.00 (0.00)	0.02 (0.01)	0.06 (0.03)
	SIREN	TF	0.00 (0.00)	0.01 (0.01)	0.04 (0.03)	0.05 (0.05)
	SIREN	None	0.70 (0.05)	0.65 (0.05)	0.54 (0.03)	0.51 (0.01)
	SIREN	Rand	0.00 (0.00)	0.00 (0.00)	0.02 (0.02)	0.05 (0.07)
	ReLU	FPE	0.02 (0.01)	0.03 (0.01)	0.15 (0.07)	0.91 (0.29)
	ReLU	None	0.65 (0.07)	0.62 (0.04)	0.59 (0.06)	0.77 (0.12)
	Tanh	FPE	0.00 (0.00)	0.00 (0.00)	0.03 (0.02)	0.08 (0.06)

B. Ablation results

B.1. Model size

We report here the full results for the model size ablation. For all tasks (cfr. Tables 14,15,16) except *same_diff* accuracy remains high until the highest holdout. In the *same_diff* task (cfr. Table 14) we observe a proportional worsening of performance as holdout increases.

Table 13
Performance comparison on *same_diff* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported.

Model	Parameters	m = 0	m = 50	m = 85	m = 95	m = 98
LSTM	300k	100.00 (0.00)	100.00 (0.00)	90.06 (19.87)	84.97 (22.96)	79.92 (33.24)
MLP	300k	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
	1.9M	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)

Table 14
Performance comparison on *RMTS* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported.

Model	Parameters	m = 0	m = 50	m = 85	m = 95
LSTM	300k	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	92.34 (1.18)
MLP	300k	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
	1.9M	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)

Table 15
Performance comparison on *dist3* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported.

Model	Parameters	m = 0	m = 50	m = 85	m = 95
LSTM	300k	100.00 (0.00)	100.00 (0.01)	100.00 (0.00)	96.35 (7.40)
MLP	300k	100.00 (0.00)	99.97 (0.02)	99.72 (0.17)	99.19 (0.62)
	1.9M	99.99 (0.01)	99.97 (0.04)	99.70 (0.29)	98.89 (1.36)

Table 16
Performance comparison on *identity_rules* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported.

Model	Parameters	m = 0	m = 50	m = 85	m = 95
LSTM	300k	100.00 (0.00)	100.00 (0.00)	99.98 (0.06)	95.27 (2.49)
MLP	300k	99.99 (0.01)	99.82 (0.22)	99.64 (0.32)	98.94 (0.96)
	1.9M	100.00 (0.00)	99.96 (0.03)	99.70 (0.20)	99.06 (0.37)

B.2. Temporal context norm

We report here the full results for the temporal context norm ablation.

There is a clear trade-off between expressiveness and over-fitting, whereby TCN severely and efficiently constricts parametrisation to improve performance across the board. On the *same_diff* task, Table 17 shows that model expressiveness greatly benefits performance, with our MLP model surpassing the LSTM-based ESBN across all holdouts except the last, only significantly underperforming when the amount of data is very limited.

On the other hand, for the *identity_rules* task (cfr. Table 20) we observe that overparametrisation of the MLP model is punished by a reduction in accuracy. This is especially striking for the 1.9M parameters MLP. Here performance is not strongly correlated with holdout, even increasing with increasing holdout, indicating that a smaller model generalises better. This distinctive effect is confirmed by the fact that indeed the smallest MLP performs the best out of our models.

On the *dist3* and *identity_rules* tasks the smallest model performs the best, while it is the other way around for *same_diff* and *rmts*. This is once more an indicator that there is a strong need to strike a balance between expressiveness and the natural tendency to over-fit, role which TCN serves remarkably well.

Taken together, LSTM and TCN provide strong regularisation performance which balances even an highly overparametrised model like the original ESBN. This fact is highly beneficial, as seen in the near-perfect performance seen in [13], but is not necessary and reveals itself to be a hindrance in the more constrained cases shown in the ablations.

In fact our MLP model retains all of the expressive power and removes these limitations. It is then reasonable that any powerful enough regularisation could take TCN's place in our model, but further investigation in this direction is needed.

Table 17
Performance comparison on *same_diff* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported. Temporal context normalisation is not used.

Model	Parameters	m = 0	m = 50	m = 85	m = 95	m = 98
LSTM	1.9M	49.94 (0.07)	51.11 (2.31)	52.07 (6.48)	50.61 (1.46)	50.15 (0.21)
MLP	300k	81.19 (23.27)	86.52 (14.34)	78.14 (22.96)	63.08 (19.04)	41.34 (20.19)
	800k	100.00 (0.00)	92.95 (5.93)	90.15 (14.49)	76.46 (18.14)	52.89 (4.95)
	1.9M	100.00 (0.00)	89.80 (8.38)	97.56 (1.65)	76.25 (17.87)	51.07 (0.96)

Table 18
Performance comparison on *rmts* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported. Temporal context normalisation is not used.

Model	Parameters	m = 0	m = 50	m = 85	m = 95
LSTM	1.9M	96.36 (1.16)	90.00 (8.11)	49.92 (0.24)	50.65 (0.49)
MLP	300k	99.89 (0.03)	92.84 (4.44)	93.80 (2.70)	64.49 (18.97)
	800k	99.97 (0.02)	94.72 (2.35)	95.84 (1.22)	90.00 (3.03)
	1.9M	99.98 (0.03)	85.85 (18.55)	96.73 (2.10)	88.05 (5.77)

Table 19
Performance comparison on *dist3* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported. Temporal context normalisation is not used.

Model	Parameters	m = 0	m = 50	m = 85	m = 95
LSTM	1.9M	99.99 (0.01)	99.17 (0.31)	77.58 (2.80)	68.48 (16.78)
MLP	300k	99.99 (0.01)	96.19 (1.93)	95.92 (2.03)	91.26 (5.58)
	800k	99.97 (0.05)	87.09 (23.58)	91.33 (12.84)	84.37 (10.58)
	1.9M	99.83 (0.36)	89.30 (24.12)	82.96 (23.81)	72.14 (13.29)

Table 20
Performance comparison on *identity_rules* task. For each variant and holdout percentage the accuracy (%) is reported together with the standard deviation in parentheses. In each case 10 runs are completed. LSTM refers to the original ESBN, while MLP uses SIREN activation and fractional power encoding consistent with the best results in Table 1. The number of parameters for each architecture is also reported. Temporal context normalisation is not used.

Model	Parameters	m = 0	m = 50	m = 85	m = 95
LSTM	1.9M	100.00 (0.00)	99.37 (0.24)	97.44 (0.01)	96.05 (0.92)
MLP	300k	99.35 (1.55)	85.43 (25.10)	96.50 (1.20)	88.00 (7.75)
	800k	99.99 (0.02)	83.49 (27.27)	97.35 (2.00)	79.80 (4.32)
	1.9M	74.98 (35.37)	73.44 (33.93)	89.95 (13.60)	77.41 (16.27)