entropy

MDPI

Review

Attention Mechanisms and Their Applications to Complex Systems

Adrián Hernández and José M. Amigó *

Centro de Investigación Operativa, Universidad Miguel Hernández, Av. de la Universidad s/n, 03202 Elche, Spain; adrian.hernandez@goumh.umh.es
* Correspondence: jm.amigo@umh.es

Abstract: Deep learning models and graphics processing units have completely transformed the field of machine learning. Recurrent neural networks and long short-term memories have been successfully used to model and predict complex systems. However, these classic models do not perform sequential reasoning, a process that guides a task based on perception and memory. In recent years, attention mechanisms have emerged as a promising solution to these problems. In this review, we describe the key aspects of attention mechanisms and some relevant attention techniques and point out why they are a remarkable advance in machine learning. Then, we illustrate some important applications of these techniques in the modeling of complex systems.

Keywords: attention; deep learning; complex and dynamical systems; self-attention; neural networks; sequential reasoning

Citation: Hernández, A.;Amigó, J.M. Attention Mechanisms and Their Applications to Complex Systems. *Entropy* **2021**, *23*, 283. https://doi.org/10.3390/e23030283

Academic Editor: José A. Tenreiro Machado

Received: 10 January 2021
Accepted: 23 February 2021
Published: 26 February 2021

1. Introduction

The combination of deep neural networks and the computational capabilities of Graphics Processing Units (GPUs) [1] has brought a breakthrough to the field of machine learning, improving the performance of several tasks such as image recognition, machine translation, language modelling, time series prediction, etc. [2–5].

Recurrent neural networks (RNNs) and long short-term memories (LSTMs), which were specially designed for sequence modelling [6–9], and convolutional neural networks (CNNs) to a lesser extent, have been successfully used to model, analyze and predict complex systems. Indeed, they are able to capture temporal dependencies and nontrivial relationships in complex systems, specifically in the sequential data generated by them. By complex systems we mean, generally speaking, systems that evolve over time in a possibly more general setting than that of dynamic systems.

However, these classic deep learning models do not perform sequential reasoning [10], a process that is based on perception with attention. In the brain, attention mechanisms allow to focus on one part of the input or memory (image, text, etc) while giving less attention to others, thus guiding the process of reasoning.

Attention mechanisms have provided and will provide a paradigm shift in machine learning [11,12]. These mechanisms allow a model to focus only on a set of elements and to decompose a problem into a sequence of attention based reasoning tasks [13]. Moreover, they can be applied to model complex systems in a flexible and promising way. When it comes to their application, information processing in the system and internal structure are crucial.

Here, as shown in Table 1, we describe the evolution of machine learning techniques and demonstrate how attention mechanisms, in combination with classic models, allow modeling certain important characteristics of complex systems, e.g., sequential reasoning, integration of different parts and long term dependencies.

Table 1. Classic deep learning and attention techniques described in this paper and their capabilities to model complex systems.

Techniques	Capabilities in Modeling Complex Systems
Classic models (RNNs, LSTMs . . .)	Are universal approximators, provide perception, temporal dependence and short memory
Seq2seq with attention	Integrates parts, models long term dependencies, guides a task by focusing on a set of elements (temporal, spatial, features . . .)
Memory networks	Integrate external data with the current task and provide an explicit external memory
Self-attention	Generalization of neural networks, relates input vectors in a more direct and symmetric way

In this paper, which is aimed at researchers with prior knowledge of deep learning, we review recent progress in attention mechanisms. We focus on differentiable attention, in which the attention weights are learned together with the rest of the model parameters. In Section 2, we present a general overview of the use of deep learning in modeling dynamical systems and, more generally, complex systems. We also elaborate on the need for attention mechanisms. In Section 3, we present the key aspects, the advantages and the main modes of operation of attention (Section 3.1). Then we describe some important attention techniques such as attention in seq2seq models (Section 3.2), as well as self-attention and memory networks (Section 3.3), emphasizing why they represent significant progress in machine learning. Finally, in Sections 4.1–4.4 we illustrate some interesting uses of theses techniques to model complex systems and in Section 5 we discuss these techniques. For the convenience of the reader, all abbreviations in this paper are listed last.

2. Traditional Deep Learning and the Need for Attention

In recent years, we have seen major advances in the field of artificial intelligence and machine learning. The combination of deep neural networks with the computational capabilities of Graphics Processing Units (GPUs) [1] has improved the performance of several tasks such as image recognition, machine translation, language modelling, time series prediction, game playing and more [2–5]. Deep learning models have evolved to take into account the computational structure of the problem to be resolved.

In a feedforward neural network (FNN) composed of multiple layers, the output (without the bias term) at layer l, see Figure 1, is defined as

$$\boldsymbol{x}^{l+1} = f(W^l \boldsymbol{x}^l), \tag{1}$$

W^l being the weight matrix at layer l. f is the activation function and $\boldsymbol{x}^{l+1}$, the output vector at layer l and the input vector at layer $l+1$. The weight matrices for the different layers are the parameters of the model.

Learning is the mechanism by which the parameters of a neural network are adapted to the environment in the training process. This is an optimization problem that has been addressed using gradient-based methods, in which given a cost function $f : \mathbb{R}^n \to \mathbb{R}$, the algorithm finds local minima $w^* = \arg\min_w f(w)$ by updating each layer parameter w_{ij} with the rule $w_{ij} := w_{ij} - \eta \nabla_{w_{ij}} f(w)$, where $\eta > 0$ is the learning rate.

Therefore, a deep learning model consists of the forward pass, in which the computational graph with the multiple layers is built, and the backward pass, in which the gradients are calculated and the parameters are updated. Then, all the functions of the parameters used in the model must be differentiable.

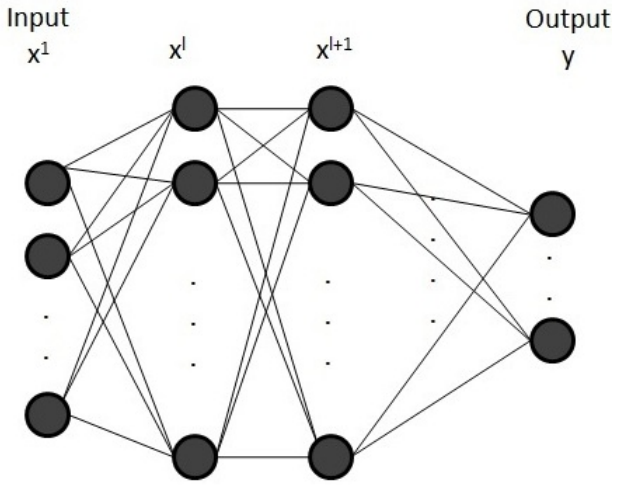

Figure 1. Multilayer neural network.

RNNs (see Figure 2) are a basic component of modern deep learning architectures, especially of encoder–decoder networks. The following equations define the time evolution of an RNN:

$$\boldsymbol{h}_t = f^h(W^{ih}\boldsymbol{x}_t + W^{hh}\boldsymbol{h}_{t-1}), \tag{2}$$

$$\boldsymbol{y}_t = f^o(W^{ho}\boldsymbol{h}_t), \tag{3}$$

where W^{ih}, W^{hh} and W^{ho} are weight matrices. f^h and f^o are the hidden and output activation functions while $\boldsymbol{x}_t$, $\boldsymbol{h}_t$ and $\boldsymbol{y}_t$ are the network input, hidden state and output.

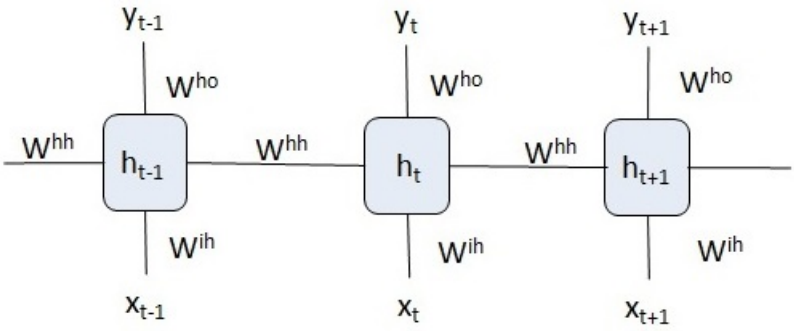

Figure 2. Temporal structure of a recurrent neural network.

LSTMs [14] are an evolution of RNNs in that they feature an RNN structure with gated units, i.e., regulators. Specifically, LSTMs are composed of a memory cell, an input gate, an output gate and a forget gate, and allow gradients to flow unchanged. The memory cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

Here we refer to systems that contain a very large number of variables interacting with each other in non-trivial ways as complex systems [15]. Their behaviour is intrinsically difficult to model due to the dependencies and interactions between their parts and they have emergence properties arising from those interactions such as adaptation, evolution, learning, etc. In Section 4, we describe the use of attention mechanisms to model the sequential data generated by complex systems.

Dynamical systems are a special class of complex systems. At any given time, a dynamical system has a state that can be represented by a point in a state space (manifold). The evolution equations of the dynamical system describes what future states follow from the current state. This process can be deterministic, if its entire future is uniquely determined by its current state, or non-deterministic otherwise [16] (e.g., a random dynamical

system [17]). Furthermore, it can be a continuous-time process, represented by differential equations or a discrete-time process, represented by difference equations or maps. Thus,

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}; \boldsymbol{\theta}) \tag{4}$$

for autonomous discrete-time deterministic dynamical systems with parameters $\boldsymbol{\theta}$, and

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta}) \tag{5}$$

for non-autonomous discrete-time deterministic dynamical systems driven by an external input $\boldsymbol{x}_t$. Dynamical systems with multiple time lags can be rewritten as a higher dimensional dynamical system with time lag 1.

A key aspect in modelling dynamical systems is, of course, temporal dependence. Traditionally, there have been two ways to implement it in the neural network paradigm [18]:

1. Classic feedforward neural networks with time delayed states in the inputs but perhaps with an unnecessary increase in the number of parameters.
2. RNNs since, as shown in Equations (2) and (3), they have a temporal recurrence that make them appropriate for modelling discrete dynamical systems of the form given in Equations (4) and (5). As said in the Introduction, RNNs were precisely designed for sequence modelling. [6].

Therefore, RNNs seem the ideal candidates to model, analyze and predict dynamical systems and more generally complex systems. Theoretically, the temporal recurrence of RNNs allows to model and identify dynamical systems described with equations with any temporal dependence.

To learn chaotic dynamics, recurrent radial basis function (RBF) networks [19] and evolutionary algorithms that generate RNNs have been proposed [20]. "Nonlinear Autoregressive model with exogenous input" (NARX) [21] and boosted RNNs [22] have been applied to predict chaotic time series.

LSTMs have also succeeded in various applications to complex systems such as model identification and time series prediction [7–9]. Another remarkable application of the LSTM is machine translation [3,23].

Although the classic models above work well, they have limitations that make it difficult to perform sequential reasoning and achieve more general intelligence [10,24]. Among these limitations, we highlight the following:

1. Classic models only perform perception, representing a mapping between inputs and outputs.
2. Classic models follow a hybrid model where synaptic weights perform both processing and memory tasks but do not have an explicit external memory.
3. Classic models do not carry out sequential reasoning. This essential process is based on perception and memory through attention and guides the steps of the machine learning model in a conscious and interpretable way.

In the next section, we present attention mechanisms as an important step to address these limitations.

3. Attention Mechanisms

3.1. Differentiable Attention

As explained in Section 2, classic deep learning models do not perform sequential reasoning, a process that is based on attention.

In the brain, reasoning is the process of establishing and verifying facts combining attention with new or existing information. The role of the attention mechanisms is to focus on one part of the input or memory (image, text, etc), thus guiding the process of reasoning.

As described in [25], there are several classes of attention in neuroscience: attention as a level of alertness, attention over sensory inputs, attention to select and execute tasks

and attention for memory encoding and retrieval. In [26], the authors modeled the interaction between top-down attention and bottom-up stimulus contrast effects and found that external attention inputs bias neurons to move to different parts of their nonlinear activation functions. Insects have been a source of inspiration in intelligence and attention mechanisms. In [27], a multiclass support vector machine with inhibition is inspired by the brain structure of insects. In [28], a multi-layer spiking neural network is presented that models the Mushroom Bodies and their interactions to other key elements of the insect brain, the Central Complex and the Lateral Horns.

Analogously, a learning problem in machine learning can be decomposed into a sequence of tasks, where in each task it is necessary to focus on one part of an input (or transformed input) or a memory. Once again, neural information processing in the brain, in which several layers interact with each other [29], has been a source of inspiration for machine learning.

Generally formulated, attention in machine learning is a sequential process in which a learning task is guided by a set of elements of the input source (or memory). This is achieved by integrating the attention value into the task.

Attention mechanisms have provided and will provide a paradigm shift in machine learning. Specifically, this change is from traditional large-scale vector transformations to more conscious processes (i.e., that focus only on a set of elements), e.g., decomposing a problem into a sequence of attention based reasoning tasks [13,30–34].

As stated in Section 2, to integrate a component into a deep learning model that learns using gradient descent, all the functions of the parameters in the component must be differentiable. One way to make attention mechanisms differentiable is to formulate them as a convex combination of the input or memory. In this case, all the steps are differentiable and can be learned, and the combination weights must add up to one (forcing them to focus on some parts more than others). In this way, the mechanisms learn which parts it needs to focus on.

As in [11], this convex combination, shown in Figure 3, is described as mapping a query and a set of key-value pairs to an output:

$$att(\boldsymbol{q}, \boldsymbol{s}) = \sum_{i=1}^{T} \alpha_i(\boldsymbol{q}, \boldsymbol{k}_i)\boldsymbol{V}_i, \tag{6}$$

where, as seen in Figure 3, $\boldsymbol{k}_i$ and $\boldsymbol{V}_i$ are the key and the value vectors from the source/memory $\boldsymbol{s}$, and $\boldsymbol{q}$ is the query vector (task). $\alpha_i(\boldsymbol{q}, \boldsymbol{k}_i)$ is the similarity function between the query and the corresponding key and is calculated by applying the softmax function,

$$Softmax(z_i) = \frac{exp(z_i)}{\sum_{i'} exp(z_{i'})}, \tag{7}$$

to the score function $score(\boldsymbol{q}, \boldsymbol{k}_i)$:

$$\alpha_i = \frac{\exp(score(\boldsymbol{q}, \boldsymbol{k}_i))}{\sum_{i'=1}^{T} \exp(score(\boldsymbol{q}, \boldsymbol{k}_{i'}))}. \tag{8}$$

The score function can be computed using a feedforward neural network:

$$score(\boldsymbol{q}, \boldsymbol{k}_i) = \boldsymbol{Z}_a \tanh(\boldsymbol{W}_a[\boldsymbol{q}, \boldsymbol{k}_i])), \tag{9}$$

as proposed in [35], where $\boldsymbol{Z}_a$ and $\boldsymbol{W}_a$ are matrices to be jointly learned with the rest of the model and $[\boldsymbol{q}, \boldsymbol{k}_i]$ is a linear function or concatenation of $\boldsymbol{q}$ and $\boldsymbol{k}_i$. Furthermore, in [36] the authors use a cosine similarity measure for content-based attention, namely,

$$score(\boldsymbol{q}, \boldsymbol{k}_i) = \cos((\boldsymbol{q}, \boldsymbol{k}_i)), \tag{10}$$

where $((\boldsymbol{q}, \boldsymbol{k}_i))$ denotes the angle between $\boldsymbol{q}$ and $\boldsymbol{k}_i$.

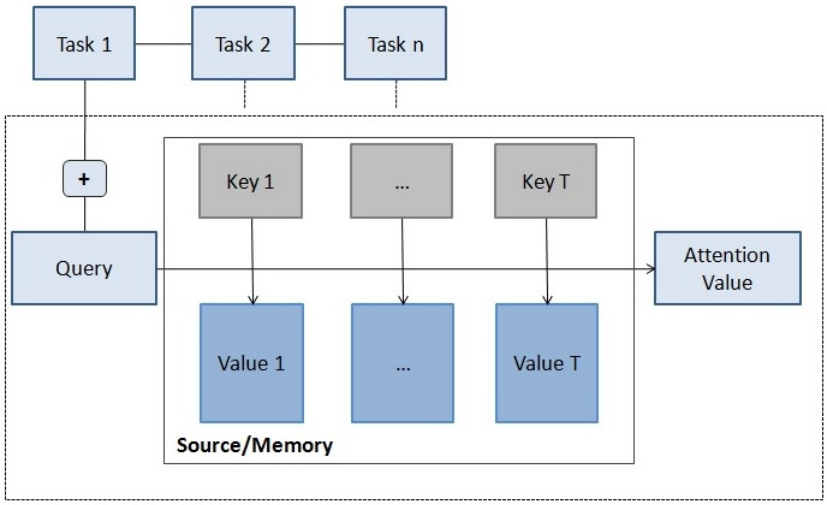

Figure 3. Attention diagram. Attention as a sequential process of reasoning in which the task (query) is guided by a set of elements (values) of the source (or memory).

Then, attention can be seen as a sequential process of reasoning in which the task (query) is guided by a set of elements of the input source (or memory) using attention.

The attention process can focus on:

1. Temporal dimensions, e.g., different time steps of a sequence.
2. Spatial dimensions, e.g., different regions of an image.
3. Different elements of a memory.
4. Different features or dimensions of an input vector, etc.

Depending on where the process is initiated, we have:

1. Top-down attention, initiated by the current task.
2. Bottom-up, initiated spontaneously by the source or memory.

To apply the attention mechanism, it is necessary to break down the learning process into a sequence of attention-guided tasks.

Then, due to its flexibility, an attention mechanism can be added in multiple ways to any deep learning architecture that models a complex system. In Section 4, we illustrate this flexibility as follows:

1. Through a conventional attention (the query is different from the key and the value) in Section 4.2, with the encoder selecting input features and the decoder selecting time steps.
2. Through a memory network in which a memory of historical data guides the current prediction task in Section 4.3.
3. Through self-attention (the keys, values and queries come from the same source) in Section 4.4. Here, to encode a vector of the input sequence, self-attention allows the model to focus in a direct way on other vectors in the sequence.

3.2. Attention in seq2seq Models

An encoder–decoder model maps an input sequence to a target one with both sequences of arbitrary length [3]. They have applications ranging from machine translation to time series prediction.

More specifically, this mechanism uses an RNN (or any of its variants such as an LSTM or a GRU, Gated Recurrent Unit) to map the input sequence to a fixed-length vector, and another or any of its variants (RNN) to decode the target sequence from that vector (see Figure 4). Such a seq2seq model typically features an architecture composed of:

1. An encoder which, given an input sequence $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ with $\boldsymbol{x}_t \in \mathbb{R}^n$, maps $\boldsymbol{x}_t$ to
$$\boldsymbol{h}_t = f_1(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t), \tag{11}$$
where $\boldsymbol{h}_t \in \mathbb{R}^m$ is the hidden state of the encoder at time t, m is the size of the hidden state and f_1 is an or any of its variants (RNN).
2. A decoder, where $\boldsymbol{s}_t$ is the hidden state and whose initial state $\boldsymbol{s}_0$ is initialized with the last hidden state of the encoder $\boldsymbol{h}_T$. It generates the output sequence $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{T'})$, $\boldsymbol{y}_t \in \mathbb{R}^o$ (the dimension o depending on the task), where
$$\boldsymbol{y}_t = f_2(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}), \tag{12}$$
and f_2 is an or any of its variants (RNN) with an additional layer depending on the task (e.g., a linear layer for series prediction or a softmax layer for translation).

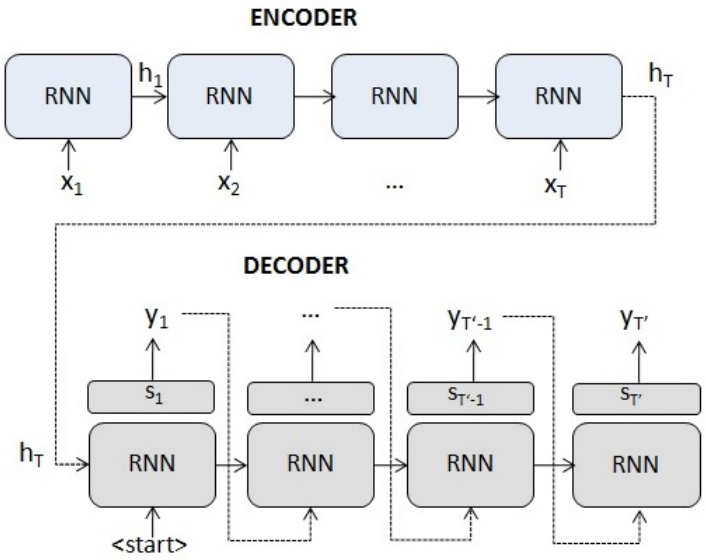

Figure 4. An encoder–decoder network.

Because the encoder compresses all the information of the input sequence in a fixed-length vector (the final hidden state $\boldsymbol{h}_T$), the decoder possibly does not take into account the first elements of the input sequence. The use of this fixed-length vector is a limitation to improve the performance of the encoder–decoder networks. Moreover, the performance of encoder–decoder networks degrades rapidly as the length of the input sequence increases [37]. This occurs in applications such as machine translation and time series prediction, where it is necessary to model long time dependencies.

The key to solve this problem is to use an attention mechanism to guide the decoding task. In [35], an extension of the basic encoder–decoder architecture was proposed by allowing the model to automatically search and learn which parts of a source sequence are relevant to predict the target element. Instead of encoding the input sequence in a fixed-length vector, it generates a sequence of vectors, choosing the most appropriate subset of these vectors during the decoding process.

Equipped with the attention mechanism, the encoder is a bidirectional RNN [38] with a forward hidden state $\overrightarrow{\boldsymbol{h}_i} = f_1(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{x}_i)$ and a backward one $\overleftarrow{\boldsymbol{h}_i} = f_1(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{x}_i)$. The encoder state is represented as a simple concatenation of the two states,

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}_i}], \tag{13}$$

with $i = 1, \ldots, T$. The encoder state includes both the preceding and following elements of the sequence, thus capturing information from neighbouring inputs.

The decoder has an output

$$\boldsymbol{y}_t = f_2(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t) \tag{14}$$

for $t = 1, \ldots, T'$. f_2 is an RNN with an additional layer depending on the task (e.g., a linear layer for series prediction or a softmax layer for translation), and the input is a concatenation of $\boldsymbol{y}_{t-1}$ with the context vector $\boldsymbol{c}_t$, which is a sum of hidden states of the input sequence weighted by alignment scores:

$$\boldsymbol{c}_t = \sum_{i=1}^{T} \alpha_{ti} \boldsymbol{h}_i. \tag{15}$$

Similar to Equation (8), the weight α_{ti} of each state $\boldsymbol{h}_i$ is calculated by

$$\alpha_{ti} = \frac{\exp(score(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{T} \exp(score(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}. \tag{16}$$

In this attention mechanism, the query is the state $\boldsymbol{s}_{t-1}$ and the key and the value are the hidden states $\boldsymbol{h}_i$. The score measures how well the input at position i and the output at position t match. α_{ti} are the weights that implement the attention mechanism, defining how much of each input hidden state should be considered when deciding the next state $\boldsymbol{s}_t$ and generating the output $\boldsymbol{y}_t$ (see Figure 5).

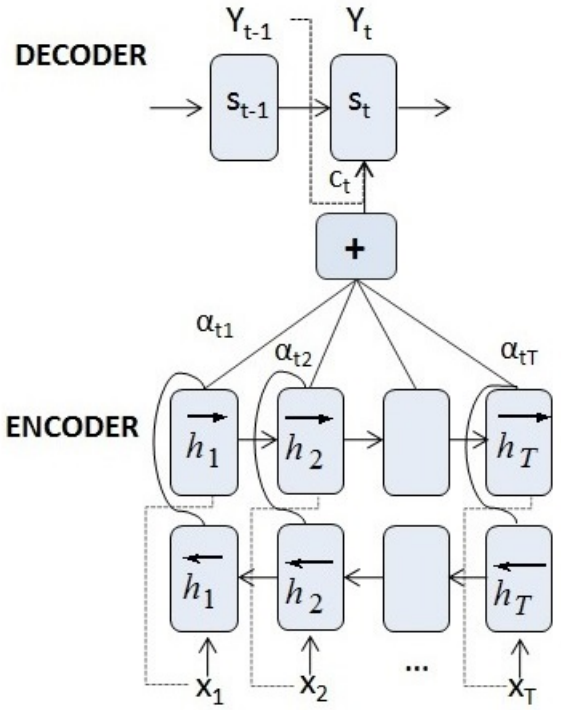

Figure 5. An encoder–decoder network with attention.

As we have described previously, the score function can be parametrized using different alignment models such as feedforward networks and the cosine similarity.

An example of a matrix of alignment scores is shown in Figure 6. This matrix provides interpretability to the model since it allows to know which part (time-step) of the input is more important to the output.

The attention mechanism then transforms an encoder–decoder sequential model into a non-sequential model in which the attention mechanism guides the decoding task based on the encoded states.

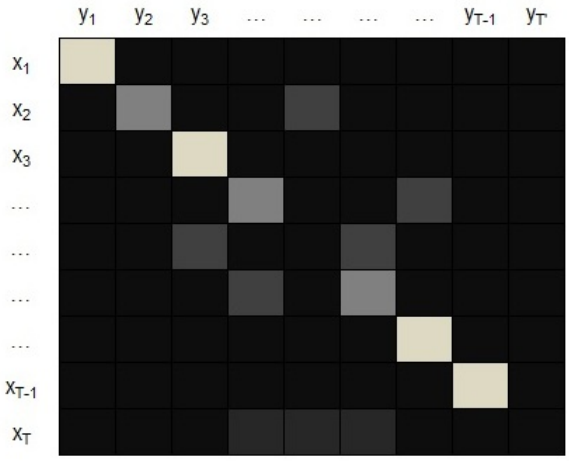

Figure 6. A matrix of alignment scores. It represents how much of each input state should be considered when deciding the next state and generating the output.

3.3. Self-Attention and Memory Networks

A variant of the attention mechanism is self-attention, in which the attention component relates different positions of a single sequence in order to compute a representation of the sequence. In this way, the keys, values and queries come from the same source. The mechanism can connect distant elements of the sequence more directly than using RNNs [12].

Similar to the description given in [11], for an input sequence $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$, the self-attention process can be implemented by the following steps:

1. For each of the input vectors, create a query $\boldsymbol{Q}_t$, a key $\boldsymbol{K}_t$ and a value vector $\boldsymbol{V}_t$ by multiplying the input vector $\boldsymbol{x}_t$ by three matrices that are trained during the learning process, $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$ and $W_i^V \in \mathbb{R}^{d \times d_v}$.
2. For each query vector $\boldsymbol{Q}_t$, the self-attention value is computed by mapping the query and all the key-values to an output, $Attention(\boldsymbol{Q_t}, K, V) = \sum_{j=1}^{T} \alpha_j(\boldsymbol{Q_t}, \boldsymbol{K}_j)\boldsymbol{V}_j$, where

$$\alpha_j(\boldsymbol{Q_t}, \boldsymbol{K}_j) = softmax(\frac{\boldsymbol{Q_t}\boldsymbol{K_j}^T}{\sqrt{d_k}}) \tag{17}$$

3. This self-attention process is performed h times in what is called multi-headed attention. Each time, the input vectors are projected into a different query, key and value vector using different matrices W_i^Q, W_i^K and W_i^V for $i = 1, \ldots, h$. On each of these projected queries, keys and values, the attention function is performed in parallel, producing d_v dimensional output values, that are concatenated and once again projected to the final values. This multi-headed attention process allows the model to focus on different positions from different representation subspaces.

When the model is processing a vector of the input sequence, single self-attention allows the model to focus on other vectors in the sequence to get a better representation of this vector. With multi-headed self-attention (see Figure 7), each attention head is focusing on a different set of vectors when processing the vector.

The transformer [11], a network architecture based only on self-attention, is composed of an encoder and a decoder:

1. Encoder: Composed of a stack of six identical layers, each layer with a multi-head self-attention process and a position-wise fully connected feed-forward network. Around each of the sub-layers, a residual connections followed by layer normalization is employed.

2. Decoder: Is also composed of a stack of six identical layers (with self-attention and a feed-forward network) with an additional third sub-layer to perform attention over the output of the encoder (as in the seq2seq with attention). The self-attention sub-layer is modified to prevent a vector from attending to subsequent vectors in the sequence.

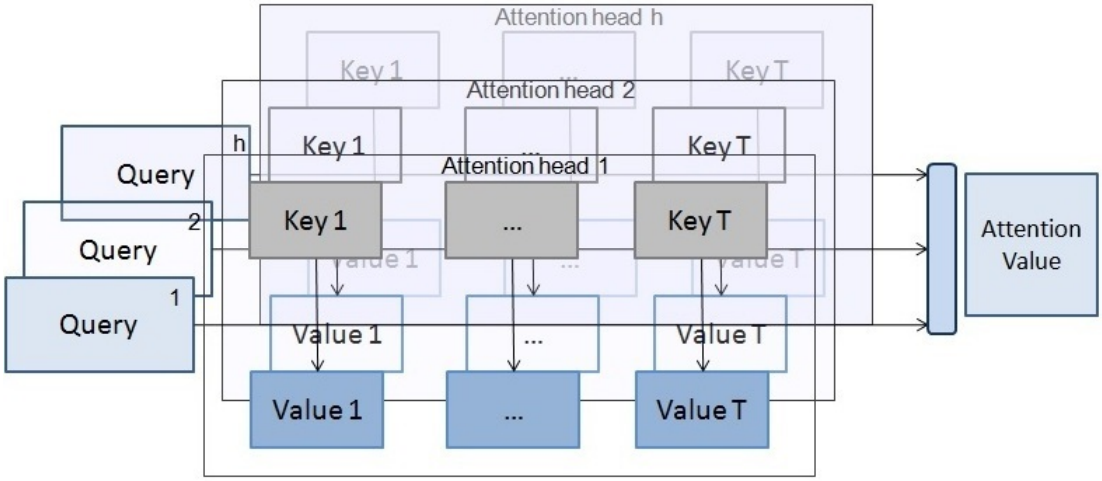

Figure 7. Multi-headed attention. Self-attention process performed in parallel h times in different subspaces. The output values are concatenated and projected to a final value.

The transformer allows to replace CNNs and RNNs, improving machine translation tasks while using less training time. The transformer is also the basic component of GPT-3 (Generative Pre-Trained transformer-3), a pre-trained language model which achieves good performance in few-shot learning on many Natural Language Processing tasks without fine-tuning [39].

Another variant of attention are end-to-end memory networks [40], which are neural networks with a recurrent attention model over an external memory. The model, trained end-to-end, is described in more detail in Section 4.3 and outputs an answer based on a query and a set of inputs $x_1, x_2, \ldots, x_n$ stored in a memory.

4. Attention Mechanisms in Complex Systems

4.1. Where and How to Apply Attention

In the previous sections we have described various attention mechanisms. These mechanisms allow a task to focus on a set of elements of an input sequence, an intermediate sequence or a memory source.

Due to its flexibility, an attention mechanism can be added to any deep learning model in multiple ways. Therefore, when applying it to model complex systems, it will be necessary to decide the following issues:

1. In which part of the model should be introduced?
2. What elements of the model will the attention mechanism relate?
3. What dimension (temporal, spatial, input dimension, etc.) is the mechanism going to focus on?
4. Will self-attention or conventional attention be used?
5. What elements will correspond to the query, the key and the value?

In the following sections we describe some illustrative cases of application of attention mechanisms to model complex systems. As we will see, how the information is processed in the system and how the different elements are related will be key when defining the aforementioned issues.

4.2. Attention in Different Phases of a Model

In a non-autonomous dynamical system, the current state is a transformation of the previous states and the current input, which contains n dimensions or features. More generally, the dependencies between time steps can be dynamic, i.e., time-changing. In such complex systems, attention mechanisms learn to focus on the most relevant parts of the system input or state.

A representative attention mechanism in this context implements a dual-stage attention, namely, an encoder with input features attention and a decoder with temporal attention, as pointed out in [41]. Next we describe this architecture, in which the first stage extracts the relevant input features and the second selects the relevant time steps of the model.

Let $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ with $\boldsymbol{x}_t \in \mathbb{R}^n$ be the input sequence. T is the length of the time interval and n the number of input features or dimensions. $\boldsymbol{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)$ is the input at the time step t and $\boldsymbol{x}^k = (x_1^k, x_2^k, \ldots, x_T^k)$ is the k input feature series.

Encoder with input attention

Given an input sequence X, the encoder maps $\boldsymbol{u}_t$ to

$$\boldsymbol{h}_t = f_1(\boldsymbol{h}_{t-1}, \boldsymbol{u}_t), \tag{18}$$

where $\boldsymbol{h}_t \in \mathbb{R}^m$ is the hidden state of the encoder at time t, m is the size of the hidden state and f_1 is an or any of its variants (RNN). $\boldsymbol{x}_t$ is replaced by $\boldsymbol{u}_t$, which adaptively selects the relevant input features as follows:

$$\boldsymbol{u}_t = (\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, ..., \alpha_t^n x_t^n). \tag{19}$$

Here

$$\alpha_t^k = \frac{\exp(score(\boldsymbol{h}_{t-1}, \boldsymbol{x}^k))}{\sum_{i=1}^{n} \exp(score(\boldsymbol{h}_{t-1}, \boldsymbol{x}^i))}, \tag{20}$$

is the attention weight measuring the importance of the k input feature at time t, where $\boldsymbol{x}^k = (x_1^k, x_2^k, \ldots, x_T^k)$ is the k input feature series and the score function can be computed using a feedforward neural network, a cosine similarity measure or other similarity functions.

Therefore, this first attention stage extracts the relevant input features with the query, keys and values shown in Figure 8.

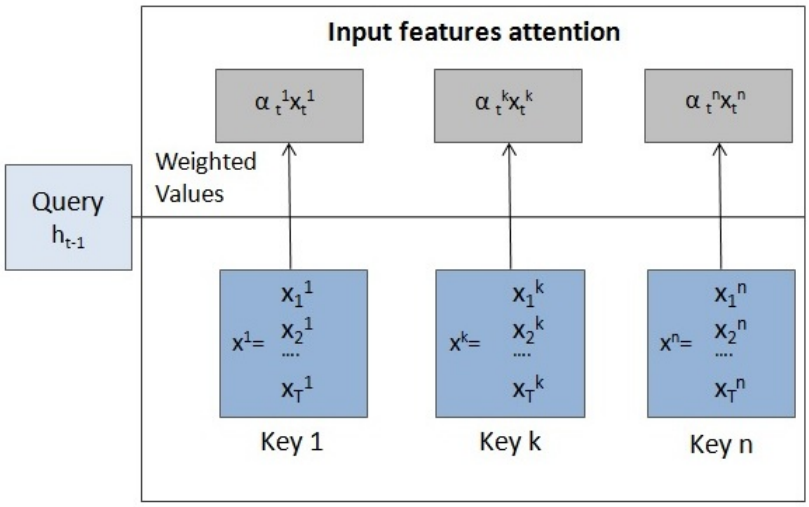

Figure 8. Diagram of the input features attention mechanism.

Decoder with temporal attention

Similar to the attention decoder described in Section 3.2, the decoder has an output

$$\boldsymbol{y}_t = f_2(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t) \tag{21}$$

for $t = 1, \ldots, T'$. f_2 is an or any of its variants (RNN) with an additional linear or softmax layer, and the input is a concatenation of $\boldsymbol{y}_{t-1}$ with the context vector $\boldsymbol{c}_t$, which is a sum of hidden states of the input sequence weighted by alignment scores:

$$\boldsymbol{c}_t = \sum_{i=1}^{T} \beta_t^i \boldsymbol{h}_i. \tag{22}$$

The weight β_t^i of each state $\boldsymbol{h}_i$ is computed using the similarity function, $score(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)$, and applying a softmax function, as described in Section 3.2.

This second attention stage selects the relevant time steps, as shown in Figure 9 with the corresponding query, keys and values.

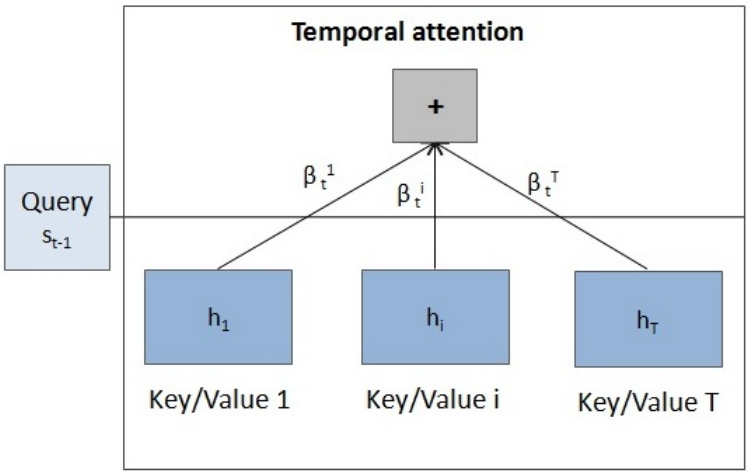

Figure 9. Diagram of the temporal attention mechanism.

Further remarks

In [41], the authors define this dual-stage attention RNN and show that the model outperforms a classical model in time series prediction.

A comparison is made between LSTMs and attention mechanisms for financial time series forecasting in [42]. It is shown that an LSTM with attention performs better than stand-alone LSTMs.

A temporal attention layer is used in [43] to select relevant information and to provide model interpretability, an essential feature to understand deep learning models. In [44], interpretability is further studied in detail, concluding that attention weights partially reflect the impact of the input elements on model prediction.

4.3. Memory Networks

Memory networks allow long-term or external dependencies in sequential data to be learned thanks to an external memory component. Instead of taking into account only the most recent states, memory networks also consider the entire list of states or the states of a memory.

Here we define one possible application of memory networks to complex systems, following an approach based on [40]. We are given a time series of historical data $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_{T'}$ with $\boldsymbol{n}_i \in \mathbb{R}^n$ and the input series $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ with $\boldsymbol{x}_t \in \mathbb{R}^n$ the current input, which is the query in the attenton mechanism.

The set $\{\boldsymbol{n}_i\}$ are converted into memory vectors $\{\boldsymbol{m}_i\}$ and output vectors $\{\boldsymbol{c}_i\}$ of dimension d. The query $\boldsymbol{x}_t$ is also transformed to obtain a internal state $\boldsymbol{u}_t$ of dimension d. These transformations correspond to a linear transformation: $A\boldsymbol{n}_i = \boldsymbol{m}_i, B\boldsymbol{n}_i = \boldsymbol{c}_i, C\boldsymbol{x}_t = \boldsymbol{u}_t$, where A, B, C are parameterizable matrices.

A match between $\boldsymbol{u}_t$ and each memory vector $\boldsymbol{m}_i$ is computed by taking the inner product followed by a softmax function:

$$p_t^i = Softmax(\boldsymbol{u}_t^T \boldsymbol{m}_i). \tag{23}$$

The final vector from the memory, $\boldsymbol{o}_t$, is a weighted sum over the transformed memory inputs $\{\boldsymbol{c}_i\}$:

$$\boldsymbol{o}_t = \sum_i p_t^i \boldsymbol{c}_i. \tag{24}$$

To generate the final prediction $\boldsymbol{y}_t$, a linear layer is applied to the sum of the output vector $\boldsymbol{o}_t$ and the transformed input $\boldsymbol{u}_t$, and to the previous output $\boldsymbol{y}_{t-1}$:

$$\boldsymbol{y}_t = f(W^1(\boldsymbol{o}_t + \boldsymbol{u}_t) + W^2 \boldsymbol{y}_{t-1}) \tag{25}$$

A basic diagram of the model is shown in Figure 10. This model is differentiable end-to-end by learning the matrices (the final matrices W^1 and W^2, and the three transformation matrices A, B and C) to minimize the prediction error.

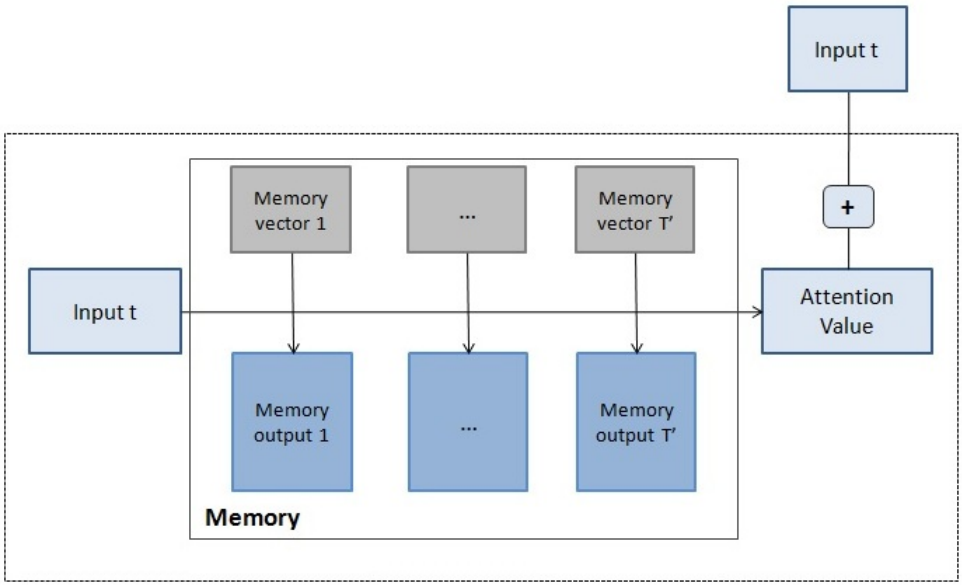

Figure 10. Basic diagram of a memory network. For each input, the attention mechanism integrates a weighted sum over the memory vectors.

In [45], the authors propose a similar model based on memory networks with a memory component, three encoders and an autoregressive component for multivariate time-series forecasting. Compared to non-memory RNN models, their model is better at modeling and capturing long-term dependencies and, moreover, it is interpretable.

Differentiable Neural Computers (DNCs) [46] consist of a neural network that uses attention and can read from, and write to, an external memory. Taking advantage of these capabilities, an enhanced DNC for electroencephalogram (EEG) data analysis is proposed in [47]. By replacing the LSTM network controller with a recurrent convolutional network, the potential of DNCs in EEG signal processing is convincingly demonstrated.

4.4. Self-Attention

An important aspect to model complex systems is to capture the temporal dependence and the relationship between the parts that make up the system.

If we compare the computational graph of an see Figure 2 (RNN) with the graph of an attention module (Figure 11), we observe that even adding a memory unit (LSTM), the attention module relates each of the inputs in a more direct and symmetric way to form the output vector.

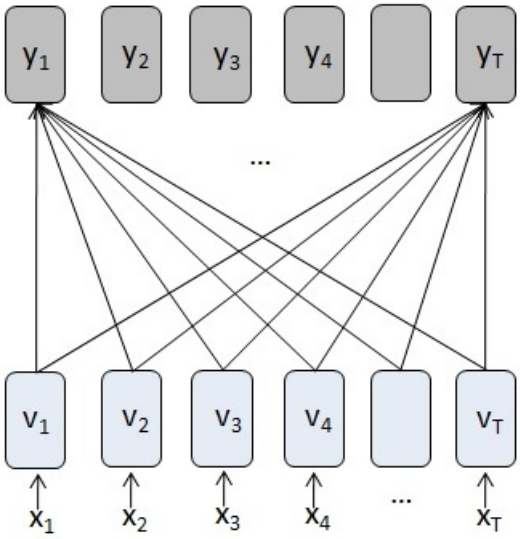

Figure 11. Self-attention graph. The self-attention component calculates how much each input vector contributes to form each output vector.

The distance, in number of edges in the graph, between an input and an output distant in time, is shorter and is the same for all input vectors in the self-attention module. However, this is at the cost of not prioritizing local interactions, which has a high computational cost for very long sequences.

The transformer, as we have pointed out, is composed of a stack of multi-headed self-attention components. With multi-headed attention, the input vectors are projected into a different query, key and value vector, performing the self-attention process h times. When processing a vector, each attention head is focusing on a different set of vectors from different representation subspaces.

These mentioned characteristics make self-attention and the transformer a promising building block in deep learning models for complex systems.

In [48], the authors propose a dual self-attention network for multivariate time (dynamic-period or non-periodic) series forecasting. In [49], the authors utilize attention models for clinical time-series modeling. They employ a masked self-attention mechanism and use positional encoding and dense interpolation for incorporating temporal order.

Further understanding of the transformer architecture is carried out in [50], where the authors show that the transformer architecture can be interpreted as a numerical Ordinary Differential Equation (ODE) solver for a convection-diffusion equation in a multi-particle dynamic system. They interpret how words (vectors) in a sentence are abstracted by passing through the layers of the transformer as approximating the movement of multiple particles in the space using the Lie–Trotter splitting scheme and the Euler's method.

5. Discussion

After the success of recent years, one of the most important challenges that deep learning faces is to improve input-output models, adopting new primitives that provide reasoning, abstraction, search and memory capabilities.

Similar to what happens in the brain, attention mechanisms allow the reasoning or cognitive process to be guided in a flexible way. This improvement is important when modeling complex systems due to their temporal dependence and complex relationships.

As we have seen, attention mechanisms has the following benefits in modeling such systems:

1. By focusing on a subset of elements, it guides the reasoning or cognitive process.
2. These elements can be tensors (vectors) from the input layer, from the intermediate layer or be external to the model, e.g., a external memory.
3. It can focus on temporal dimensions (different time steps of a sequence), spatial dimensions (different regions of space) or different features of an input vector.
4. It can relate each of the input vectors in a more direct and symmetric way to form the output vector.

More specifically, as shown in Table 2, for each of the techniques and applications described, we discuss its application potential, characteristics and advantages.

1. *One stage conventional attention.* The attention mechanism allows guiding any complex system task such as modeling, prediction, identification, etc. To do this, it focuses on a set of elements from the input layer or from an intermediate layer. These elements can be temporal, spatial or feature dimensions. For example, to model a dynamical system with an input of dimension n, one can add an attention mechanism to focus and integrate the different input dimensions. The attention mechanism is combined, as we have seen, with an RNN or an LSTM and allows modeling long temporal dependencies. This technique, like the rest, adds complexity to the model. To calculate the attention weights between a task (query) of T elements and an attended region (key, value) of T elements, it is necessary to perform T^2 multiplications.
2. *Several stages conventional attention*. This case is similar to the previous, one-stage conventional attention but with several attention phases or stages. The attention mechanism is also combined with an RNN or an LSTM and allows modeling long temporal dependencies. As we have seen, the model can focus on a set of feature elements from the input layer and on a set of temporal steps from an intermediate layer. This enables multi-step reasoning. The downside is that more computational cost is added to the model with T^2 multiplications for each attention stage.
3. *Memory networks*. In memory networks, any complex system task such as modeling, prediction or identification is guided by an external memory. Then, memory networks allow long-term or external dependencies in sequential data to be learned thanks to an external memory component. Instead of taking into account only the most recent states, these networks consider the states of a memory or external data as well. Such is the case of time series prediction also based on an external source that can influence the series. To calculate the attention weights between a task (query) of T elements and an attended memory of T' elements, it is necessary to perform TT' multiplications.
4. *Self-attention*. In self-attention, the component relates different positions of a single sequence in order to compute a transformation of the sequence. The keys, values and queries come from the same source. It is a generalization of neural networks, since they perform a direct transformation of the input but the weights are dynamically calculated. The attention module relates each of the inputs in a more direct way to form the output vector but at the cost of not prioritizing local interactions. Their use case is general since they can replace neural networks, RNNs or even CNNs. To calculate the attention weights for a sequence of T elements it is necessary to perform T^2 multiplications.
5. *Combination of the above techniques*. It is interesting to combine several of the previous techniques but at the cost of increasing the complexity and adding the computational cost of each of the components. For example, the transformer, which can be used in a multitude of tasks such as sequence modeling, generative models, predictions, machine translation, multi-tasking, etc. The transformer combines self-attention with conventional attention. In the encoder, the transformer has a stack of self-attention blocks. The decoder also has self-attention blocks and an additional layer to perform attention over the output of the encoder.

Table 2. Characteristics of attention techniques in complex systems described in this paper.

Techniques	Operation	Use Cases	Costs	Applications
One-stage att.	Over the input or an intermediate layer Temporal, spatial ...	Integrate parts Long term dependencies	Complexity T^2 operations	Modeling Prediction
Several stages	Over the input, over an intermediate layer Temporal, spatial ...	Integrate several parts Long term dependencies Multi-step reasoning	Complexity T^2 operations each att. stage	Modeling Prediction Sequential reasoning
Memory networks	Over external data Temporal, spatial ...	Integrate a memory	Complexity TT' operations	Modeling Reasoning over a memory
Self-attention	Relate elements of the same sequence	General Encode an input	Complexity Non local T^2 operations	Replace neural networks
Combination	Combine the above elements	All of the above	Sum of the costs	All of the above

However, despite the theoretical advantages and some achievements, further studies are needed to verify the benefits of the attention mechanisms over traditional networks in complex systems.

Author Contributions: Writing—original draft preparation, A.H.; Writing—review, J.M.A.; Content discussion and final version, A.H. and J.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Spanish Ministry of Science and Innovation, grant PID2019-108654GB-I00.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DNC	Differentiable Neural Computers
CNN	Convolutional neural network
EEG	Electroencephalogram
FNN	Feed-forward neural network
GPT-3	Generative Pre-Trained transformer-3
GPU	Graphics Processing Units
LSTM	Long short-term memory
MDPI	Multidisciplinary Digital Publishing Institute
NARX	Nonlinear Autoregressive model with exogenous input
ODE	Ordinary Differential Equation
RBF	Recurrent radial basis function
RNN	Recurrent Neural Network

References

1. Yadan, O.; Adams, K.; Taigman, Y.; Ranzato, M. Multi-GPU Training of ConvNets. *arXiv* **2013**, arXiv:1312.5853.
2. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
3. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the NIPS 2014, Montreal, QC, Canada, 8–13 December 2014.
4. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.R.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
5. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Available online: http://www.deeplearningbook.org (accessed on 26 February 2021).
6. Chang, B.; Chen, M.; Haber, E.; Chi, E.H. AntisymmetricRNN: A Dynamical System View on Recurrent Neural Networks. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
7. Wang, Z.; Xiao, D.; Fang, F.; Govindan, R.; Pain, C.; Guo, Y. Model identification of reduced order fluid dynamics systems using deep learning. *Int. J. Numer. Methods Fluids* **2018**, *86*, 255–268. [CrossRef]

8. Wang, Y. A new concept using LSTM Neural Networks for dynamic system identification. In Proceedings of the 2017 American Control Conference (ACC), Seattle, WA, USA, 24–26 May 2017; pp. 5324–5329. [CrossRef]
9. Li, Y.; Cao, H. Prediction for Tourism Flow based on LSTM Neural Network. *Procedia Comput. Sci.* **2018**, *129*, 277–283. [CrossRef]
10. Marcus, G. Deep Learning: A Critical Appraisal. *arXiv* **2018**, arXiv:1801.00631.
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the NIPS 2017, Long Beach, CA, USA, 4–9 December 2017.
12. Tang, G.; Müller, M.; Rios, A.; Sennrich, R. Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In Proceedings of the EMNLP 2018, Brussels, Belgium, 31 October–4 November 2018.
13. Hudson, D.A.; Manning, C.D. Compositional Attention Networks for Machine Reasoning. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
14. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
15. Gros, C. *Complex and Adaptive Dynamical Systems. A Primer,* 3rd ed.; Springer: Basel, Switzerland, 2008; Volume 1. [CrossRef]
16. Layek, G. *An Introduction to Dynamical Systems and Chaos*; Springer: Basel, Switzerland, 2015; pp. 1–622. [CrossRef]
17. Arnold, L. *Random Dynamical Systems*; Springer: Berlin/Heidelberg, Germany, 2003.
18. Narendra, K.S.; Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1990**, *1*, 4–27. [CrossRef]
19. Miyoshi, T.; Ichihashi, H.; Okamoto, S.; Hayakawa, T. Learning chaotic dynamics in recurrent RBF network. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 1, pp. 588–593. [CrossRef]
20. Sato, Y.; Nagaya, S. Evolutionary algorithms that generate recurrent neural networks for learning chaos dynamics. In Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996; pp. 144–149. [CrossRef]
21. Diaconescu, E. The use of NARX neural networks to predict chaotic time series. *WSEAS Trans. Comput. Res.* **2008**, *3*, 182–191.
22. Assaad, M.; Boné, R.; Cardot, H. Predicting Chaotic Time Series by Boosted Recurrent Neural Networks. In Proceedings of the International Conference on Neural Information Processing 2006, Hong Kong, China, 3–6 October 2006; Volume 4233, pp. 831–840. [CrossRef]
23. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; doi:10.3115/v1/D14-1179. [CrossRef]
24. Hernández, A.; Amigó, J.M. The Need for More Integration Between Machine Learning and Neuroscience. In *Nonlinear Dynamics, Chaos, and Complexity: In Memory of Professor Valentin Afraimovich*; Springer: Singapore, 2021; pp. 9–19. [CrossRef]
25. Lindsay, G.W. Attention in Psychology, Neuroscience, and Machine Learning. *Front. Comput. Neurosci.* **2020**, *14*, 29. [CrossRef]
26. Deco, G.; Rolls, E. Neurodynamics of Biased Competition and Cooperation for Attention: A Model With Spiking Neurons. *J. Neurophysiol.* **2005**, *94*, 295–313. [CrossRef] [PubMed]
27. Huerta, R.; Vembu, S.; Amigó, J.; Nowotny, T.; Elkan, C. Inhibition in Multiclass Classification. *Neural Comput.* **2012**, *24*, 2473–2507. [CrossRef]
28. Arena, P.; Patané, L.; Termini, P.S. Modeling attentional loop in the insect Mushroom Bodies. In Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, 10–15 June 2012; pp. 1–7.
29. Hernández, A.; Amigó, J.M. Multilayer adaptive networks in neuronal processing. *Eur. Phys. J. Spec. Top.* **2018**, *227*, 1039–1049. [CrossRef]
30. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6077–6086.
31. Gan, Z.; Cheng, Y.; Kholy, A.E.; Li, L.; Liu, J.; Gao, J. Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Florence, Italy, 28 July–2 August 2019.
32. Jetley, S.; Lord, N.; Lee, N.; Torr, P. Learn To Pay Attention. *arXiv* **2018**, arXiv:1804.02391.
33. Hahne, L.; Lüddecke, T.; Wörgötter, F.; Kappel, D. Attention on Abstract Visual Reasoning. *arXiv* **2019**, arXiv:1911.05990.
34. Xiao, T.; Fan, Q.; Gutfreund, D.; Monfort, M.; Oliva, A.; Zhou, B. Reasoning About Human-Object Interactions Through Dual Attention Networks. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 3918–3927.
35. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.
36. Graves, A.; Wayne, G.; Danihelka, I. Neural Turing Machines. *arXiv* **2014**, arXiv:1410.5401.
37. Cho, K.; van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In Proceedings of the SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 103–111. [CrossRef]
38. Graves, A.; Jaitly, N.; Rahman Mohamed, A. Hybrid speech recognition with Deep Bidirectional LSTM. In Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, 8–12 December 2013; pp. 273–278.

39. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
40. Sukhbaatar, S.; Szlam, A.; Weston, J.; Fergus, R. End-To-End Memory Networks. In Proceedings of the NIPS 2015, Montreal, QC, Canada, 7–12 December 2015.
41. Qin, Y.; Song, D.; Cheng, H.; Cheng, W.; Jiang, G.; Cottrell, G.W. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *arXiv* **2017**, arXiv:1704.02971.
42. Hollis, T.; Viscardi, A.; Yi, S.E. A Comparison of LSTMs and Attention Mechanisms for Forecasting Financial Time Series. *arXiv* **2018**, arXiv:1812.07699.
43. Vinayavekhin, P.; Chaudhury, S.; Munawar, A.; Agravante, D.J.; Magistris, G.D.; Kimura, D.; Tachibana, R. Focusing on What is Relevant: Time-Series Learning and Understanding using Attention. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 2624–2629.
44. Serrano, S.; Smith, N.A. Is Attention Interpretable? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), Florence, Italy, 28 July–2 August 2019.
45. Chang, Y.Y.; Sun, F.Y.; Wu, Y.H.; de Lin, S. A Memory-Network Based Solution for Multivariate Time-Series Forecasting. *arXiv* **2018**, arXiv:1809.02105.
46. Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwinska, A.; Colmenarejo, S.G.; Grefenstette, E.; Ramalho, T.; Agapiou, J.; et al. Hybrid computing using a neural network with dynamic external memory. *Nature* **2016**, *538*, 471–476. [CrossRef] [PubMed]
47. Ming, Y.; Pelusi, D.; Fang, C.N.; Prasad, M.; Wang, Y.K.; Wu, D.; Lin, C.T. EEG data analysis with stacked differentiable neural computers. *Neural Comput. Appl.* **2020**, *32*, 7611–7621. [CrossRef]
48. Huang, S.; Wang, D.; Wu, X.; Tang, A. DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2129–2132. [CrossRef]
49. Song, H.; Rajan, D.; Thiagarajan, J.J.; Spanias, A. Attend and Diagnose: Clinical Time Series Analysis using Attention Models. *arXiv* **2017**, arXiv:1711.03905.
50. Lu, Y.; Li, Z.; He, D.; Sun, Z.; Dong, B.; Qin, T.; Wang, L.; Liu, T. Understanding and Improving transformer from a Multi-Particle Dynamic System Point of View. *arXiv* **2019**, arXiv:1906.02762.