MEMORY-AUGMENTED RESERVOIR COMPUTING

Abstract

This seminar explores implementing **external working memories** to reservoir models, with a focus on theoretical aspects of computational **expressiveness**.

The landscape of recent memory-augmented reservoir computing is examined, showing the Reservoir memory machine (**RMM**) as well as the Reservoir stack machine (**RSM**).

State of the art **end-to-end models** are then briefly investigated, along with some interesting future research directions.

MOTIVATION

Neural networks with external memories

Neural Turing Machines[1] and **Differentiable Neural Computers**[2] have been explored extensively within the ML landscape.

The main motivation behind such systems is to enable the network to model **long term dependencies**, for which standard **RNN**s tend to fail.

1. Alex Graves, Greg Wayne, Ivo Danihelka (2014)
2. Alex Graves, et al. (2016)

MOTIVATION

Being notoriously hard to train in an **end-to-end** fashion (as in deep learning), the application of such networks is currently limited.

Here we will focus in particular on **memory-augmented Reservoir Computing** methods, which have been showing promising results lately.

Advantages

Some advantages of these randomized recurrent architectures include much **lower training and inference times**, compared to the their deep counterparts, along with some nice theoretical results.

MOTIVATION

Limitations

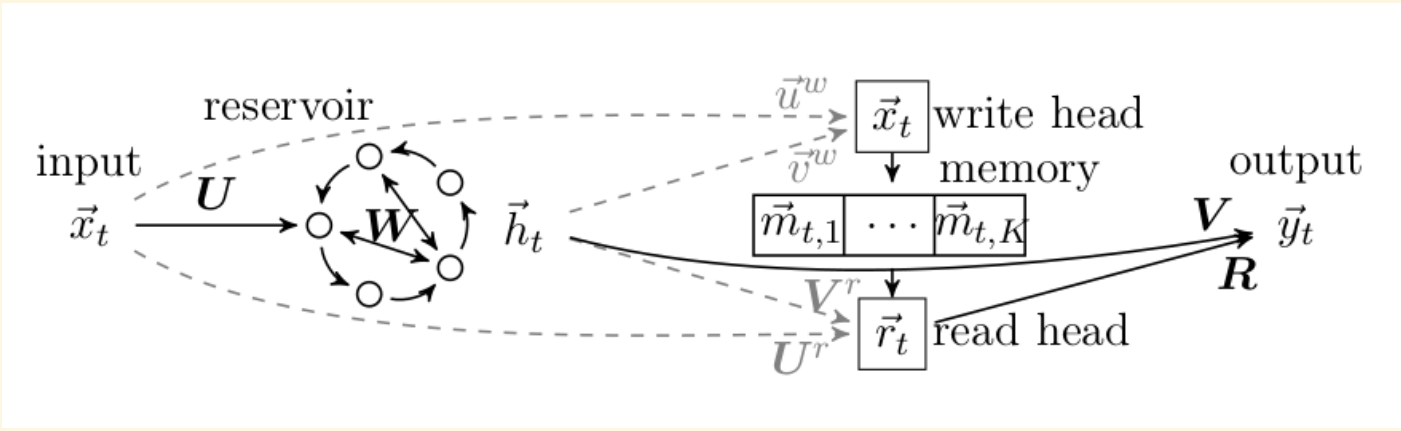
While proving useful in some scenarios, memory-augmented reservoir computing still hold onto some problems, like the need to produce **target memory control sequences** for the training algorithm. These aspects will be analyzed later on.

IDEA

Extending ESNs

The main idea behind memory-augmented reservoir computing, is that of extending Echo State Networks (ESN) with external memories.

By separating memory from computation we can thus design systems that are able to model arbitrarily long term dependencies in the input without interference[1].



1. Benjamin Paassen, Alexander Schulz (2020)

ECHO STATE NETWORKS

Echo State Networks are a kind of **Randomized RNN**s and are studied under the field of **Reservoir Computing**.

A **standard** echo state network follows the same update equations of a vanilla RNN:

Definition (*RNN equations*)

$$h_t = f(x_t, h_t) = \tanh(\mathbf{U} \cdot x_t + \mathbf{W} \cdot h_{t-1})$$
$$y_t = \mathbf{V} \cdot h_t$$

With the following differences:

1. $\mathbf{U}$ *and* $\mathbf{W}$ *are fixed*
2. *The network must hold the* ***Echo State Property***.

ECHO STATE PROPERTY

Provides a theoretical that the state of the reservoir will depend on the input and not on the **initial conditions**, with past influences degrading over time. This is related to the concept of **Markovianity**[1], which is typically achieved by designing the model to be a contractive map.

Definition (*Contractive reservoir*)

Let $\mathcal{R} = (\mathbf{U}, \mathbf{W}, b, \sigma, h_0)$ be a reservoir. We say $\mathcal{R}$ is *contractive* with constant $C \in (0, 1)$ if for any $h, h' \in \mathbb{R}^m$ and any $x \in \mathbb{R}^n$ it holds:

$$||f(x, h) - f(x, h')|| \leq C \cdot ||h - h'||$$

where f is defined as in the update equations of a vanilla RNN.

1. Claudio Gallicchio, Alessio Micheli (2011)

LIMITATIONS OF ESN

Memory capacity

Since echo state network hold the echo state property, the influence of past inputs is limited by the contractive properties of the network. As such, **long term dependency** tasks are out of reach of such models.

The memory capacity is limited by the number of neurons of the reservoir[1]

1. Igor Farkaš, Radomír Bosák, Peter Gergeľ (2016)

RESERVOIR MEMORY MACHINE

Echo state property

The echo state property is necessary in order for the network not to discriminate on initial conditions, but it forcefully limits its capability to model long term dependencies in the input.

For this reason, **Reservoir memory machines** have been proposed.

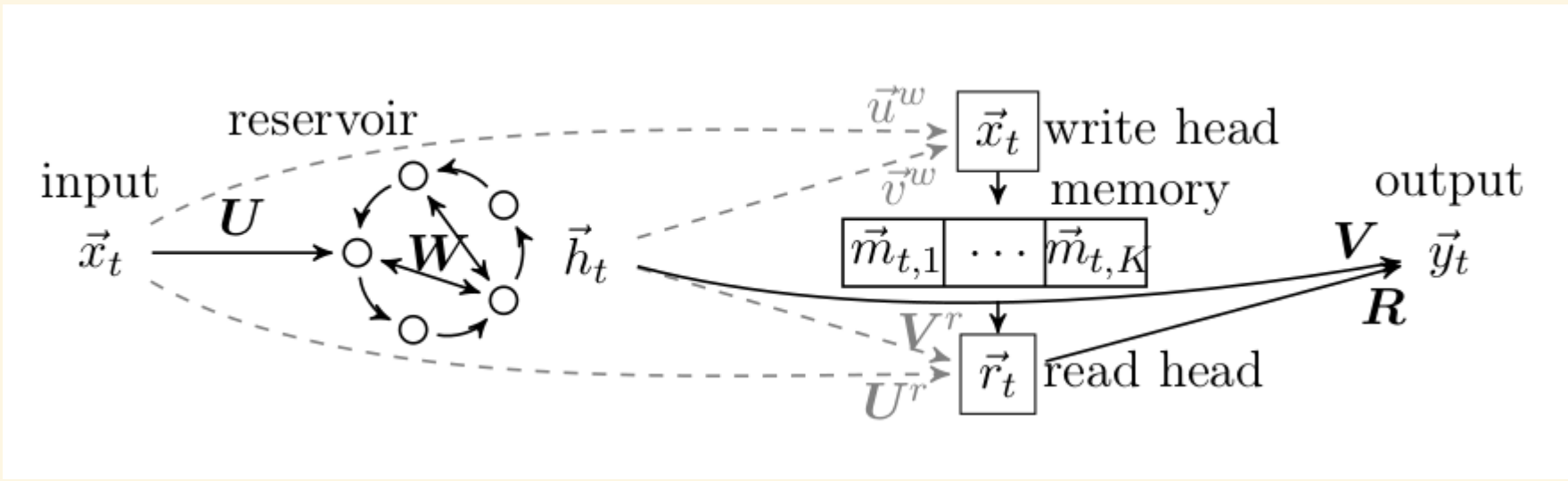
RESERVOIR MEMORY MACHINE

The vanilla **RMM**[1] has been introduced as an extension of standard ESNs. In particular the network is equipped with an external memory that it can access using a **read** and a **write** head.

1. Benjamin Paassen, Alexander Schulz (2020)

RESERVOIR MEMORY MACHINE

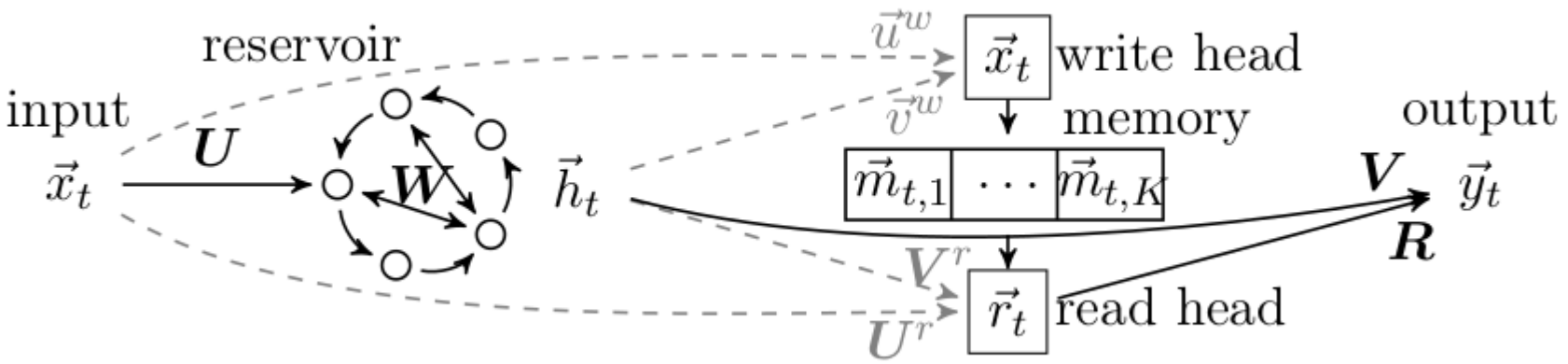
The vanilla **RMM**[1] has been introduced as an extension of standard ESNs. In particular the network is equipped with an external memory that it can access using a **read** and a **write** head.



1. Benjamin Paassen, Alexander Schulz (2020)

RESERVOIR MEMORY MACHINE

DYNAMICS



1. Copy previous memory state: $\mathbf{M}_t \leftarrow \mathbf{M}_{t-1}$
2. Compute write head discriminator: $c_t^w = u^w \cdot x^t + v^w \cdot h_t$
    1. if $c_t^w > 0$ then $\mathbf{M}_{t,k} \leftarrow x_t$ and $k_t \leftarrow k_{t-1} + 1$
3. Compute read head discriminator: $c_t^r = \mathbf{U}^r \cdot x_t + \mathbf{V}^r \cdot h_t$
    1. $l_t \leftarrow \begin{cases} l_t & \text{if } c_{t,1}^r = \max\{c_{t,1}^r, c_{t,2}^r, c_{t,3}^r\} \\ l_{t-1}+1 & \text{if } c_{t,2}^r = \max\{c_{t,1}^r, c_{t,2}^r, c_{t,3}^r\} \\ 1 & \text{otherwise} \end{cases}$
4. Set $r_t \leftarrow \mathbf{M}_{t,l_t}$
5. Compute the output of the network: $y_t = \mathbf{V} \cdot h_t + \mathbf{R} \cdot r_t$

RESERVOIR MEMORY MACHINE

DYNAMICS

1. Copy previous memory state: $\mathbf{M}_t \leftarrow \mathbf{M}_{t-1}$
2. Compute write head discriminator: $c_t^w = u^w \cdot x^t + v^w \cdot h_t$
    1. if $c_t^w > 0$ then $\mathbf{M}_{t,k} \leftarrow x_t$ and $k_t \leftarrow k_{t-1} + 1$
3. Compute read head discriminator: $c_t^r = \mathbf{U}^r \cdot x_t + \mathbf{V}^r \cdot h_t$
    1. $l_t \leftarrow \begin{cases} l_t & \text{if } c_{t,1}^r = \max\{c_{t,1}^r, c_{t,2}^r, c_{t,3}^r\} \\ l_{t-1} + 1 & \text{if } c_{t,2}^r = \max\{c_{t,1}^r, c_{t,2}^r, c_{t,3}^r\} \\ 1 & \text{otherwise} \end{cases}$
4. Set $r_t \leftarrow \mathbf{M}_{t,l_t}$
5. Compute the output of the network: $y_t = \mathbf{V} \cdot h_t + \mathbf{R} \cdot r_t$

Differentiability

Note that since the network is not trained in an end-to-end fashion, there's no need to make the memory mechanics differentiable.

RESERVOIR MEMORY MACHINE

DYNAMICS

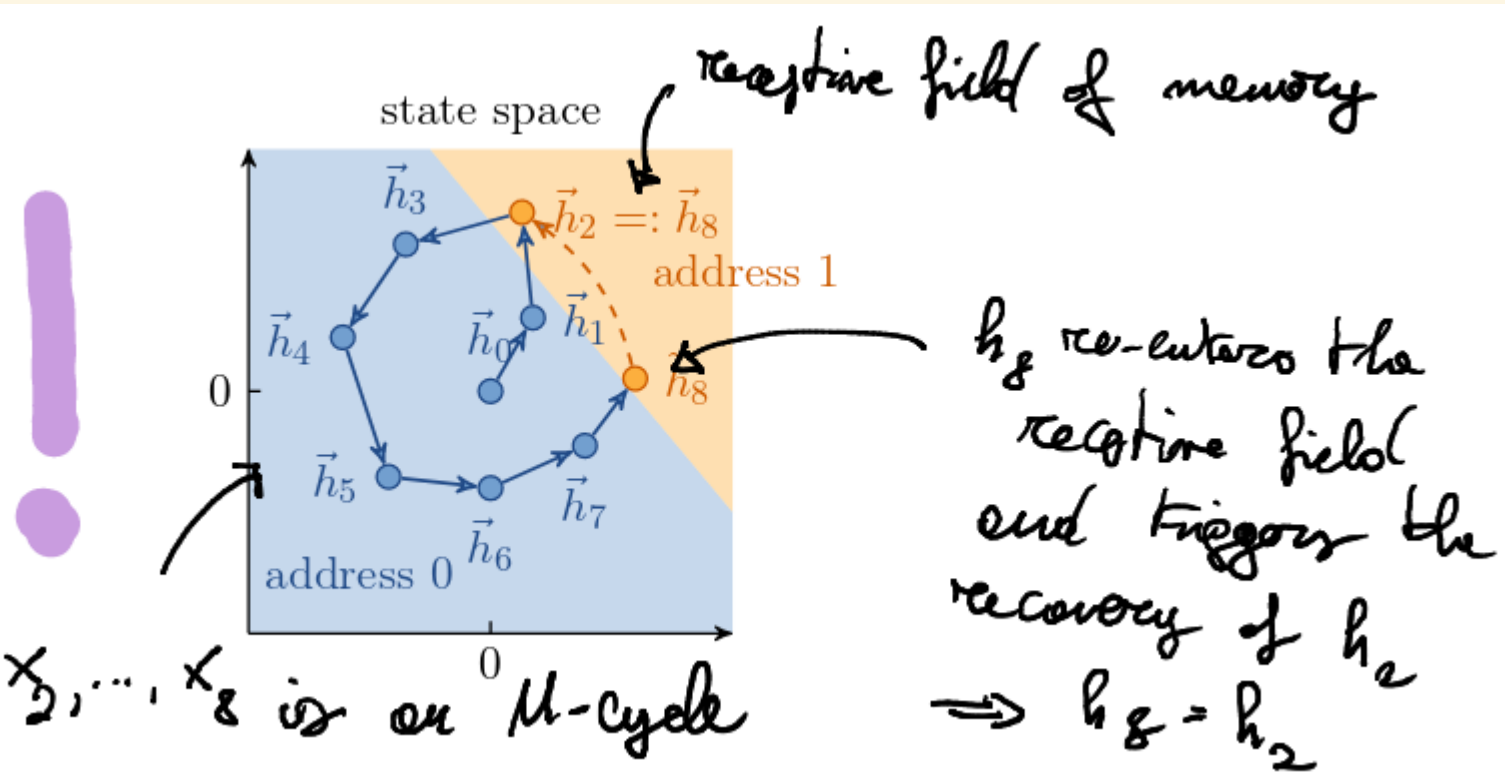
Improvements over base model

Imposing the memory's cells to be only **written once**, the RMM can use a **single head** for both reading and writing to memory

RESERVOIR MEMORY MACHINE

DYNAMICS

The RMM operates as a standard ESN until $a_t \neq 0$ for the first time



The first time h_t enters the receptive field of a memory address $a_t > 0$ (orange region), it is stored in memory. When later on the hidden state enters again the same receptive field, the previously stored data is retrieved (in the example, $h_8 = h_2$)

RESERVOIR MEMORY MACHINE

DYNAMICS

RMM dynamics

$$\tilde{h}_t = f(x_t, h_{t-1}) = \sigma\left(\mathbf{U} \cdot x_t + \mathbf{W} \cdot h_{t-1} + b\right)$$
$$a_t = c(\tilde{h}_t)$$
$$\mathbf{M}_{t,l} = \begin{cases} \tilde{h}_t & \text{if } l = a_t \text{ and } \mathbf{M}_{t-1,l} = 0 \\ \mathbf{M}_{t-1,l} & \text{otherwise} \end{cases}$$
$$h_t = \begin{cases} \tilde{h}_t & \text{if } a_t = 0 \\ \mathbf{M}_{t,a_t} & \text{otherwise} \end{cases}$$
$$y_t = g(h_t)$$

where $\mathbf{M}_{0,l} = 0$ except if $c(h_0) > 0$, for which $\mathbf{M}_{0,c(h_0)} = h_0$

RESERVOIR MEMORY MACHINE

TRAINING

Training details

- The authors recommend **Legendre delay networks** as reservoirs, because they are specifically designed to losslessly reconstruct past states.[1]
- They also have fewer hyperparameters and are **deterministically constructed**.

1. Aaron Voelker, Ivana Kajić, Chris Eliasmith (2019)

RESERVOIR MEMORY MACHINE

TRAINING

Training details

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.
-
-
-

RESERVOIR MEMORY MACHINE

TRAINING

Training details

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.
- The hidden states h_t are computed via the defined dynamics
-
-

RESERVOIR MEMORY MACHINE

TRAINING

Training details

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.
- The hidden states h_t are computed via the defined dynamics
- The addresses a_t are injected via teacher forcing, replacing the output of the classifier $c(\tilde{h}_t)$ during training.
-

RESERVOIR MEMORY MACHINE

TRAINING

Training details

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.
- The hidden states h_t are computed via the defined dynamics
- The addresses a_t are injected via teacher forcing, replacing the output of the classifier $c(\tilde{h}_t)$ during training.
- Finally, the output layer g is computed via linear regression, while the memory head c can be any classifier of choice. The authors use a support vector machine.

RESERVOIR MEMORY MACHINE

TRAINING

Training details

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.
- The hidden states h_t are computed via the defined dynamics
- The addresses a_t are injected via teacher forcing, replacing the output of the classifier $c(\tilde{h}_t)$ during training.
- Finally, the output layer g is computed via linear regression, while the memory head c can be any classifier of choice. The authors use a support vector machine.

RESERVOIR MEMORY MACHINE

TRAINING

Limitations

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.

RESERVOIR MEMORY MACHINE

TRAINING

Limitations

- A sequence $a_1, \cdots, a_T$ of target memory addresses is needed for every training sequence $x_1, \cdots, x_T$ in the dataset.

As is the case for every memory-augmented model presented here that do not make use of an end-to-end training regime, memory access dynamics require further user inputs to be learned.

RESERVOIR STACK MACHINE

RMMs as Finite state machines

It can be shown that **RMMs** can simulate any **FSM**, but fail to correctly represent *at least* some **LR(1)-automata**.

RESERVOIR STACK MACHINE

RMMs as Finite state machines

It can be shown that **RMMs** can simulate any **FSM**, but fail to correctly represent *at least* some **LR(1)-automata**.

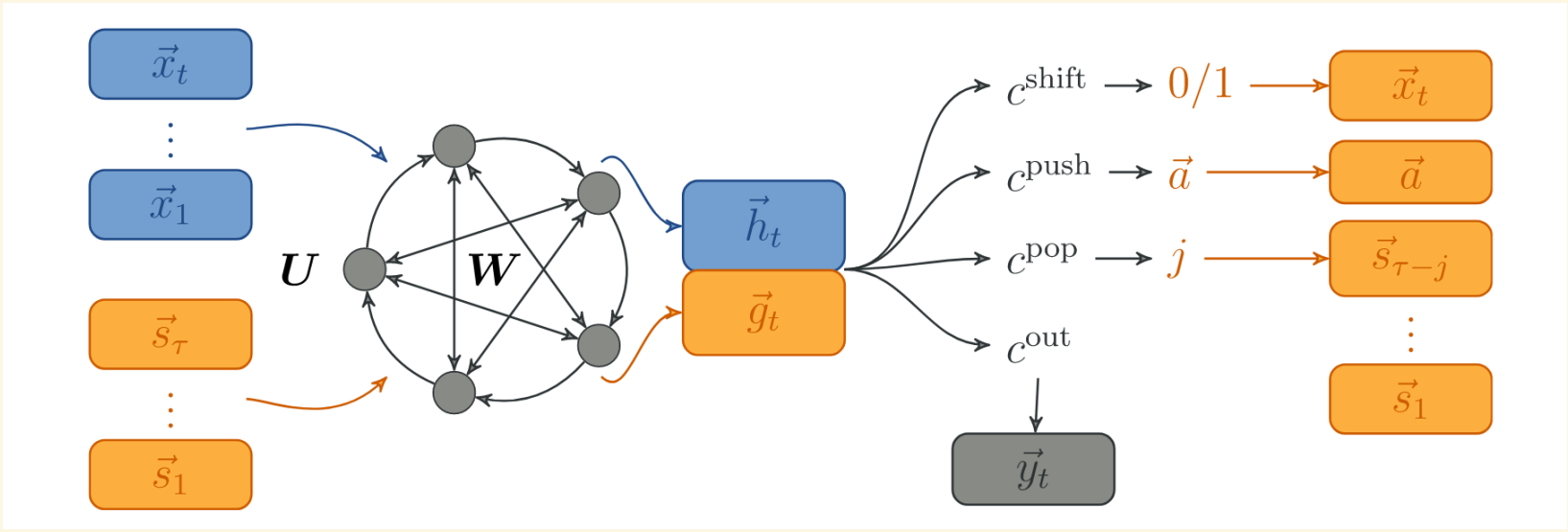
For this reason, instead of extending a network with a **fixed size** memory, it's reasonable to use a **dynamic stack** instead.

RESERVOIR STACK MACHINE

RMMs as Finite state machines

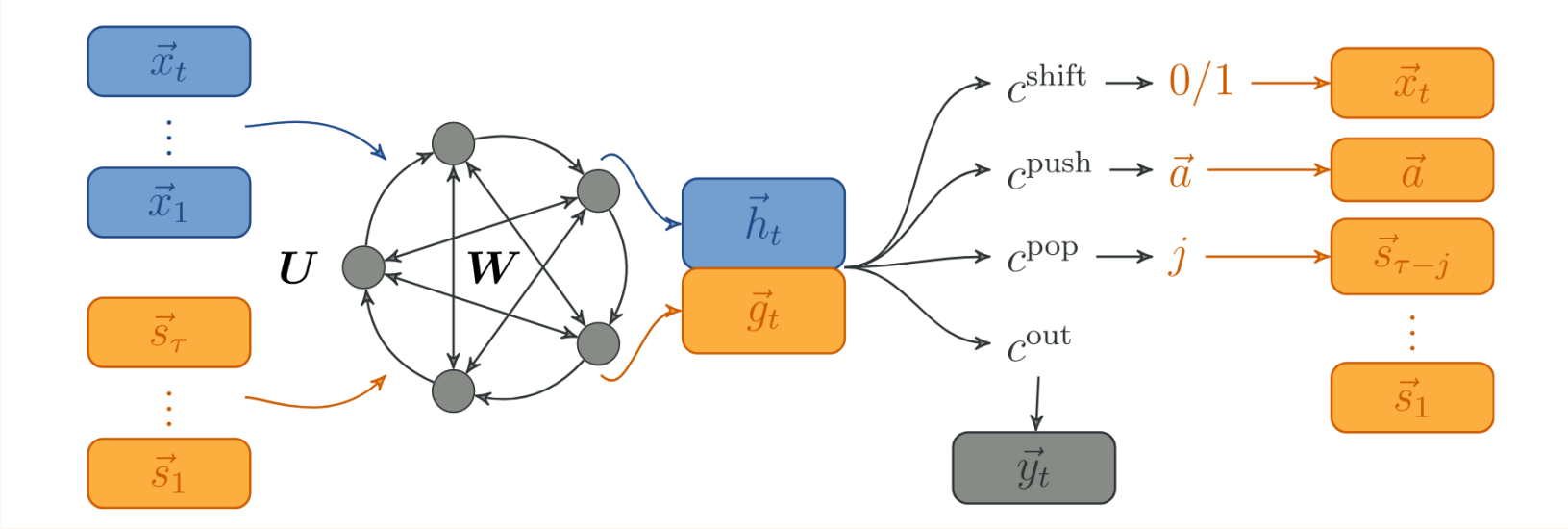
It can be shown that **RMMs** can simulate any **FSM**, but fail to correctly represent *at least* some **LR(1)-automata**.

For this reason, instead of extending a network with a **fixed size** memory, it's reasonable to use a **dynamic stack** instead.



RESERVOIR STACK MACHINE

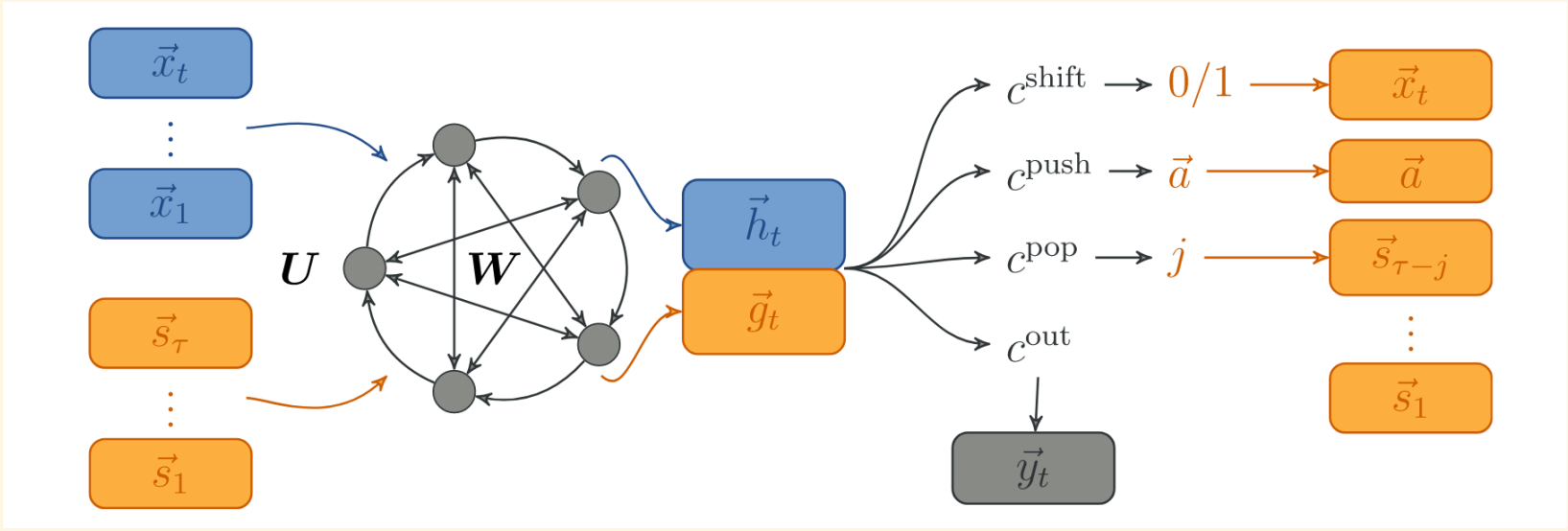
IDEA



At **each** time-step t:

RESERVOIR STACK MACHINE

IDEA

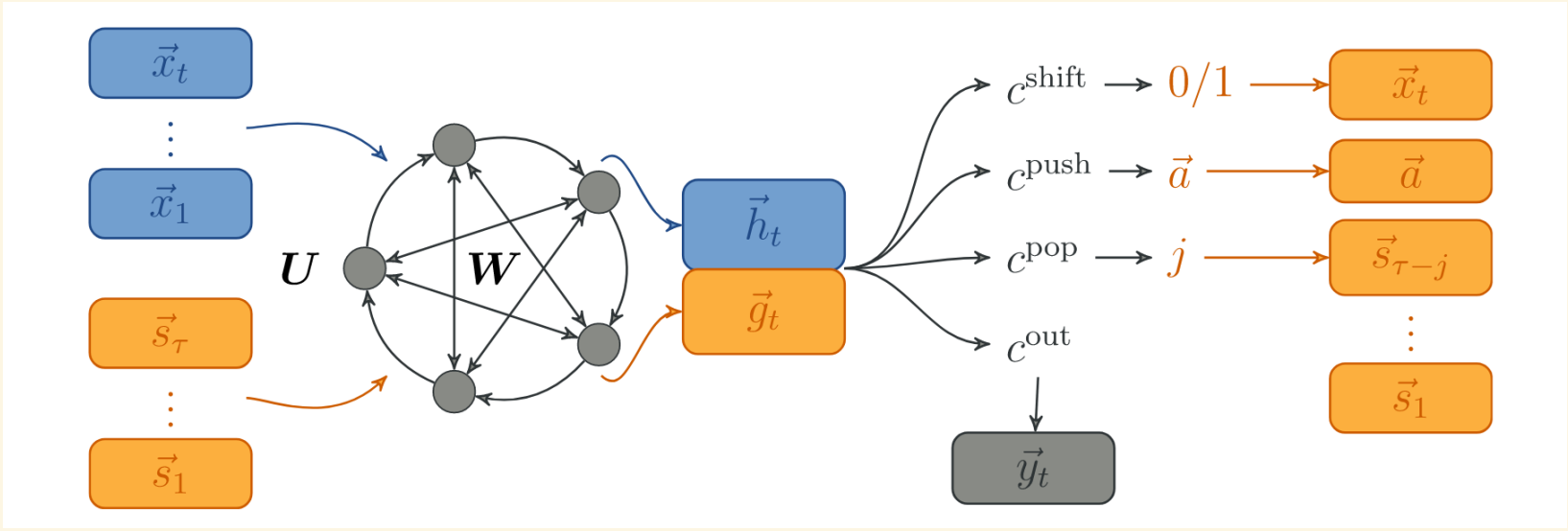


At **each** time-step t:

- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$

RESERVOIR STACK MACHINE

IDEA

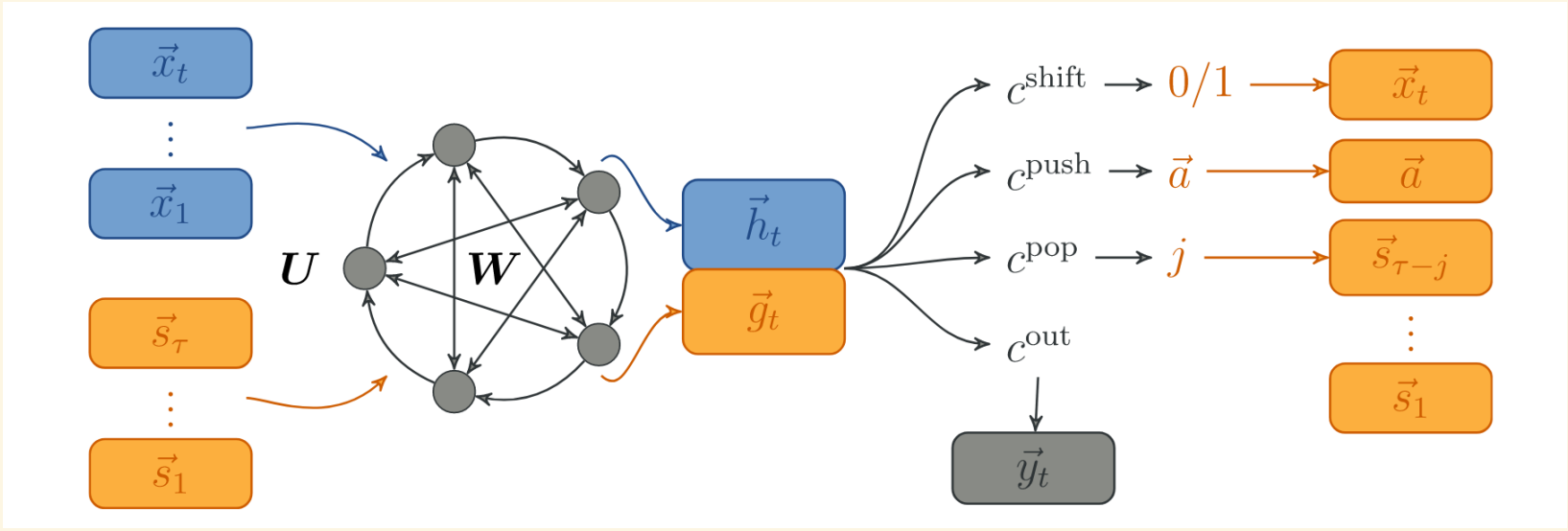


At **each** time-step t:

- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$
- c^{pop} decides what symbols (how many) to pop from the stack

RESERVOIR STACK MACHINE

IDEA

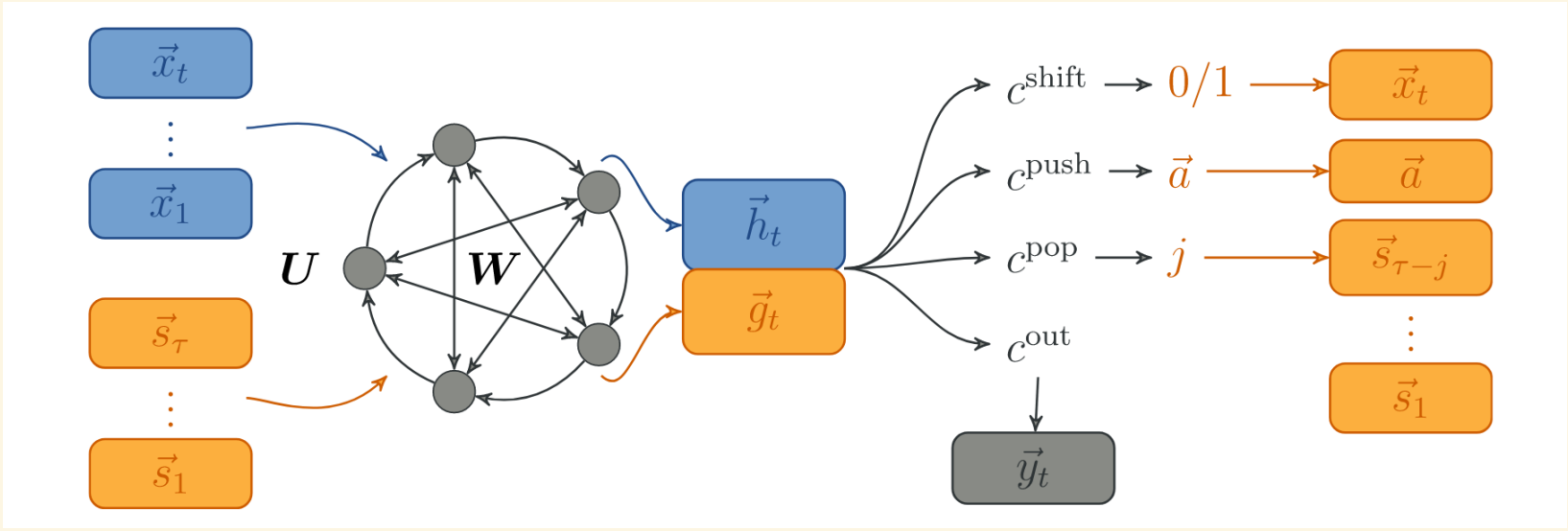


At **each** time-step t:

- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$
- c^{pop} decides what symbols (how many) to pop from the stack
- c^{push} decides which symbols to append onto the stack

RESERVOIR STACK MACHINE

IDEA

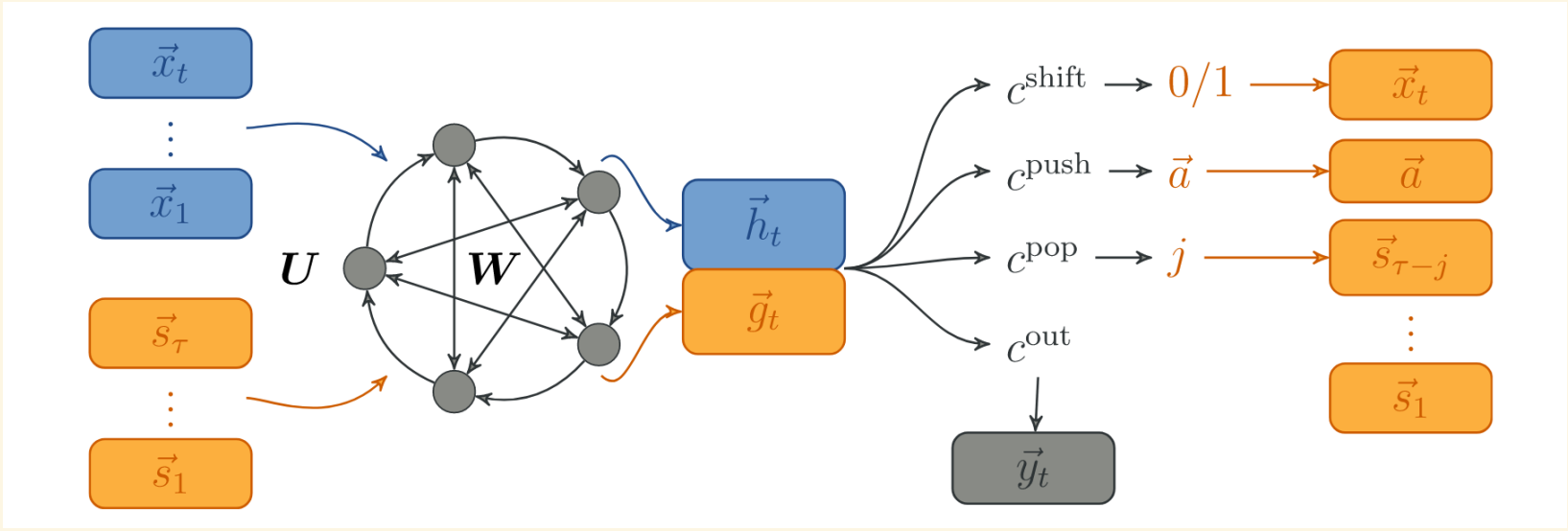


At **each** time-step t:

- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$
- c^{pop} decides what symbols (how many) to pop from the stack
- c^{push} decides which symbols to append onto the stack
- c^{out} computes the current output y_t

RESERVOIR STACK MACHINE

IDEA

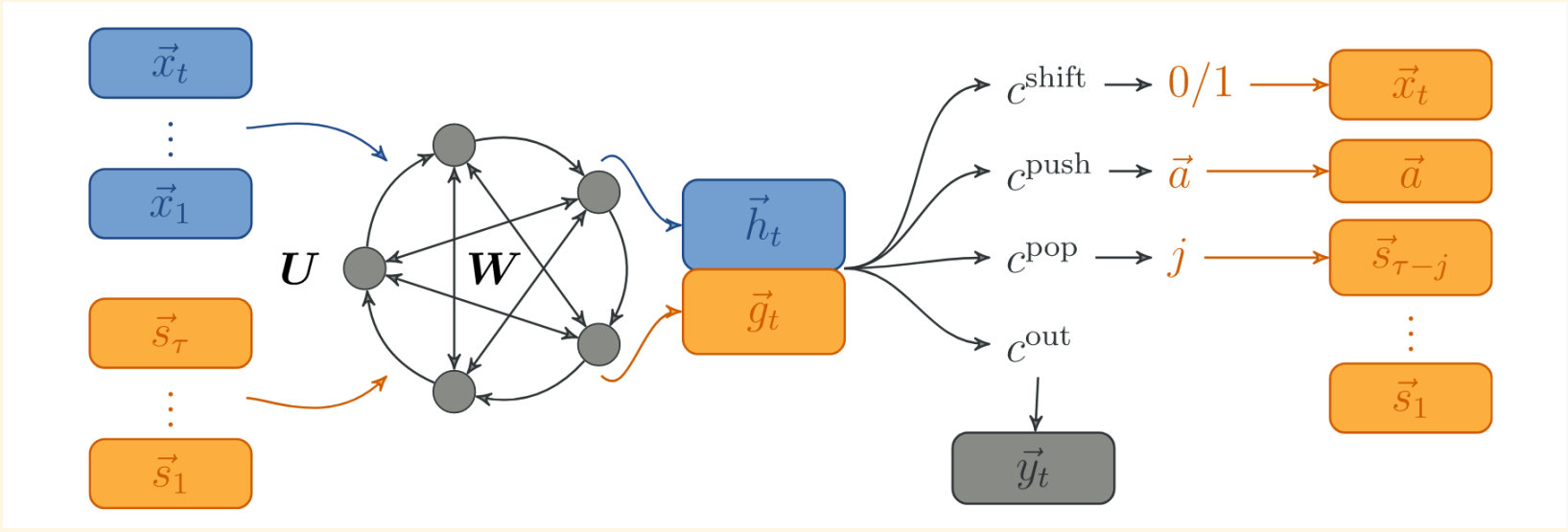


At **each** time-step t:

- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$
- c^{pop} decides what symbols (how many) to pop from the stack
- c^{push} decides which symbols to append onto the stack
- c^{out} computes the current output y_t
- c^{shift} decides whether to push the input x_t onto the stack

RESERVOIR STACK MACHINE

IDEA

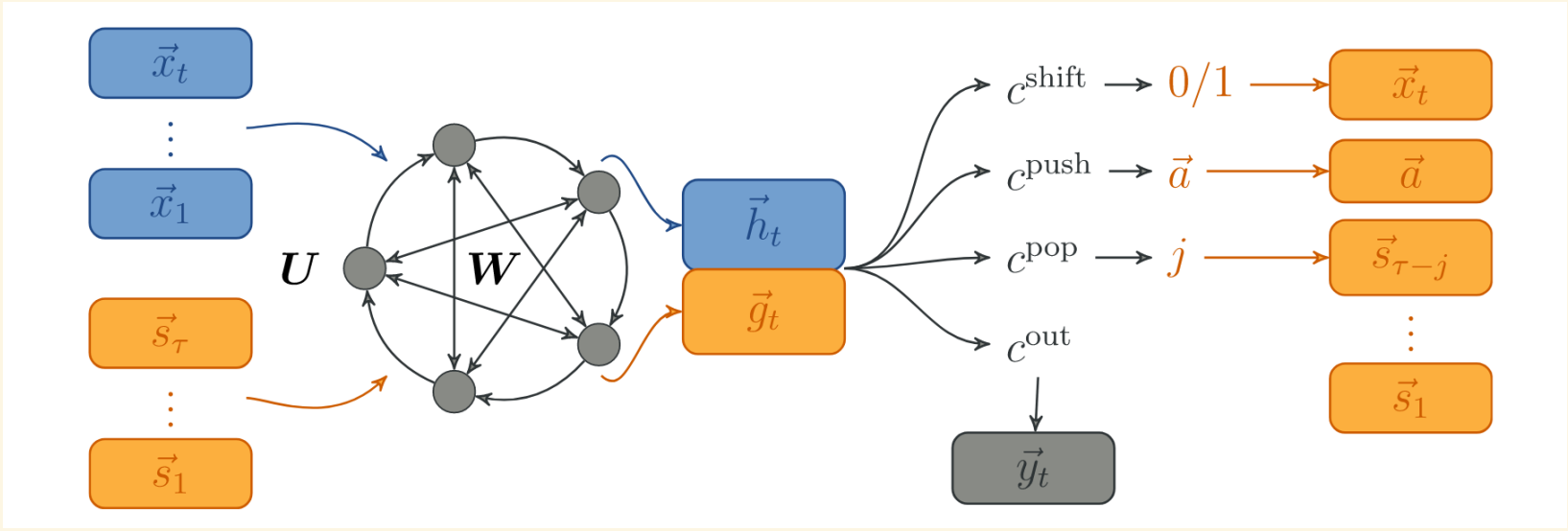


At **each** time-step t:

- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$
- c^{pop} decides what symbols (how many) to pop from the stack
- c^{push} decides which symbols to append onto the stack
- c^{out} computes the current output y_t
- c^{shift} decides whether to push the input x_t onto the stack
- Every time the stack changes, g_t gets updated

RESERVOIR STACK MACHINE

IDEA



At **each** time-step t:

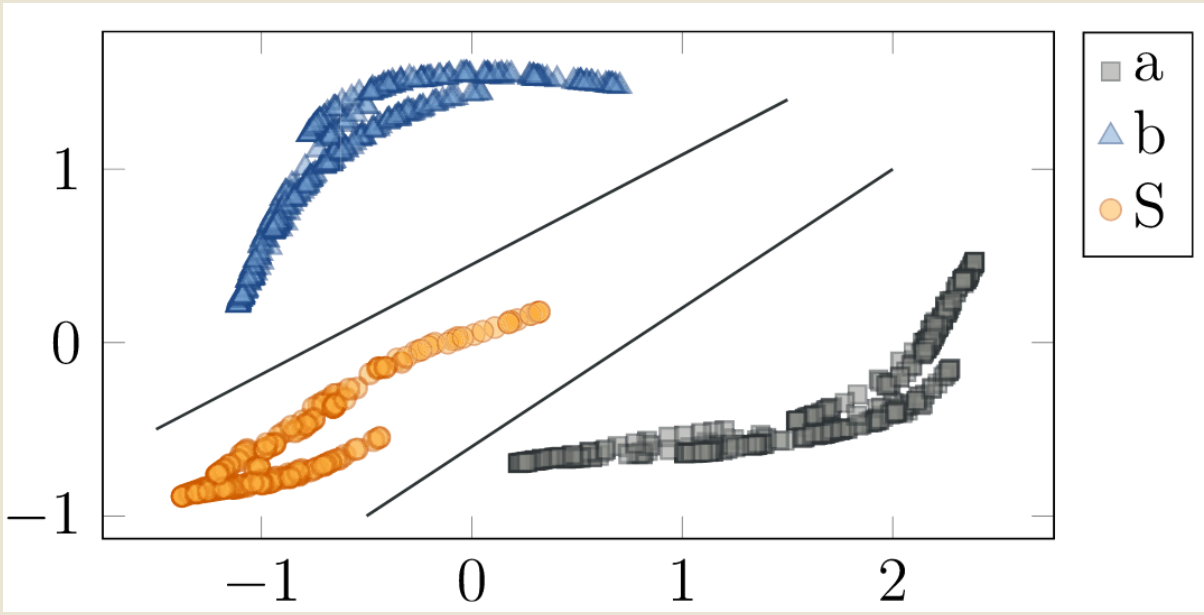
- **Encode** both $x_1, \cdots, x_t$ (*input*) as h_t and $s_1, \cdots, s_\tau$ (*stack content*) as g_t using the same reservoir $(\mathbf{U}, \mathbf{W}, \sigma)$
- c^{pop} decides what symbols (how many) to pop from the stack
- c^{push} decides which symbols to append onto the stack
- c^{out} computes the current output y_t
- c^{shift} decides whether to push the input x_t onto the stack
- Every time the stack changes, g_t gets updated
- Each of the above classifiers takes as input (h_t, g_t)

RESERVOIR STACK MACHINE

TRAINING

Training is largely similar to that of **RMMs**, but since it uses more than one classifier, more training signals are required. Nonetheless, given a *sufficiently rich*[1] reservoir, linear classifiers are enough.

Linear separability of RSM representations

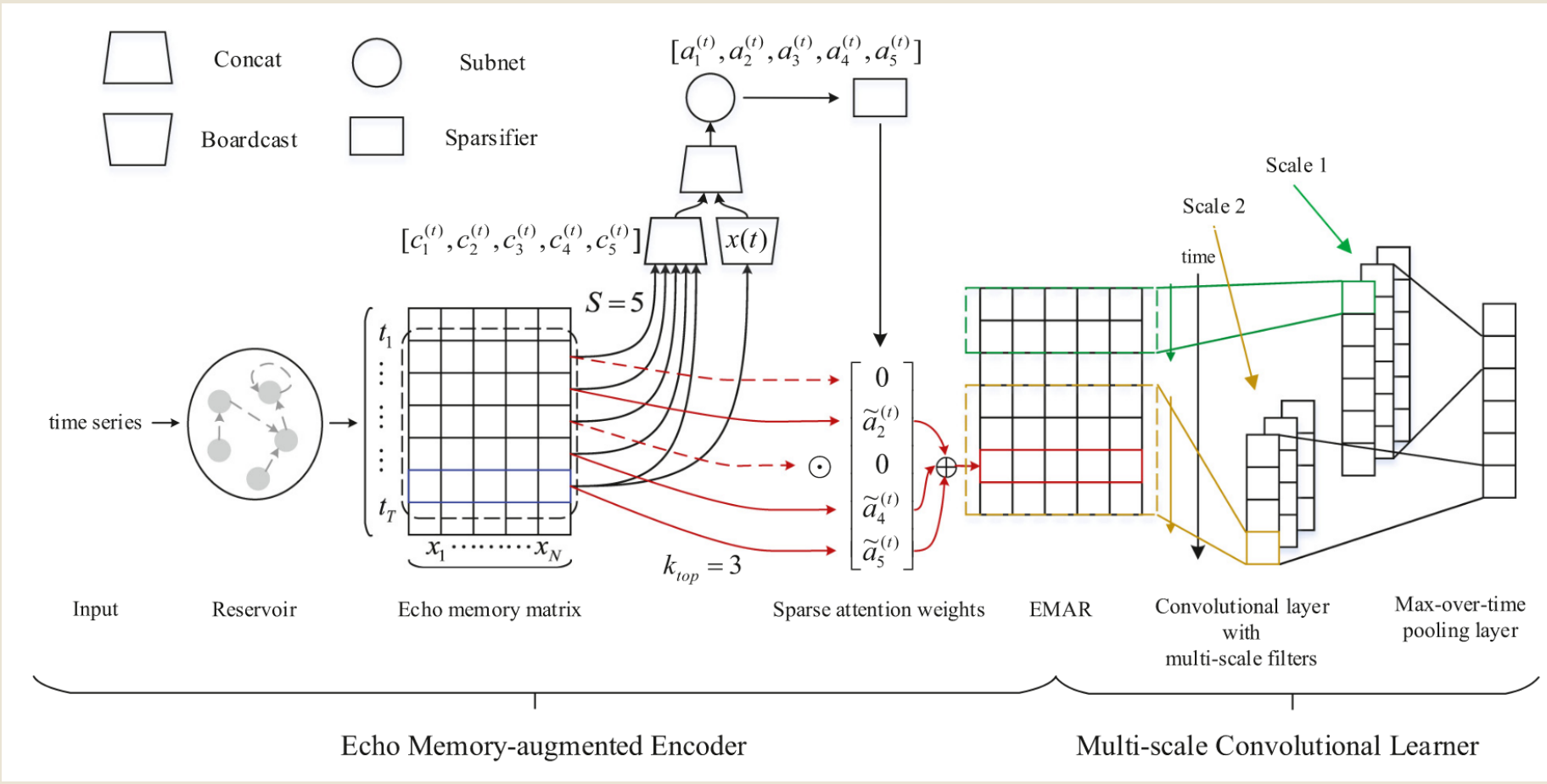


Contractive reservoirs are generally good separators[2]

1. The concept of richness here is related to the separability properties of the reservoir: Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)

END-TO-END MEMORY RESERVOIR MODELS

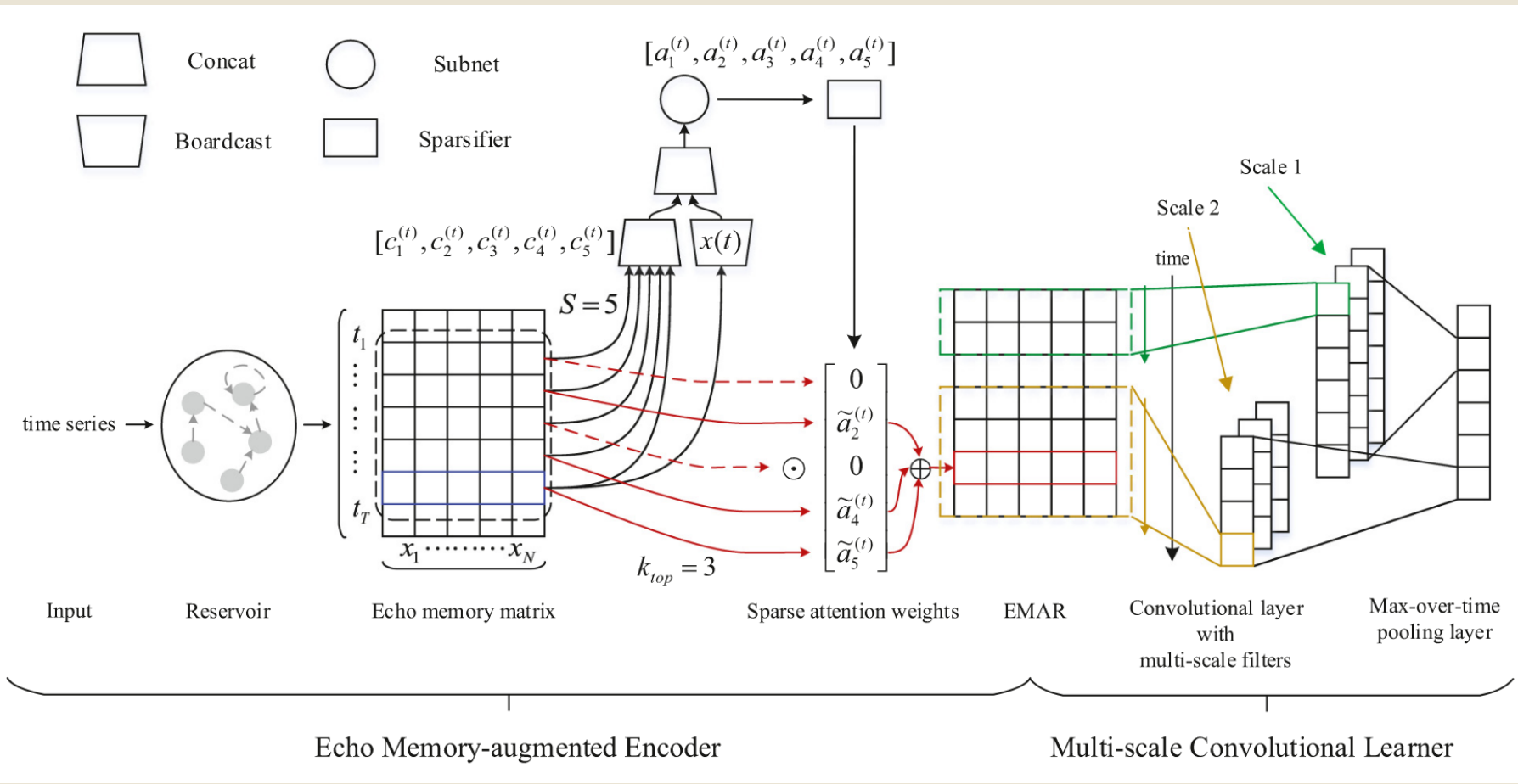
Echo memory-augmented network[1]



1. Qianli Ma, Zhenjing Zheng, Wanqing Zhuang, Enhuan Chen, Jia Wei, Jiabing Wang (2021)

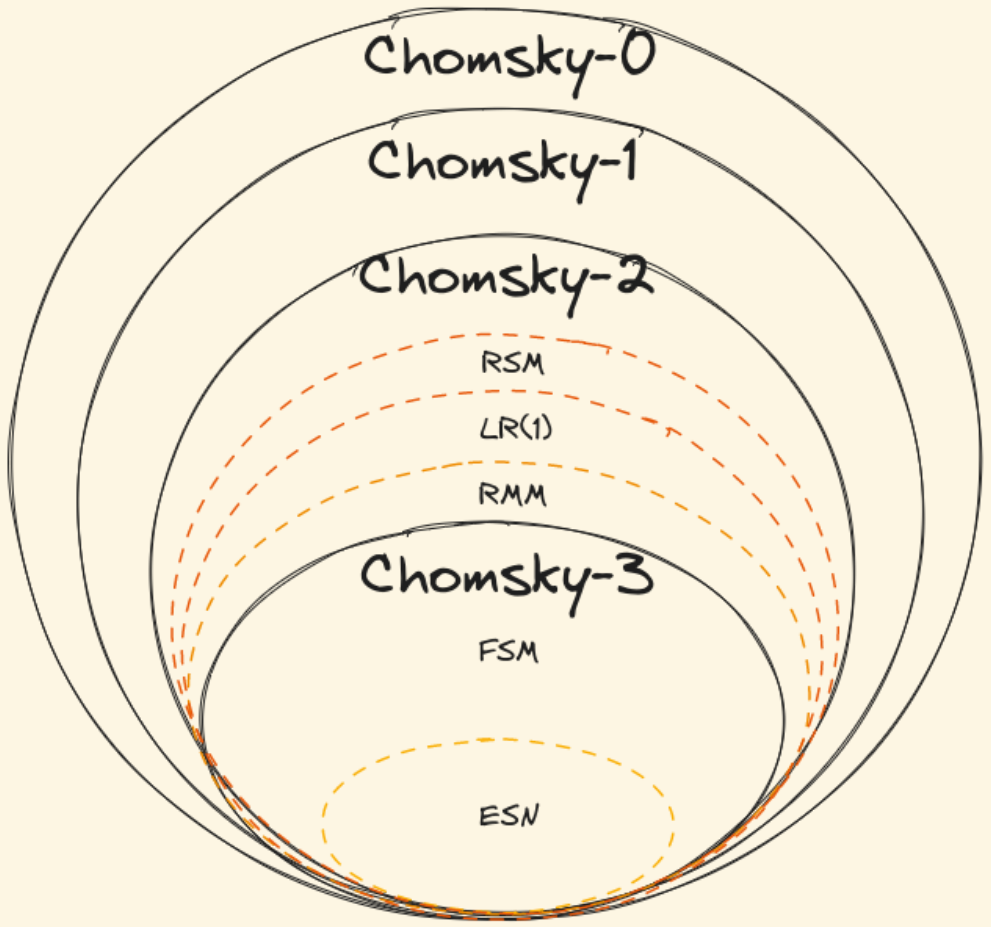
END-TO-END MEMORY RESERVOIR MODELS

Echo memory-augmented network



End-to-end memory systems have been shown to exhibit state of the art performance in time-series classification. This particular model uses an **ESN** to produce a memory matrix containing input embeddings. A deep convolutional head paired with an attention mechanism are used to extract features from the input and the memory matrix.

RESULTS



Main theoretical results

1. **ESNs** cannot simulate every **FSM** *(ESN ⊂ FSM)*
2. **RMMs** can simulate any **FSM** *(FSM ⊆ RMM)*
3. **RMMs** cannot simulate every **LR(1) automata** *(RMM ⊂ LR(1))*
4. **RSM** can simulate any **LR(1) automata** *(LR(1) ⊆ RSM)*

RESULTS

Given their **Markovianity**, contractive reservoirs will produce the same state for two (*long enough*) sequences sharing the same suffix

Example (ESN $\subset$ FSM)

Consider the input sequences b^T (b repeated T times) and ab^{T-1}.

A **Moore Machine** would easily distinguish between the two, while they would be completely indistinguishable for a **Contractive reservoir**.

The authors provide proof for the statement above $\longrightarrow$

RESULTS

Theorem[1] (ESN $\subset$ FSM)

Consider the Moore Machine $\mathcal{A}_{ab^*}$ that recognizes the language ab^*.

Let $(\mathbf{U}, \mathbf{W}, b, \sigma, h_0)$ be some Contractive reservoir with constant $C \in (0, 1)$.

Then it holds: For any continuous function $g : \mathbb{R}^n \to \mathbb{R}$ there exists some $T \in \mathbb{N}$ and some sequences $x_1, \cdots, x_T \in \{a, b\}^*$ such that $g(h_T) \neq z_T$, where h_T is the state of the reservoir for the one-hot encoded input sequence, and where z_T is the final output of the Moore machine $\mathcal{A}_{ab^*}$ for the same input.

1. Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)

RESULTS

Theorem[1] (ESN $\subset$ FSM)

Consider the Moore Machine $\mathcal{A}_{ab^*}$ that recognizes the language ab^*.

Let $(\mathbf{U}, \mathbf{W}, b, \sigma, h_0)$ be some Contractive reservoir with constant $C \in (0, 1)$.

Then it holds: For any continuous function $g : \mathbb{R}^n \to \mathbb{R}$ there exists some $T \in \mathbb{N}$ and some sequences $x_1, \cdots, x_T \in \{a, b\}^*$ such that $g(h_T) \neq z_T$, where h_T is the state of the reservoir for the one-hot encoded input sequence, and where z_T is the final output of the Moore machine $\mathcal{A}_{ab^*}$ for the same input.

Note: this relies on $C < 1$. Reservoirs at the **Edge of chaos** might be computationally more powerful, but it is yet unclear[2].

1. Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)
2. Claudio Gallicchio, Alessio Micheli, Luca Pedrelli (2018)

RESULTS

Theorem[1] (FSM $\subseteq$ RSM)

Let $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \rho, q_0)$ be a Moore Machine with $\Sigma \subset \mathbb{R}^m$, $Q = \{1, \cdots, L\}$ and $\Gamma = \{1, \cdots, K\}$ for some $m, L, K \in \mathbb{N}$.

Further let $(\mathbf{U}, \mathbf{W}, b, \sigma, h_0)$ be a reservoir *sufficiently rich*[2].

Then there exist functions

$$c : \mathbb{R}^n \to \{0, \cdots, L\},$$
$$g : \mathbb{R}^n \to \{1, \cdots, K\} \text{ such that}$$

$\mathcal{M}_{\mathcal{A}} = (\mathbf{U}, \mathbf{W}, b, \sigma, h_0, \{0, \cdots, L\}, c, g)$ is a RMM that simulates $\mathcal{A}$.

1. Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)
2. Here the concept of "richness" is related to the memory capacity of the reservoir: Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)

RESULTS

Theorem[1](RMM $\subset$ LR(1))

Let $\mathcal{A}_{\text{palin}}$ be an LR(1) automata that recognizes the language of palindromes over $\{a, b\}$ with \$ in the middle.

Let $\Sigma = \{a, b, \$\}$ be encodings of the respective symbols.

Let $\mathcal{M}$ be a RMM with a uniformly continuous output function g, i.e., $\forall \epsilon, \exists$ some $\tilde{\delta}_\epsilon$ such that for any two $h, h' \in \mathbb{R}^m$ with $||h - h'|| < \tilde{\delta}_\epsilon$ it holds $|g(h) - g(h')| < \epsilon$.

Then, $\mathcal{M}$ does not simulate $\mathcal{A}_{\text{palin}}$

1. Benjamin Paaßen, Alexander Schulz, Barbara Hammer (2022)

RESULTS

LR(1) $\subseteq$ RSM

As already anticipated, the extension of **ESNs** with a **stack** has been proven sufficient to simulate any **LR(1)-automata**[1]

1. Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)

FUTURE RESEARCH

> Stacks and Turing machines

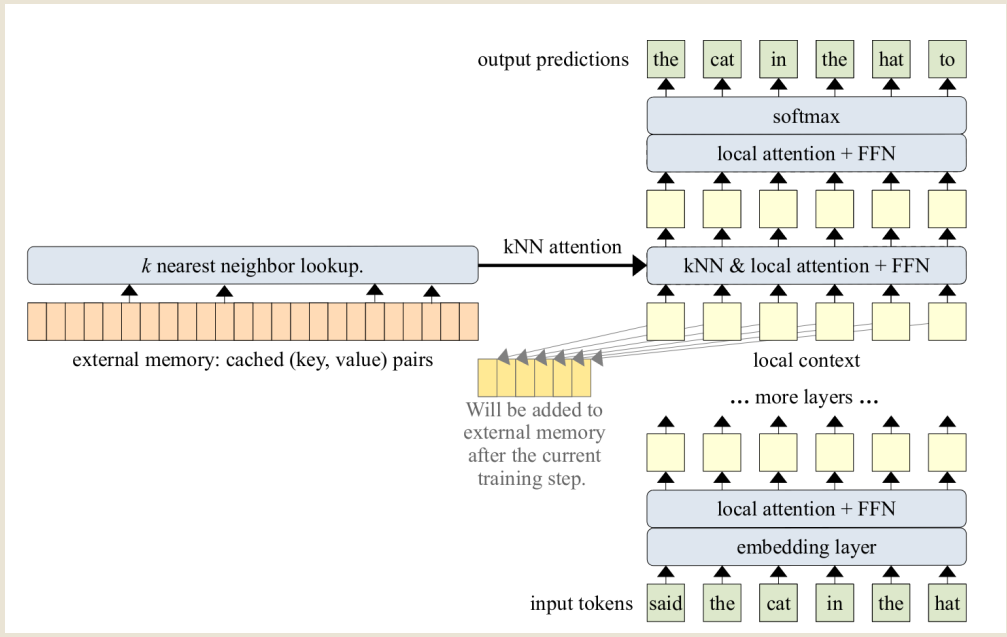
As noted in[1], research on using reservoir models with more than one stack to match the computational power of **Turing machines** is still needed, along with the discovery of automatic ways of producing teaching signals for *non* end-to-end systems such as **RMM**s and **RSM**s.

1. Benjamin Paaßen, Alexander Schulz, Barbara Hammer (2022)

FUTURE RESEARCH

Recent work on **Reservoir transformers**[1] might prove an intriguing direction of research. Some experiments on extending transformers with working memories have already been carried out and are an interesting frontier of research in the NLP domain.

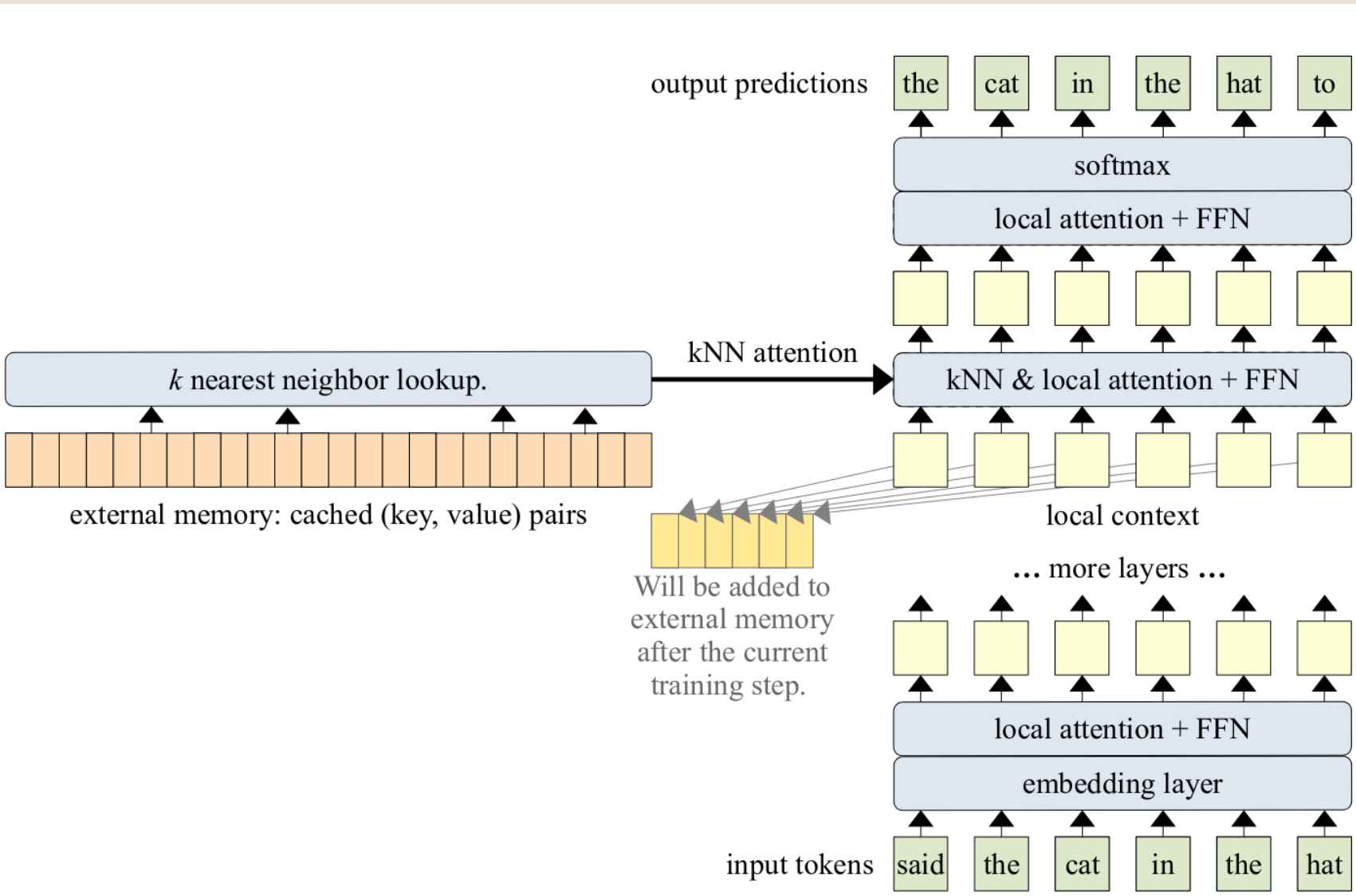
Memorizing transformer



1. Sheng Shen, Alexei Baevski, Ari S. Morcos, Kurt Keutzer, Michael Auli, Douwe Kiela (2021)

FUTURE RESEARCH

Memorizing transformer[1]



1. Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, Christian Szegedy (2022)

CONCLUSIONS

Summary

- Memory augmented reservoir methods have been shown to be a fast and cheap alternative to end-to-end DNCs.
- **RMM**s and **RSM**s can (theoretically) simulate any finite state automata and LR(1) automata, respectively.
- Extensions of current models (e.g. with more than one stack) are possible and of theoretical interest.
- **State of the art** results can be reached with end-to-end reservoir memory-augmented methods.

CONCLUSIONS

Limitations

- **RMM**s and **RSM**s both need additional teaching signals in order to train the external memory controller.
- Their practical use remain uncertain because of their need of additional training data and because they lack content-based addressing.
- Some benchmark tasks of NTMs are still out of reach of non end-to-end models.

REFERENCES

- Aaron Voelker, Ivana Kajić, Chris Eliasmith (2019)
- Alex Graves, et al. (2016)
- Alex Graves, Greg Wayne, Ivo Danihelka (2014)
- Benjamin Paassen, Alexander Schulz (2020)
- Benjamin Paaßen, Alexander Schulz, Terrence C. Stewart, Barbara Hammer (2022)
- Benjamin Paaßen, Alexander Schulz, Barbara Hammer (2022)
- Claudio Gallicchio, Alessio Micheli (2011)
- Claudio Gallicchio, Alessio Micheli, Luca Pedrelli (2018)
- Igor Farkaš, Radomír Bosák, Peter Gergeľ (2016)
- Qianli Ma, Zhenjing Zheng, Wanqing Zhuang, Enhuan Chen, Jia Wei, Jiabing Wang (2021)
- Sheng Shen, Alexei Baevski, Ari S. Morcos, Kurt Keutzer, Michael Auli, Douwe Kiela (2021)
- Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, Christian Szegedy (2022)

THE END

THANK YOU FOR YOUR TIME