nature machine intelligence

Review article

https://doi.org/10.1038/s42256-025-01034-6

Back to recurrent processing at the crossroad of transformers and state-space models

Received: 3 July 2024

Accepted: 3 April 2025

Published online: 15 May 2025

Check for updates

Matteo Tiezzi[1], Michele Casoni[2], Alessandro Betti[3], Tommaso Guidi[2], Marco Gori[2] & Stefano Melacci[2]

It is a longstanding challenge for the machine learning community to develop models that are capable of processing and learning from long sequences of data. The exceptional results of transformer-based approaches, such as large language models, promote the idea of parallel attention as the key to succeed in such a challenge, temporarily obscuring the role of classic sequential processing of recurrent models. However, in the past few years, a new generation of neural models has emerged, combining transformers and recurrent networks motivated by concerns over the quadratic complexity of self-attention. Meanwhile, (deep) state-space models have also emerged as robust approaches to function approximation over time, thus opening a new perspective in learning from sequential data. Here we provide an overview of these trends unified under the umbrella of recurrent models, and discuss their likely crucial impact in the development of future architectures for large generative models.

The outstanding results achieved by large language models (LLMs)[1,2] and by their even more recent multi-modal variants[3], rely on attention-based neural architectures with several analogies to the encoder or decoder from the originally proposed transformers[4]. Sequential data are processed by attention models, usually handling input tokens in parallel with multi-headed self-attention blocks. The 'attention is all you need' message somewhat hindered further developments on the already-established sequential processing of recurrent models[5–7]. However, the need to reduce the computational burden and to improve LLMs in both training and inference has been renewing the interest in recurrent architectures for processing sequential data[8,9]. Moreover, seminal approaches in continuous-time recurrent neural networks (RNNs)[10,11] opened up an alternative path in handling long-range sequences with the so-called (deep) state-space models (SSMs)[12]. This path inspired a plethora of works aimed at injecting pure SSMs into deep architectures[13,14], contributing to the proposal of a new class of powerful recurrent units, linear recurrent units (LRUs)[8] and other variants[15,16].

These novel routes, paired with further research in the optimization schemes of RNNs[17], yielded a new chapter in the narrative of sequence processing ('Key concepts of recurrent models' section), rooted on the resurgence of recurrence, which is the main topic of this paper ('Resurgence of recurrence' section). We provide a review of the main advancements in the field, connecting them with earlier research activities on RNN models, thus allowing a wide range of audience to quickly get into this topic. See Fig. 1 for an overview. We finally provide an analysis on the limitations of the described families of models, current challenges and future directions ('Future directions and challenges' section).

Key concepts of recurrent models

In many real-world applications, such as those related to natural language, video processing, time-series forecasting and others, leveraging the order in which patterns are provided to the machine is crucial for discovering cues in the task at-hand[18]. Let us consider a sequence of patterns $(u_t)_{t=1}^{L} := (u_1, \ldots, u_L)$, where L is an integer that represents the sequence length and $u_t \in \mathbb{R}^{d_{\text{in}}}$ is the tth element of the sequence, and $d_{\text{in}} \geq 1$ the number of features of each element. Recurrent models process one pattern at a time, following the sequence order,

[1]IIT, Istituto Italiano di Tecnologia, Genova, Italy. [2]DIISM, University of Siena, Siena, Italy. [3]IMT, Scuola Alti Studi, Lucca, Italy. e-mail: stefano.melacci@unisi.it

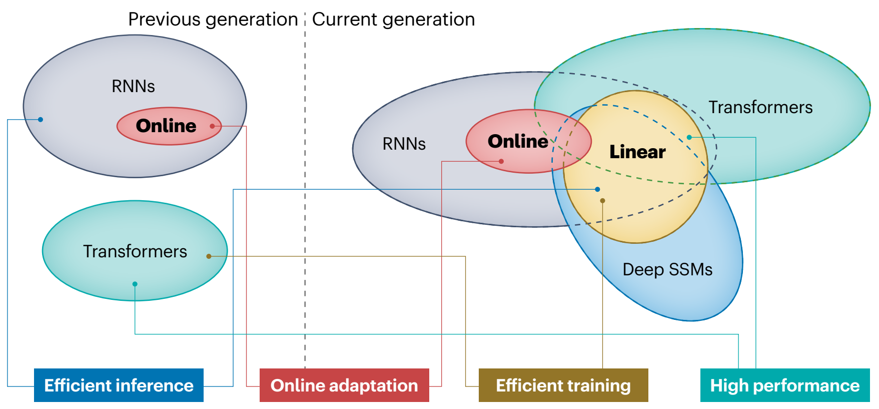

Fig. 1 | Previous and current generation of neural models for sequence processing. The current generation of neural models (roughly since 2021) features a strong intersection between popular RNNs and transformers. These models are now sharing the same notions of 'recurrence', with a strong emphasis on linearity and state-space modelling (nonlinearity is introduced by other additional components). The 'Online' oval is a way to remark that, in the literature of the previous generation, online learning is mostly studied in the case of RNNs. Similarly, the 'Linear' oval indicates that linear recurrent models, not frequent at all in the previous generation, are very common in the current one.

progressively (sequentially) encoding the information into d_s units whose output values model a state $x_t \in \mathbb{R}^{d_s}$. While in standard feed-forward networks hidden neurons develop internal representations of the current input patterns, in recurrent models the state encodes the temporal properties of the input data that was sequentially provided so far. The state is further processed for generating an output signal $y_t \in \mathbb{R}^{d_{\text{out}}}$, $d_{\text{out}} \geq 1$. The Elman's architecture[18] was one of the pillars to the foundations of classic RNNs, and it is based on the following equations

$$\begin{aligned} x_t &= f(x_{t-1}, u_t) = \sigma(Ax_{t-1} + Bu_t), && \text{State-update function} \\ y_t &= g(x_t, u_t) \;\;\; = \sigma_{\text{out}}(Cx_t + Du_t), && \text{Output function} \end{aligned} \tag{1}$$

where we considered a state matrix $A \in \mathbb{R}^{d_s \times d_s}$, an input matrix $B \in \mathbb{R}^{d_s \times d_{\text{in}}}$, an output matrix $C \in \mathbb{R}^{d_{\text{out}} \times d_s}$ and a feedthrough matrix $D \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Here, $D \neq 0$ introduces a skip connection that was not included in the original Elman network[18]. We denote with σ and σ_{out} pointwise nonlinearities on the state and output computations, respectively, often selected to be tanh or sigmoid activations. If σ is the identity function, then we say the RNN model is linear. RNNs are able to simulate universal Turing machines[19], and have been proven to be universal approximators, that is, they are able to approximate any open dynamical system with an arbitrary accuracy[20]. Recurrent models can be considered a possible discretization of continuous-time dynamical systems[21], equipped with nonlinear state-dependencies. This view connects recurrent networks to the generic notion of an SSM, a concept that is widely used in maths and engineering[22], especially in control theory, to represent dynamical systems across a variety of types (linear, nonlinear, time-varying and so on). SSMs define state evolution through differential equations that, once discretized, yield difference equations involving the previous state and the current inputs, that is, the feedback mechanisms of recurrent models. For instance, continuous-time RNNs are modelled as time-invariant systems, whose dynamics can be easily related to their discrete counterparts of equation (1)

$$\begin{aligned} \dot{x}(t) &= -x(t) + \sigma(Ax(t) + Bu(t)), \\ y(t) &= \sigma_{\text{out}}(Cx(t) + Du(t)). \end{aligned} \tag{2}$$

There are several books, surveys and reviews in which the reader can find further details and properties of RNN-based modes[5,23,24]. In the 'Other trends' section, we further explore different approaches based on continuous-time dynamical systems, including neural ordinary differential equations (ODEs)[25].

Learning

Let us introduce the instantaneous loss function $\ell(y_t, \hat{y}_t)$, which quantifies to what degree the predicted output y_t matches some supervised/target output $\hat{y}_t$ (supervision might not be present for all ts). Let us collect the model parameters in $\theta := \{A, B, C, D\}$, where the loss ℓ depends on θ due to the way y_t is computed (equation (1)). Backpropagation through time[26] is the de facto standard algorithm to learn θ by gradient descent, where the chain rule is applied over the unfolded network to compute gradients[27]. Given a sequence and the empirical risk $\mathcal{L}(\theta) = \frac{1}{L}\sum_{t=1}^{L} \ell(y_t, \hat{y}_t)$, it can be shown that the gradient with respect to θ is a sum of products of partial gradients, $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{L}\sum_{t=1}^{L} \frac{\partial \ell(y_t, \hat{y}_t)}{\partial y_t}\frac{\partial y_t}{\partial x_t}\sum_{j=1}^{t}(\prod_{s=j}^{t}\frac{\partial x_s}{\partial x_{s-1}})\frac{\partial x_{j-1}}{\partial \theta}$. Here, a central role is played by the term $\prod_{s=j}^{t} \partial x_s / \partial x_{s-1}$, which transports the error back in time from step t to step j. We can rewrite this term in the form of a product of Jacobi matrices[28]

$$\prod_{s=j}^{t} \frac{\partial x_s}{\partial x_{s-1}} = \prod_{s=j}^{t} A^T \operatorname{diag}\left(\sigma'(x_{s-1})\right), \tag{3}$$

where T is the transpose operator, diag(·) converts a vector into a diagonal matrix, and σ' computes the derivative of the activation function σ in equation (1) in an element-wise fashion (recall that we assumed σ to be a pointwise nonlinearity). Depending on the properties of A and σ, the products in equation (3) have a key role in the learning dynamics.

Limits

Unless the partial terms $\partial x_s / \partial x_{s-1}$ are close to 1, the product in equation (3) could explode or vanish[29]. In details, in the simplified case of a linear model (that is, setting σ to the identity function) a sufficient condition for long-term components to vanish is that of having the largest eigenvalue of the state matrix A smaller than 1, that is, $\lambda < 1$. Conversely, $\lambda > 1$ is a necessary condition for gradients to explode, causing older states to have little contribution. See ref. 28 for further details. When gradients undergo vanishing during their backwards propagation through the network, the critical credit assignment of backpropagation is compromised[30]. In fact, information regarding minor state changes in the distant past loses its ability to influence future states. Conversely, when gradients explode, optimization algorithms encounter substantial difficulties, leading to an unstable behaviour or to critical numerical issues. To make RNNs better suited to handle longer sequences, popular variants were proposed and widely used in the scientific literature and several real-world applications, such as long short-term memories (LSTMs)[31], gated recurrent units[32] and others[5]. One of the main features of these architectures consists of the introduction of gating functions, specialized in controlling how the internal representation is updated depending on the actual input and the previous hidden state. While the definition of gating in recent literature depends on the specifics of the considered model, the term gate commonly refers to a learnable function that regulates the contribution of another component by element-wise multiplication. Typically the gate yields values in $[0, 1]$ (formally in $(0, 1)$ in the case of sigmoids), ranging from a closed gate to a fully open gate, that is, zeroing the modulated signal or leaving it untouched, respectively. Sigmoidal-like gates are sometimes replaced by ReLU (rectified linear unit), SiLU (sigmoid linear unit) or Swish-based activations, ranging in $[-\epsilon, \infty)$ (for $\epsilon \geq 0$, a small-positive or zero constant), which can also amplify the signal[15,33–35]. Moreover, gating functions usually depend either on the current input only, or on both the current input and the hidden state (or input) at the previous time instant. This is different from, for example, softmax attention, which depends on the whole input sequence (as we discuss in the following). Going beyond the concept of gating, there exist other solutions to mitigate stability issues. Some works force the hidden-to-hidden weight matrix to be either orthogonal or unitary (that is, belonging to the orthogonal group, often referred to as unitary and orthogonal RNNs; see also refs. 36,37 and related surveys[24]). This constraint ensures that the eigenvalues have a unit norm, which stabilizes the dynamics of the network. While this approach aids in preserving long-term memory, it also limits the expressive power of the model.

Recurrence versus attention

Given an input sequence $(u_t)_{t=1}^{L}$, computing the corresponding output $(y_t)_{t=1}^{L}$ by sequentially evaluating equation (1) is $O(L)$. This interesting property, paired with efficient gradient propagation schemes[38] and outstanding experimental results, favoured the growth of popularity of RNNs (different variants) together with deep learning, in language, vision and beyond[39]. However, since 2017, RNNs have been shadowed by transformers[4], which promoted a stateless architecture, fully based on self-attention, 'dispensing with recurrence and convolutions entirely'. Indeed, architectures involving recurrence are based on feedback connections that allow them to develop a form of memory (that is, the state), updated over time without having access to the entire input sequence processed so far. In principle, such an evolving state encodes the temporal features of the whole input sequence, up to the current time instant. Conversely, transformers are stateless, in the sense that they explicitly require the entire sequence (or a temporal window) to produce an output. Of course, context vectors in transformers somewhat resemble the notion of state, even if not progressively updated with a feedback mechanism. The relation between attention and the hidden state in recurrent models has been explored in recent works[40,41]. The essence of the self-attention mechanism consists in producing a representation of the input sequence that is based on comparing each u_t with itself and with all the other elements of the sequence. We consider here a simple attention head, with no further projections, to better connect it to the RNN case of equation (1). In particular, the attention procedure exploits a triple of query–key–value for each u_t, namely, query q_t, key k_t and value v_t, indicated with (q_t, k_t, v_t). Query, key and value are three vectors computed in function of u_t, by means of three learnable maps, usually linear, that is, $q_t = q(u_t, W_q)$, $k_t = k(u_t, W_k)$, $v_t = q(u_t, W_v)$, with learnable parameters W_q, W_k, W_v. To compute the output y_t, the query is compared with the L keys associated with the input sequence, owing to a customizable similarity function that yields L similarity scores. Finally, y_t is the outcome of averaging the L values associated with the elements of the input sequence, weighed by the similarity scores. Formally

$$y_t = \sum_{i=1}^{L} \frac{\mathrm{sim}(q_t, k_i)}{\sum_{j=1}^{L} \mathrm{sim}(q_t, k_j)} v_i, \qquad \begin{aligned} q_z &= q(u_z, W_q),\\ k_z &= k(u_z, W_k),\\ v_z &= v(u_z, W_v), \end{aligned} \tag{4}$$

being $\mathrm{sim}(\cdot,\cdot)$ a similarity function. For example, the softmax version of attention exploits $\mathrm{sim}(a, b) = \exp(a^T b/\sqrt{|a|})$ (where a and b are generic mono-dimensional arrays, assumed to be column vectors, and T is the transpose operator). Notice that the dependence on some previous state x_{t-1}, as in equation (1), is not present anymore, and the cost of computing the output sequence $(y_t)_{t=1}^{L}$ is now quadratic, $O(L^2)$. However, despite the larger complexity, the constraint on sequential computations of equation (1) disappears in transformers, and each y_t can be computed in parallel to the other ones, which became an attractive property for graphics processing unit-oriented implementations.

Resurgence of recurrence

When dealing with long sequences, (1) the quadratic complexity of self-attention in transformers becomes a serious concern, driving the need for novel paths to achieve efficient processing[42]. Moreover, (2) the need for more accurately preserving extremely long-term dependencies on sequences favoured the proposal of new models. To meet these two requirements, new solutions were explored by focusing again on recurrent formulations, leading to the current resurgence of recurrence[8,43]. We identify several properties that turn out to be (partially) shared by several recent works that will be discussed in the following, and for which we provide an overview in Table 1.

Transformers

To reduce the quadratic complexity of stateless self-attention, many solutions based on stateful recurrent computations have been introduced, revising the originally proposed attention procedure[9,35,43]. We subdivide the plethora of scientific papers that follow this research direction as a function of their most evident property.

Linear transformers. The most evident architectural step towards reintroducing recurrence is the one of linear transformers[43], implementing a self-attention procedure with linear complexity (with respect to L). This is achieved by redefining the similarity function of equation (4) as a kernel $\mathcal{K} : \mathbb{R}^{d_k} \times \mathbb{R}^{d_k} \mapsto \mathbb{R}^{+}$, that is, $\mathrm{sim}(q, k) := \mathcal{K}(a, b) = \phi(a)^T \phi(b)$, where $\phi : \mathbb{R}^{d_k} \mapsto \mathbb{R}^{d_{\mathrm{ker}}}$ is a nonlinear feature map. We get

$$y_t = \sum_{i=1}^{L} \frac{v_i \phi(q_t)^T \phi(k_i)}{\sum_{j=1}^{L} \phi(q_t)^T \phi(k_j)} = \frac{S_L \phi(q_t)}{\phi(q_t)^T z_L}, \tag{5}$$

where matrix $S_L := \sum_{j=1}^{L} v_j \otimes \phi(k_j)$, with $\otimes$ being the outer product between vectors, and $z_L := \sum_{j=1}^{L} \phi(k_j)$. The linear complexity is evident when we consider that S_L and z_L can be computed once, and reused to compute all the L outputs y_ts. Moreover, if transformers are exploited in a causal setting (such as in autoregressive problems), where, at time t, only elements whose index is $\leq t$ are used, then equation (5) can be rewritten in a stateful form, with the internal representation being recursively computed as the sequence is processed, in contrast to what happens in stateless models. Indeed, the structure of equation (1) is recovered, where the internal state x_t is composed of the state matrix S_t and the normalization vector z_t, that is, $x_t := (S_t, z_t)$. The state is iteratively updated with what is referred to as additive interaction

$$\begin{aligned} \begin{pmatrix} S_t \\ z_t \end{pmatrix} \equiv x_t &= f(x_{t-1}, u_t) = \begin{pmatrix} S_{t-1} + v_t \otimes \phi(k_t) \\ z_{t-1} + \phi(k_t) \end{pmatrix}, && \text{State-update function} \\ y_t &= g(x_t, u_t) \quad = \frac{S_t \phi(q_t)}{\phi(q_t)^T z_t}, && \text{Output function} \end{aligned} \tag{6}$$

for some initial S_0 and z_0 (for instance, null). As emphasized in Fig. 2, this type of transformer challenges the traditional distinction from RNNs. If the cost to compute ϕ is $O(d_{\mathrm{ker}})$, then the overall run-time complexity of a linear transformer for a sequence of length L is $O(L d_{\mathrm{ker}} d_{\mathrm{k}})$. The role of the kernel function ϕ has been investigated in

Table 1 | Previous generation versus current generation architectures for long sequence processing

		Model	Recurrent	Linear	Gating	Diagonal
Previous generation		Recurrent net[31]	✓		PC	
		Transformer[4]				
Current generation	Transformer	Linear transformer[43]	✓	✓		✓
		RWKV[9,54]	✓	✓	PC	✓
		RetNet[35]	✓	✓	C	✓
		Transformer-XL[56]	✓			
	SSMs	S4[12]	✓	✓		
		H3[64], DSS[61], S4D[60]	✓	✓		✓
		Mamba[15], Griffin[16]	✓	✓	C	✓
	RNNs	xLSTM[75]	✓	✓	PC	✓
		HRGN[76]	✓	✓	C	✓
		Neural oscillators[83]	✓			

The table follows the organization of Fig. 1. We showcase how some of the features discussed in this paper are distributed over recent approaches, comparing a few representatives of recent transformers ('Transformers' section), modern SSMs ('State-space models' section), and other recent instances of recurrent networks ('Other trends' section). We consider: recurrent formulations, the use of linear recurrent models, the presence of gating mechanisms and the presence of diagonal transition matrices. Following the definition of gating we provided in 'Key concepts of recurrent models', we differentiate among gating functions that exploit data at the current time instant only (C) and gating based also on the state (or input) at the previous time instant (P). The tick marks indicate what features (columns) are present in the corresponding models (rows).

several subsequent works[44,45]. Original linear transformers exploit $\phi_{\text{elu}}(x) = \text{elu}(x) + 1$, where elu(·) denotes the exponential linear unit, a kernel that preserves the dimension of the input key vector ($d_{\text{ker}} = d_{\text{k}}$), yielding $O(Ld_{\text{k}}^{2})$.

Improving linear transformers. The computational advantage of linear attention is provided by the finite dimension of the feature map. Softmax attention, for instance, could be expressed in the form of equation (5), but this would require an infinite-dimensional feature map[41,43]. The computational advantages come at the price of a reduced expressivity and it turns out that linear transformers underperform vanilla self-attention, the main cause being attributed to the choice of the kernel function[44,45]. Several works were proposed to overcome this issue, leveraging random feature maps to achieve unbiased estimations of shift-invariant kernels, such as the Gaussian one, to approximate the effect of softmax in vanilla attention (see, for example, random feature attention[44]). The TransNormer model[46] identifies the origin of the degradation in the performance of kernel-based linear transformers in unbounded gradients and sparse distribution of attention scores. The authors of TransNormer proposed to exploit a linear kernel and remove the z_t normalization factor in equation (6) (right), introducing layer normalization on the attention scores. This intuition was already pointed out by Schlag et al.[45] (DeltaNet) and succeeding works that emphasized the formal relationships between kernel-based linear transformers and fast weight programmers (FWPs)[47]. FWPs are built on the intuition of making the model weights input dependent, with a two-network system where a slow net with slow-varying weights continually reprograms a fast net's weights, making them dependent on the spatio-temporal context of a given input stream. The kernel-based causal perspective of linear attention, equation (6), is an FWP with an additional recurrent normalization factor z_t. Still inspired by connections to FWP, two other ingredients that were studied to improve recurrence are gating functions and the introduction of decay terms, to help modulate the impact of input information. For instance, recent variants of linear transformers such as RetNet[35] and TransNormerLLM[48], among several other improvements, introduce a forgetting mechanism via a fixed decay factor γ in the S_{t-1} term of equation (6) (left), to control the capability of each token to pool information from its surrounding tokens based on encoding-related priors. GateLoop[49] and gated linear attention[50] explore a data-dependent gating mechanism for linear transformers. The FLASH model[34] combines a gated linear unit[51] and attention in a unified softmax-free single-head layer, referred to as gated attention unit (GAU), which achieves multi-head transformer performances without quality loss. MEGA[52] integrates a variant of GAU with a multidimensional exponential moving average that captures local dependencies that exponentially decay over time.

Alternative factorization of attention. Kernel-based formulations exploited to linearize self-attention are effective in reducing the computational cost with respect to the sequence length, but they still result in quadratic complexity with respect to the feature dimension, $O(Ld_{\text{k}}^{2})$, which is unfriendly to large model sizes[53]. A recent alternative emerged from a low-rank factorization of the sim() function in equation (4), that is, $\text{sim}\,(a, b) = \sigma_q(a) \odot \psi(b)$. Attention-free transformers[53] implement σ_q as an element-wise nonlinear activation function (for example, sigmoid) and $\psi(a) = \text{e}^{a}$, and perform the following operations

$$y_t = \sigma_q(q_t) \odot \frac{\sum_{i=1}^{L} \text{e}^{k_i + p_{t,i}} \odot v_i}{\sum_{i=1}^{L} \text{e}^{k_i + p_{t,i}}} \qquad (7)$$

where $\odot$ is the Hadamard product, $p_{t,i}$ denotes a learnable position bias involving token i at time t, and the division is intended to operate in an element-wise manner. This approach eliminates the need for dot product in self-attention by rearranging the order in which computations are performed, and it can still be written in additive recurrent form when dealing with the causal setting, as previously discussed in the case of equation (6). Inspired by attention-free transformers, the receptance weighted key value (RWKV) model[9] replaces the position embeddings of attention-free transformers with an exponential decay, allowing to exploit the model recurrently both for training and inference, leading to significant gains in efficiency. Subsequently, the authors proposed two model evolutions (RWKV-5 and -6)[54]: the former introduces an additional gating mechanism and matrix-valued states and the latter further applies data-dependent dynamic recurrence.

Structuring recurrence. Amid the multiple advantages brought by the vanilla self-attention mechanism, early attempts to tackle language modelling with transformers were plagued by the inability (1) to model intra-sequential relations due to the static order dependencies available in standard positional encodings (that is, the absence of explicit temporal information) and (2) to propagate inter-sequence

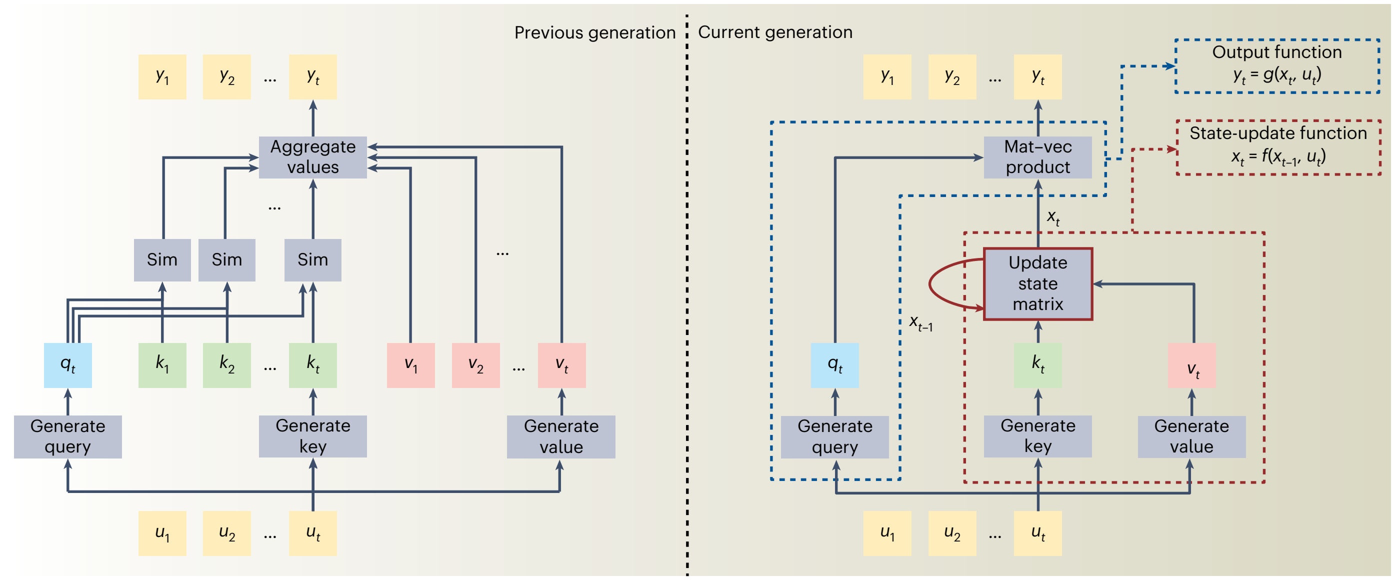

Fig. 2 | Evolution of the output computation in transformers with causal attention. Transformers, when considering causal attention (inference), compute y_t given a new input token u_t. Left: previous generation, with attention involving all the previous keys and the currently generated one, $k_1, k_2, \ldots, k_t$, and aggregation involving all the previous/current values $v_1, v_2, \ldots, v_t$. Right: linear transformers, depending only on data at time t and on the previous state. The recurrent state-update function (dashed red lines) and the output function (dashed blue lines) are emphasized. Description of the grey blocks: 'Generate query', 'Generate key' and 'Generate value' represent functions q, k and v of equation (4) (right); 'Sim' implements the function 'sim' of the same equation (left); 'Aggregate values' evaluates y_t given the similarity scores and the set of values (equation (4), left); 'Update state' computes the current state (equation (6), top), where, for simplicity, we assumed ϕ to be the identity mapping and discarded the z_t-based normalization factor, thus $x_t = S_t$ (matrix); 'Mat–vec product' computes the matrix-by-vector product in equation (6) (bottom).

information among the processed contexts. Several attempts to tackle these two issues have been presented in the past few years. In addition, such approaches inspired recent subquadratic methods that split the overall computation into sequence chunks that are processed in parallel. Overall, we can distinguish among: intra-sequence recurrent modelling[55], where additional encoders are introduced or embeddings are further structured (split embeddings); segment-level recurrence, which introduces the ability to temporally connect different contexts by means of a cache memory (Transformer-XL[56], recurrent memory transformers[57]); and chunk-level recurrence, in which the input sequence is divided into chunks before processing, opening to subquadratic chunk-wise parallel implementations, with (serial) inter-chunk recurrent connections[58]. Other works maintain the context history into a compressive memory, such as Infini-Transformer[59]. It employs both causal local attention and long-term linear attention mechanisms (exploited to retrieve content from memory) into a single transformer block.

State-space models

Concurrently with the recurrent-transformers wave, the need to preserve extremely long-term dependencies in sequences has led to many instances of SSMs combined with deep learning[12,14]. SSMs are formalized as a system of ODEs and, in recent literature, they are based on linear time-invariant systems

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), && \text{State-update function} \\ y(t) &= Cx(t) + Du(t), && \text{Output function} \end{aligned} \tag{8}$$

where, for simplicity, we overloaded the notation t, using it to indicate both a continuous variable and the index of a discrete step. While the emphasis on learning with SSMs originated from the experience with RNNs[10] promoting linear recurrence ('Key concepts of recurrent models'), deep SSMs followed their own independent path. Recently, they have been formally re-connected to RNNs through a careful neural architecture design[8].

Early works. The development of the line of works behind modern deep SSMs can be traced back to Legendre RNNs[10] and HiPPO[11], in which the authors proposed methods to perform online function approximation of (a projection of) the state of an RNN, introducing a linear SSM to drive the update of the approximation coefficients. No learning is involved in the approximation procedure, which is only a way to encode the (projected) state trajectory onto a compact fixed-size representation. As the proposed function approximation is designed for scalar functions, the state x_t is first projected, by means of projection π, onto a single value. In details, the scalar function approximation strategy is the outcome of (1) defining a notion of integrability with appropriate measure, (2) selecting specific classes of basis functions, and (3) defining online schemes to update the coefficients of the basis-based expansion. For example, the class of basis functions with which the approximation is performed in the case of HiPPO consists of translated and rescaled Legendre polynomials. Once the basis functions have been selected, the approximation can be expressed as a combination involving N coefficients, collected into vector c_t (ref. 11). A continuous time-variant SSM is designed to promote online approximation of the coefficients and, afterwards, computations are appropriately discretized[60], yielding $c_t = (1 - A/t)\,c_{t-1} + t^{-1}b\pi(x_t)$, where matrix A and vector b are evaluated in closed forms[11] and not learned from data. Notice that such an equation is a linear recurrence relation. Going back to the function-approximation-augmented RNN, the network is provided input u_t, the previous state x_{t-1} (as usual) and the coefficients c_t. Then, the network weights are learned from data. The leap from this hybrid model to more direct implementations of linear SSMs, that is, not depending on further RNN modules, has been proposed in ref. 13. In this case, the state-update function is directly $\dot{x}(t) = Ax(t) + Bu(t)$, as introduced in equation (8) (top), with learnable A and B. In ref. 13, a stack of these linear SSMs is aggregated into multilayer deep models, while in ref. 12 such models are refined to overcome some computational limitations. In these works, the emphasis is on inheriting the experience from the HiPPO theory and to introduce learning once A and B belong to a particular class of structured matrices—matrices that

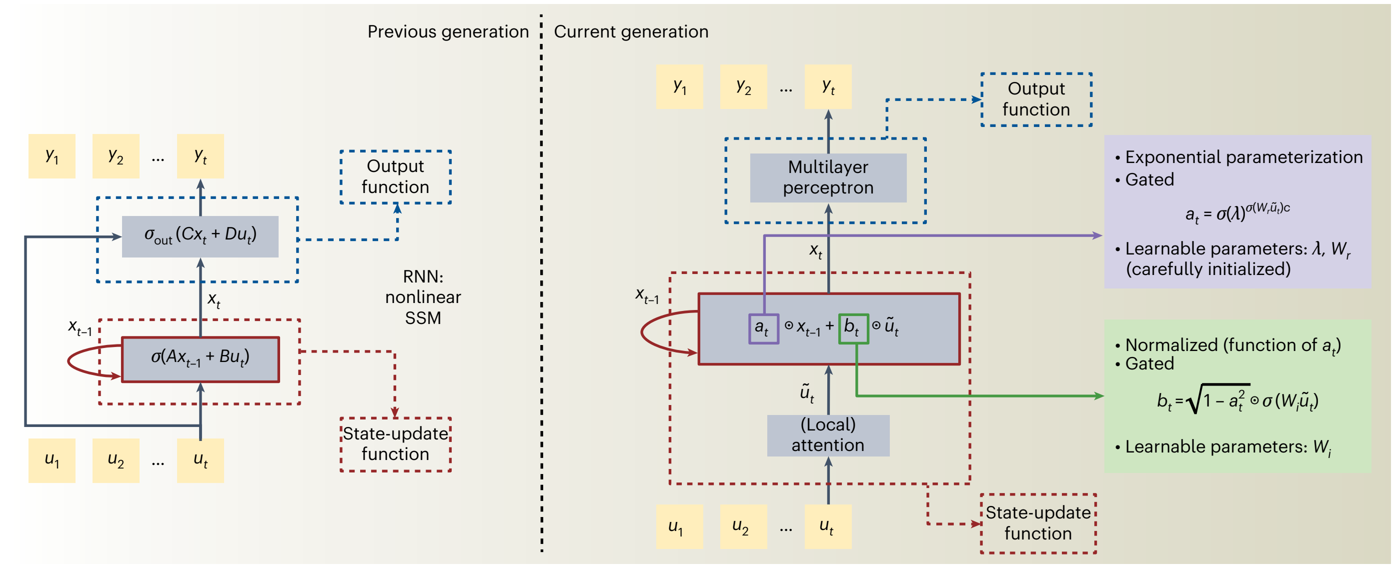
Previous generation
Current generation
y_1 y_2 … y_t
Output function
$\sigma_{out}(Cx_t + Du_t)$
x_t
x_{t-1}
$\sigma(Ax_{t-1} + Bu_t)$
RNN: nonlinear SSM
State-update function
u_1 u_2 … u_t
Multilayer perceptron
$a_t \odot x_{t-1} + b_t \odot \tilde{u}_t$
$\tilde{u}_t$
(Local) attention
• Exponential parameterization
• Gated
$a_t = \sigma(\lambda)^{\sigma(W_r\tilde{u}_t)c}$
• Learnable parameters: λ, W_r (carefully initialized)
• Normalized (function of a_t)
• Gated
$b_t = \sqrt{1 - a_t^2} \odot \sigma(W_i\tilde{u}_t)$
• Learnable parameters: W_i

Fig. 3 | Evolution of the output computation in SSMs. SSMs compute y_t given a new input token u_t. Left: previous generation, with a common example of a nonlinear, time-invariant SSM implemented by an RNN, following the notation of equation (1). Right: the core components of a recently proposed model (see ref. 16 for the full architecture) as an example of the modern notion of an SSM: the state-transition function is linear, parameterized by diagonal matrices (represented by vectors a_t and b_t, where $\odot$ is the Hadamard product), and it is also time variant due to the effects of learnable gating functions of the current input (where σ is a sigmoid). Nonlinear modelling capabilities are recovered by means of the multilayer perceptron on top, and there is also a local attention module that transforms the input, limited to k past tokens (that is, it caches the last k keys and values, instead of all of them, as in Fig. 2, left). The recurrent state-update function (dashed red lines) and the output function (dashed blue lines) are emphasized.

can be represented with subquadratic parameters using a compressed form, possibly enabling efficient matrix multiplication, exploiting the computational benefits of linearity in the state-update equation.

Recent variants. The basic principles of deep SSMs have been the subject of several recent works, leading to many variants of SSMs. Notably, S4[12], which exploits specific initialization based on HiPPO theory and learns a state matrix in diagonal form (actually diagonal low-rank), followed by diagonal state spaces (DSSs)[61], specifically focused on the effectiveness of diagonal matrix-based implementations of the state-update function and approximated HiPPO initialization, further theoretically expanded in S4D with a focus on the diagonal matrices of complex numbers with a negative real part[60]. These works exploit linear recurrence and diagonal state matrices to implement fast computational models. It turns out that the state (output) computation can be rewritten as a convolutional process, with $*$ the convolution operator[62], $x_t = \sum_{z=0}^{t-1} A^z B u_{t-z} = (K_x * u)_t$ ($y_t = \sum_{z=0}^{t-1} (CA^z B + \delta_z D) u_{t-z} = (K_y * u)_t$, with $\delta_z = 1$ only when $z = 0$, otherwise $\delta_z = 0$), where the power A^z is trivial to evaluate due to the diagonal form of A. Related routes were followed by S5[14] introducing parallel scans, multi-input multi-output formulations (going beyond the single-input single-output nature of the previous models), and other ingredients. In this case, multiple states at different time instants can be efficiently computed in parallel by associative scans[63]. There also exist hybrid models integrating gated networks and SSMs or restricting computations to real numbers[33]. H3[64] stacks two discrete SSMs, one with a shift matrix (that is, a local convolution) to remember past tokens and one with a diagonal matrix to retain state over the entire sequence. The key innovation lies in introducing gates (that is, multiplicative interactions inspired by GAU[34]) between the outputs of the SSMs and projections of the input that, combined with the shift matrix, enable H3 to compare tokens across the sequence. Despite the efficiency and effectiveness of models based on linear time-invariant systems in several benchmarks[12], there is still an expressivity gap with respect to alternative solutions (for example, attention; also refer to 'Future directions and challenges'), such as in the case of language or multi-modal data. Liquid-S4[65] was proposed to enhance the expressivity of linear time-invariant systems (see also 'Other trends' section), exploiting the original formulation of S4 (with low-rank correction) combined with a linearized version of a liquid time-constant network[66] to get a linear time-varying system. Recently, S6 (Mamba)[15] introduced a linear time-varying (input dependent) representation of state-update dynamics, referred to as a 'selection mechanism', as long as gating-like functions inspired by GAU and H3. S6 was inspired by intuitions that are related to those which led to Liquid-S4, that is, augmenting S4 with an input-dependent state transition. This allows to address limitations of previous models in several tasks (for example, induction heads[67], associative recall[68], denoise/selective copy[69]) by selectively propagating or forgetting information along the sequence length dimension depending on the current token.

SSMs re-meet recurrent nets. The results achieved by SSMs when facing long sequences, paired with efficient training and fast inference, intrigued researchers to better understand the motivations behind them[8]. It turned out that some basic ingredients of SSMs that are crucial for their performance can be immediately transposed to the design of RNNs. In particular, referring to equation (1) and to Fig. 3, they are as follows. (1) Linear recurrence. (2) Diagonal state matrix A (whose diagonal is vector a) with proper initialization (each element of a can also be a complex number, which is a convenient way to represent the diagonalization of non-symmetric matrices, and it is is also an eigenvalue of A; it is initialized by sampling the unit disk. In turn, B and C can also be learnable matrices with complex values, and only the real part of Cx_t is preserved when computing y_t). (3) Exponential parameterization of the learnable elements in a to favour stability. (4) Normalization of the Bu_t, where also B can be a diagonal matrix (with diagonal b). These choices are initially evaluated in ref. 8, where the so-called LRU is proposed. A deep RNN is instantiated by a linear encoder of the input data, followed by a stack of blocks, each of them composed by an LRU and a multi-layer perceptron (MLP; or a gated linear unit) with the same input and output sizes, which introduces nonlinearity in the net. Indeed, combining linear RNN layers with nonlinear MLP blocks allows to capture complex mappings without nonlinearities in the recurrence[70]. Overall, these models achieve performances that are similar to the state-of-the art in the long-range arena benchmark[71]. In language modelling, the RG-LRU model[16] (presented in two flavours, Hawk and Griffin) achieved impressive results by also exploiting gated linear recurrences with local attention, as summarized in Fig. 3. Several theoretical analyses on representational capability and connection to other fields (for example, rough path theory, control) have been carried out recently[41]. Starting from a different perspective, the Hyena model family[62,72,73] involves a long-convolution operation, implicitly parameterized using feed-forward neural networks, for efficiency, and element-wise multiplicative gating, which selectively modulates the information flow in a very effective way for LLMs. Laughing Hyena distillery[62] has been developed to boost the efficiency of long-convolution models like Hyena, especially during the inference phase in autoregressive tasks. It revisits each convolutional filter by an SSM with minimal state dimensions, using modal interpolation to find optimal coefficients for the rational function of the SSM. The approach involves a pre-filling step, where the model computes an initial state to start generating new tokens, and a recurrent update rule for generating tokens in sequence.

Other trends

Going beyond transformers and SSMs, the resurgence of recurrence is witnessed also by the presence of several research activities specifically focused on improving RNNs from different point of views. These activities involve a large variety of approaches, that can be categorized into well-defined directions and that we only briefly summarize in the following.

Enhancing RNNs. Some works focus on exploiting diagonal state matrices (as in many SSMs) or, similarly, on formulations fully based on element-wise products, still keeping nonlinear activation functions[74]. Other works try to improve the learning/inference time by exploiting properties of specific classes of recurrent models or in interleaving recurrence and convolutions[63]. A large body of literature revisits gating mechanisms, typical of LSTMs, that is, the recent xLSTM[75] injects recent findings (exponential gating, matrix states) to enhance the model capabilities and scale up to billion of parameters. We mention approaches that introduce an additive learnable value to the original forget gate, with the purpose of pushing gate activations away from the saturated regimes, such as hierarchically gated recurrent neural networks (HGRNs)[76].

Continuous-time dynamical systems and beyond. Despite having been the subject of several investigations over the years[77], a renewed interest in continuous-time RNNs emerged due to the popularity of neural ODEs[25,78]. In the family of neural-ODE-based approaches, the hidden state solves the ODE system $\dot{x}(t) = f(x(t), u(t), t, \theta)$, with t representing continuous time, and f a neural network parameterized by θ. Several works explored neural-ODE application to irregular time-series data[79,80], injecting brain-inspired neural computation principles[81] or delving into a deeper theoretical analysis[82]. One of the main advantages of these approaches is their intrinsic nonlinear time-varying nature. Liquid time-constant networks (LTCs)[66] are an expressive form of nonlinear continuous-time RNNs that, instead of defining the derivatives of the hidden-state by a neural network f (as in neural ODEs), add more structure to the state equation to enhance the stability properties of the continuous-time RNN. Specifically, LTCs consider $\dot{x}(t) = -\left(A + B \odot f(x(t), u(t), t, \theta)\right) \odot x(t) + B \odot f(x(t), u(t), t, \theta)$. The neural network f not only directly influences the derivative of the state but also acts as an input-dependent variable timestep in the discretized version of the model. This dynamical behaviour, referred to as 'liquid', enables flexible adaptation within the learning system. In addition, LTCs incorporate input-controlled, state-controlled and data-dependent gating mechanisms, enhancing the selective processing abilities of the model by adjusting the information flow in function of different control factors. However, the nonlinear nature of neural ODEs and LTCs, while beneficial for expressive power, make training and scaling these models challenging[12]. In parallel, alternative families of ODE-inspired models, such as neural oscillators[83,84], have been proposed to approximate any continuous operator mapping between time-varying functions to desired accuracy[85]. Finally, an orthogonal direction is the one that focuses on learning algorithms for recurrent models (see refs. 17,86 and references therein).

Transformers and SSMs meet each other. Recent work studied the connection between transformers and SSMs from different points of view. Of course, the notion of recurrence creates a natural bridge between the two families of models, as we frequently remarked in this paper. Another RNN-based understanding of attention and transformers in general was recently presented in ref. 87, showing how the typical way of computing attention mirrors a 'many-to-one' RNN, processing multiple input tokens to get a single output. Other connections have been drawn by recent models emerging from the transformers literature, such as RetNet[35] (see 'Transformers' section), which introduces a retention mechanism that can be written in recurrent from, thus resembling an SSM. A further link between transformers and SSMs emerges from the concept of structured state-space duality (SSD), proposed in ref. 88. SSD is rooted on the notion of structured matrix, which is used to re-frame different SSMs and to formalize the so-called structured masked attention, which generalizes linear attention. SSMs and structured masked attention are shown to be duals of each other, both possessing SSM-like linear forms and those quadratic forms that are typical of attention models. Owing to SSD, concepts such as multi-value attention can be translated to SSMs, leading to multi-input SSMs. This opens up the development of novel architectures and optimization techniques, for example, Mamba-2[88], which leverage SSD for efficient computation and to incorporate elements from the design of transformer, such as parallel projections and normalization layers. A recent analysis also confines both transformers and SSMs to the same complexity class of problems, showing that those architectures can only recognize languages within TC^0 complexity[89].

Future directions and challenges

The previous sections reviewed the recent developments in transformers and SSMs, emphasizing the return of recurrent processing. The research perspective of the coming years must be grounded on the knowledge of the limitations of these families of models, especially when it comes to linear recurrence. There is room for finding compromises and hybrid solutions, and also to jointly think about models and how they cope with the hardware in which they are expected to run. Finally, a proposed and still open research perspective for the models in this paper is the one of learning continuously from a single, infinite-length sequence in an online manner.

Limitations of transformers

Recent works experimentally evaluated transformers in different types of task to study their limitations. Hahn[90] showed the inability of transformers to recognize simple patterns, such as parity or balanced parentheses, for very large inputs. Peng et al.[91] deepened the analysis on the limitations of transformers in some types of function composition, a critical operation for combining relational information and understanding language. For example, answering questions that require combining several facts turned out to be challenging for transformers. Chain of thoughts can help mitigate these issues, but they require significantly more tokens to describe complex chains. Moreover, transformer networks showed limits in handling time-series forecasting, where much simpler alternatives can achieve better results[92].

Limitations of linear recurrence

In 'Resurgence of recurrence', we discussed alternative attention schemes to mitigate the quadratic complexity with respect to the sequence. Such models exploit stateful architectures and linear recurrence. Of course, choosing linearity comes at the price of a performance degradation when compared with vanilla transformers, which can be motivated by the expressivity gap between linear recurrence and softmax attention. The latter, despite being less efficient in computational terms, naturally comes with the ability to recall earlier tokens of the sequence and to perform comparisons among them; such a feature is hardly recovered by stateful models, which are required to progressively encode the sequence in their state. From a theoretical point of view, this limitation was pointed out by Jelassi et al.[93], showing that the network ability to recall information observed several time steps in the past is more pronounced in transformers than SSMs. Orvieto et al.[70] claim that SSMs with linear recurrence followed by a nonlinear function parameterized by an MLP are universal approximators. Their proof comes from the fact that linear RNNs could memorize the input performing a lossless encoding, while the following MLP can then reconstruct the information due to its universal approximation capabilities. Still, the expressivity of SSMs can be enhanced by letting state-transition matrices be dependent from the input, theoretically unlocking NC^1-hard problems and resulting in improved performances[89], as shown by Liquid-S4[65].

Hybrid models

The expressivity gap issues raised by Jelassi et al.[93] result in the inability of SSM-based networks to reach the performances of attention-only competitors (for example, Mistral 7B[94]) in complex tasks such as language modelling. To solve such issues, several recent models (for example, Griffin[16]) build hybrid architectures incorporating local attention, also known as sliding window attention, into SSM-based models (LRUs[8]). Local attention allows each position to attend to a fixed number of past tokens, reducing the number of FLOPs compared with standard attention and limiting the KV-cache size to the window size. Several other hybrid architectures interleaving attention and SSM layers have been proposed recently: Jamba/Jamba 1.5[95] introduces a novel architecture that combines transformers and Mamba layers, enhanced with a mixture-of-experts mechanism, allowing the model to maintain high performance while managing memory and computational efficiency. In their detailed study of various possible hybrid architectures, Poli et al.[73] found that a compute-optimal consists in approximately one quarter of the layers being based on self-attention and the rest of SSMs. Zamba[96] is a novel 7 B SSM–transformer hybrid model capable of achieving competitive performance against leading open-weight models of similar scale. This is accomplished through an architecture that combines a Mamba backbone with a single shared attention module, capturing the benefits of attention at minimal parameter cost.

Hardware efficiency

Although the linear approximations of vanilla attention reduce computational requirements, practical implementations often lack substantial speed-ups, primarily due to memory access limitations inherent in current hardware such as graphics processing units and tensor processing units[97]. Specifically, transformers are often memory-bound due to frequent data transfers between high-bandwidth memory and on-chip storage, such as static random-access memory. Optimizations such as FlashAttention[97] address these bottlenecks through tiling and non-materialization techniques, which reduce memory I/O by preventing the storage of large attention matrices. Advanced models such as gated linear attention[50] and Mamba[15] further enhance efficiency using hardware-aware features such as kernel fusion and parallel scans, allowing them to reduce memory costs while maintaining expressiveness. The Griffin model[16] leverages custom tensor processing unit kernels to improve the efficiency in memory transfer by using chunked storage in vector memory. These developments remark that for long sequence processing, algorithmic and hardware-specific optimizations are critical for overcoming memory bandwidth limitations and enhancing efficiency in real-world applications.

Learning from an infinite-length sequence

Although many of the described models aim to process long sequences, extending to an infinite-length sequence remains largely uncharted. Architectures are required to adapt continually to new data while retaining prior knowledge—a central concept to online continual learning[98]—and the learning algorithms are required to compute gradients at each timestep. However, most existing learning algorithms are designed for datasets of finite-length sequences (for example, backpropagation through time). Despite some advancements, such as variants of real-time recurrent learning for online gradient computation[86], and state-space approaches for forward gradient computation[99], it is still challenging to meet the lifelong-learning requirements[98]. This represents a clear research opportunity, especially to design applications that process streams of video, audio or real-time sensor data[100,101].

Conclusion

The improvements in sequence processing of the past few years, both in terms of performance and scalability, are mostly represented by the resurgence of recurrent models. Interestingly, the recent literature in the context of transformers and SSMs share the same intuitions, promoting stateful models and linear recurrence, the latter paired with additional components. We reviewed the most important works, emphasizing their connection with older recurrent models. We believe that being aware of the connections in recent literature about apparently different models is crucial to create the bases to the development of novel technologies in sequence processing, which is a key topic when learning from perceptual data, videos, text and others.

References

1. Brown, T. et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020).
2. Touvron, H. et al. Llama 2: open foundation and fine-tuned chat models. Preprint at https://doi.org/10.48550/arXiv.2307.09288 (2023).
3. Gemini Team Google. Gemini: a family of highly capable multimodal models. Preprint at https://doi.org/10.48550/arXiv.2312.11805 (2023).
4. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems* Vol. 30 (NuerIPS, 2017).
5. Yu, Y., Si, X., Hu, C. & Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **31**, 1235–1270 (2019).
6. Bengio, Y., Mori, R. & Gori, M. Learning the dynamic nature of speech with back-propagation for sequences. *Pattern Recognit. Lett.* **13**, 375–385 (1992).
7. Gori, M., Hammer, B., Hitzler, P. & Palm, G. Perspectives and challenges for recurrent neural network training. *Log. J. IGPL* **18**, 617–619 (2010).
8. Orvieto, A. et al. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning* 26670–26698 (ACM, 2023).
9. Peng, B. et al. RWKV: reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (eds Bouamor, H. et al.) 14048–14077 (ACL, 2023).
10. Voelker, A., Kajić, I. & Eliasmith, C. Legendre memory units: continuous-time representation in recurrent neural networks. In *Advances in Neural Information Processing Systems 32* (NeurIPS, 2019.)
11. Gu, A., Dao, T., Ermon, S., Rudra, A. & Ré, C. HiPPO: recurrent memory with optimal polynomial projections. *Adv. Neural Inf. Process. Syst.* **33**, 1474–1487 (2020).
12. Gu, A., Goel, K. & Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations* (Curran Associates, 2021).

13. Gu, A. et al. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Adv. Neural Inf. Process. Syst.* **34**, 572–585 (2021).
14. Smith, J. T., Warrington, A. & Linderman, S. Simplified state space layers for sequence modeling. In *11th International Conference on Learning Representations* (Curran Associates, 2023).
15. Gu, A. & Dao, T. Mamba: linear-time sequence modeling with selective state spaces. Preprint at https://doi.org/10.48550/arXiv.2312.00752 (2023).
16. De, S. et al. Griffin: mixing gated linear recurrences with local attention for efficient language models. Preprint at https://doi.org/10.48550/arXiv.2402.19427 (2024).
17. Marschall, O., Cho, K. & Savin, C. A unified framework of online learning algorithms for training recurrent neural networks. *J. Mach. Learn. Res.* **21**, 5320–5353 (2020).
18. Elman, J. L. Finding structure in time. *Cogn. Sci.* **14**, 179–211 (1990).
19. Siegelmann, H. T. *Neural Networks and Analog Computation: Beyond the Turing Limit* (Springer, 2012).
20. Li, Z., Han, J., E, W. & Li, Q. Approximation and optimization theory for linear continuous-time recurrent neural networks. *J. Mach. Learn. Res.* **23**, 1997–2081 (2022).
21. Tallec, C. & Ollivier, Y. Can recurrent neural networks warp time? In *International Conference on Learning Representation* (Curran Associates, 2018).
22. Kitagawa, G. A self-organizing state-space model. *J. Am. Stat. Assoc.* **93**, 1203–1215 (1998).
23. Lipton, Z. C., Berkowitz, J. & Elkan, C. A critical review of recurrent neural networks for sequence learning. Preprint at https://doi.org/10.48550/arXiv.1506.00019 (2015).
24. Salehinejad, H., Sankar, S., Barfett, J., Colak, E. & Valaee, S. Recent advances in recurrent neural networks. Preprint at https://doi.org/10.48550/arXiv.1801.01078 (2017).
25. Kidger, P. On neural differential equations. Preprint at https://doi.org/10.48550/arXiv.2202.02435 (2022).
26. Rumelhart, D. E. et al. *Learning Internal Representations by Error Propagation* (Institute for Cognitive Science, Univ. California, San Diego 1985).
27. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
28. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* 1310–1318 (ACM, 2013).
29. Hochreiter, S. Untersuchungen zu dynamischen neuronalen netzen. Diploma, Technische Univ. München (1991).
30. Bengio, Y. & Frasconi, P. Credit assignment through time: alternatives to backpropagation. In *Advances in Neural Information Processing Systems* Vol. 6 (NeurIPS, 1993).
31. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
32. Cho, K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. 2014 Conference on Empirical Methods in Natural Language Processing* 1724–1734 (EMNLP, 2014).
33. Mehta, H., Gupta, A., Cutkosky, A. & Neyshabur, B. Long range language modeling via gated state spaces. In *11th International Conference on Learning Representations* (Curran Associates, 2023).
34. Hua, W., Dai, Z., Liu, H., Le, Q. Transformer quality in linear time. In *International Conference on Machine Learning* 9099–9117 (ACM, 2022).
35. Sun, Y. et al. Retentive network: a successor to transformer for large language models. Preprint at https://doi.org/10.48550/arXiv.2307.08621 (2023).
36. Arjovsky, M., Shah, A. & Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning* 1120–1128 (ACM, 2016).
37. Mhammedi, Z., Hellicar, A., Rahman, A. & Bailey, J. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning* (ACM, 2017).
38. Kag, A. & Saligrama, V. Training recurrent neural networks via forward propagation through time. In *International Conference on Machine Learning* 5189–5200 (ACM, 2021).
39. Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **27**, 3104–3112 (2014).
40. Karuvally, A., Sejnowski, T. & Siegelmann, H. T. Hidden traveling waves bind working memory variables in recurrent neural networks. In *41st International Conference on Machine Learning* (ACM, 2024).
41. Sieber, J., Alonso, C. A., Didier, A., Zeilinger, M. N. & Orvieto, A. Understanding the differences in foundation models: attention, state space models, and recurrent neural networks. Preprint at https://doi.org/10.48550/arXiv.2405.15731 (2024).
42. Tay, Y., Dehghani, M., Bahri, D. & Metzler, D. Efficient transformers: a survey. *ACM Comput. Surv.* **55**, 109–110928 (2023).
43. Katharopoulos, A., Vyas, A., Pappas, N. & Fleuret, F. Transformers are RNNs: fast autoregressive transformers with linear attention. In *International Conference on Machine Learning* 5156–5165 (ACM, 2020).
44. Peng, H. et al. Random feature attention. In *Proc. 9th International Conference on Learning Representations* (Curran Associates, 2021).
45. Schlag, I., Irie, K. & Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning* 9355–9366 (ACM, 2021).
46. Qin, Z. et al. The devil in linear transformer. In *Proc. 2022 Conference on Empirical Methods in Natural Language Processing* 7025–7041 (EMNLP, 2022).
47. Schmidhuber, J. Learning to control fast-weight memories: an alternative to dynamic recurrent networks. *Neural Comput.* **4**, 131–139 (1992).
48. Qin, Z. et al. Scaling transnormer to 175 billion parameters. Preprint at https://doi.org/10.48550/arXiv.2307.14995 (2023).
49. Katsch, T. GateLoop: fully data-controlled linear recurrence for sequence modeling. Preprint at https://doi.org/10.48550/arXiv.2311.01927 (2023).
50. Yang, S., Wang, B., Shen, Y., Panda, R. & Kim, Y. Gated linear attention transformers with hardware-efficient training. In *Proc. 41st International Conference on Machine Learning* (ACM, 2024).
51. Dauphin, Y. N., Fan, A., Auli, M. & Grangier, D. Language modeling with gated convolutional networks. In *International Conference on Machine Learning* 933–941 (ACM, 2017).
52. Ma, X. et al. MEGA: moving average equipped gated attention. In *11th International Conference on Learning Representations* (Curran Associates, 2022).
53. Zhai, S. et al. An attention free transformer. Preprint at https://doi.org/10.48550/arXiv.2105.14103 (2021).
54. Peng, B. et al. Eagle and Finch: RWKV with matrix-valued states and dynamic recurrence. Preprint at https://doi.org/10.48550/arXiv.2404.05892 (2024).
55. Huang, F. et al. Encoding recurrence into transformers. In *11th International Conference on Learning Representations* (Curran Associates, 2022).
56. Dai, Z. et al. Transformer-XL: attentive language models beyond a fixed-length context. In *Proc. 57th Annual Meeting of the Association for Computational Linguistics* (ACL, 2019).
57. Bulatov, A., Kuratov, Y. & Burtsev, M. Recurrent memory transformer. *Adv. Neural Inf. Process. Syst.* **35**, 11079–11091 (2022).

58. Didolkar, A. et al. Temporal latent bottleneck: synthesis of fast and slow processing mechanisms in sequence learning. *Adv. Neural Inf. Process. Syst.* **35**, 10505–10520 (2022).
59. Munkhdalai, T., Faruqui, M. & Gopal, S. Leave no context behind: efficient infinite context transformers with infini-attention. Preprint at https://doi.org/10.48550/arXiv.2404.07143 (2024).
60. Gu, A., Goel, K., Gupta, A. & Ré, C. On the parameterization and initialization of diagonal state space models. *Adv. Neural Inf. Process. Syst.* **35**, 35971–35983 (2022).
61. Gupta, A., Gu, A., Berant, J. Diagonal state spaces are as effective as structured state spaces. In *36th Conference on Neural Information Processing Systems* (NeurIPS, 2022).
62. Massaroli, S. et al. Laughing hyena distillery: extracting compact recurrences from convolutions. In *37th Conference on Neural Information Processing Systems* (NeurIPS, 2023).
63. Martin, E. & Cundy, C. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations* (Curran Associates, 2018).
64. Dao, T. et al. Hungry hungry hippos: towards language modeling with state space models. In *Proc. 11th International Conference on Learning Representations* (Curran Associates, 2023).
65. Hasani, R. et al. Liquid structural state-space models. In *11th International Conference on Learning Representations* (Curran Associates, 2023).
66. Hasani, R., Lechner, M., Amini, A., Rus, D. & Grosu, R. Liquid time-constant networks. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 35, 7657–7666 (2021).
67. Olsson, C. et al. In-context learning and induction heads. Preprint at https://doi.org/10.48550/arXiv.2209.11895 (2022).
68. Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z. & Ionescu, C. Using fast weights to attend to the recent past. *Adv. Neural Inf. Process. Syst.* **29**, 4338–4346 (2016).
69. Jing, L. et al. Gated orthogonal recurrent units: on learning to forget. *Neural Comput.* **31**, 765–783 (2019).
70. Orvieto, A., De, S., Gulcehre, C., Pascanu, R. & Smith, S. L. Universality of linear recurrences followed by non-linear projections: finite-width guarantees and benefits of complex eigenvalues. In *41st International Conference on Machine Learning* (ACM, 2024).
71. Tay, Y. et al. Long range arena: a benchmark for efficient transformers. In *9th International Conference on Learning Representations* (Curran Associates, 2021).
72. Poli, M. et al. Hyena hierarchy: towards larger convolutional language models. In *International Conference on Machine Learning* 28043–28078 (ACM, 2023).
73. Poli, M. et al. Mechanistic design and scaling of hybrid architectures. In *41st International Conference on Machine Learning* (ACM, 2024).
74. Li, S., Li, W., Cook, C., Zhu, C. & Gao, Y. Independently recurrent neural network (IndRNN): building a longer and deeper RNN. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 5457–5466 (IEEE, 2018).
75. Beck, M. et al. xLSTM: extended long short-term memory. *Adv. Neural Inf. Process. Syst.* **37**, 107547–107603 (2025).
76. Qin, Z., Yang, S. & Zhong, Y. Hierarchically gated recurrent neural network for sequence modeling. In *Thirty-seventh Conference on Neural Information Processing Systems* (NeurIPS, 2023).
77. Mandic, D. P. & Chambers, J. A. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability* (John Wiley & Sons, 2001).
78. Chen, R. T., Rubanova, Y., Bettencourt, J. & Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems* Vol. 31 (NeurIPS, 2018).
79. Rubanova, Y., Chen, R. T. & Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems* Vol. 32 (NeurIPS, 2019).
80. Kidger, P., Morrill, J., Foster, J. & Lyons, T. Neural controlled differential equations for irregular time series. *Adv. Neural Inf. Process. Syst.* **33**, 6696–6707 (2020).
81. Lechner, M. et al. Neural circuit policies enabling auditable autonomy. *Nat. Mach. Intell.* **2**, 642–652 (2020).
82. Massaroli, S., Poli, M., Park, J., Yamashita, A. & Asama, H. Dissecting neural odes. *Adv. Neural Inf. Process. Syst.* **33**, 3952–3963 (2020).
83. Rusch, T. K. & Mishra, S. UnICORNN: a recurrent model for learning very long time dependencies. In *International Conference on Machine Learning* 9168–9178 (ACM, 2021).
84. Effenberger, F., Carvalho, P., Dubinin, I. & Singer, W. The functional role of oscillatory dynamics in neocortical circuits: a computational perspective, *Proc. Natl. Acad. Sci. USA* **122**, e2412830122 (2025).
85. Lanthaler, S., Rusch, T. K. & Mishra, S. Neural oscillators are universal. *Adv. Neural Inf. Process. Syst.* **36**, 46786–46806 (2024).
86. Irie, K., Gopalakrishnan, A. & Schmidhuber, J. Exploring the promise and limits of real-time recurrent learning. In *12th International Conference on Learning Representations* (Curran Associates, 2024).
87. Feng, L. et al. Attention as an RNN. Preprint at https://doi.org/10.48550/arXiv.2405.13956 (2024).
88. Dao, T. & Gu, A. Transformers are SSMs: generalized models and efficient algorithms through structured state space duality. In *41st International Conference on Machine Learning* (ACM, 2024).
89. Merrill, W., Petty, J. & Sabharwal, A. The illusion of state in state-space models. In *41st International Conference on Machine Learning* 35492–35506 (ACM, 2024).
90. Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguist.* **8**, 156–171 (2020).
91. Peng, B., Narayanan, S. & Papadimitriou, C. On limitations of the transformer architecture. In *First Conference on Language Modeling* (COLM, 2024).
92. Zeng, A., Chen, M., Zhang, L. & Xu, Q. Are transformers effective for time series forecasting? In *Proc. AAAI Conference on Artificial Intelligence* Vol. 37, 11121–11128 (AAAI, 2023).
93. Jelassi, S., Brandfonbrener, D., Kakade, S. M. & Malach, E. Repeat after me: transformers are better than state space models at copying. In *41st International Conference on Machine Learning* (ACM, 2024).
94. Jiang, A. Q. et al. Mistral 7B. Preprint at https://doi.org/10.48550/arXiv.2310.06825 (2023).
95. Team, J. et al. Jamba-1.5: hybrid transformer-Mamba models at scale. Preprint at https://doi.org/10.48550/arXiv.2408.12570 (2024).
96. Glorioso, P. et al. Zamba: a compact 7B SSM hybrid model. Preprint at https://doi.org/10.48550/arXiv.2405.16712 (2024).
97. Dao, T. FlashAttention-2: faster attention with better parallelism and work partitioning. In *12th International Conference on Learning Representations* (Curran Associates, 2024).
98. Casoni, M. et al. Pitfalls in processing infinite-length sequences with popular approaches for sequential data. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition* 37–48 (Springer, 2024).
99. Zucchet, N., Meier, R., Schug, S., Mujika, A. & Sacramento, J. Online learning of long-range dependencies. *Adv. Neural Inf. Process. Syst.* **36**, 10477–10493 (2023).

100. Betti, A., Gori, M. & Melacci, S. Learning visual features under motion invariance. *Neural Networks* **126**, 275–299 (2020).

101. Tiezzi, M. et al. Stochastic coherence over attention trajectory for continuous learning in video streams. In *Proc. 31st International Joint Conference on Artificial Intelligence* 3480–3486 (IJCAI, 2022).

Acknowledgements
This work was partially supported by the University of Siena (Piano per lo Sviluppo della Ricerca - PSR 2024, F-NEW FRONTIERS 2024), under the project 'TIme-driveN StatEful Lifelong Learning' (TINSELL) and also by project 'CONSTR: a COllectionless-based Neuro-Symbolic Theory for learning and Reasoning', PARTENARIATO ESTESO 'Future Artificial Intelligence Research - FAIR', SPOKE 1 'Human-Centered AI' Università di Pisa, 'NextGenerationEU', CUP I53C22001380006.

Author contributions
M.T. and S.M. wrote the paper with the help of all authors. S.M. prepared the figures and supervised the work. M.T. explored the existing literature. M.T., M.C., A.B., T.G. and S.M. contributed to reading and organizing the descriptions of the works cited in this review. M.G. contributed to describing the limitations of existing works and proposing future challenges, with assistance from all authors.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to Stefano Melacci.

Peer review information *Nature Machine Intelligence* thanks Hava Siegelmann and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.