Title: Recurrent neural network

URL: https://en.wikipedia.org/wiki/Recurrent_neural_network

PageID: 1706303

Categories: Category:Neural network architectures

Supervised learning

Unsupervised learning

Semi-supervised learning

Self-supervised learning

Reinforcement learning

Meta-learning

Online learning

Batch learning

Curriculum learning

Rule-based learning

Neuro-symbolic AI

Neuromorphic engineering

Quantum machine learning

Classification

Generative modeling

Regression

Clustering

Dimensionality reduction

Density estimation

Anomaly detection

Data cleaning

AutoML

Association rules

Semantic analysis

Structured prediction

Feature engineering

Feature learning

Learning to rank

Grammar induction

Ontology learning

Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

In artificial neural networks , recurrent neural networks (RNNs) are designed for processing sequential data, such as text, speech, and time series , [1] where the order of elements is important. Unlike feedforward neural networks , which process inputs independently, RNNs utilize recurrent connections, where the output of a neuron at one time step is fed back as input to the network at the next time step. This enables RNNs to capture temporal dependencies and patterns within sequences.

The fundamental building block of RNN is the recurrent unit , which maintains a hidden state —a form of memory that is updated at each time step based on the current input and the previous hidden state. This feedback mechanism allows the network to learn from past inputs and incorporate that knowledge into its current processing. RNNs have been successfully applied to

tasks such as unsegmented, connected handwriting recognition , [2] speech recognition , [3] [4] natural language processing , and neural machine translation . [5] [6]

However, traditional RNNs suffer from the vanishing gradient problem , which limits their ability to learn long-range dependencies. This issue was addressed by the development of the long short-term memory (LSTM) architecture in 1997, making it the standard RNN variant for handling long-term dependencies. Later, gated recurrent units (GRUs) were introduced as a more computationally efficient alternative.

In recent years, transformers , which rely on self-attention mechanisms instead of recurrence, have become the dominant architecture for many sequence-processing tasks, particularly in natural language processing, due to their superior handling of long-range dependencies and greater parallelizability. Nevertheless, RNNs remain relevant for applications where computational efficiency, real-time processing, or the inherent sequential nature of data is crucial.

History

Before modern

One origin of RNN was neuroscience. The word "recurrent" is used to describe loop-like structures in anatomy . In 1901, Cajal observed "recurrent semicircles" in the cerebellar cortex formed by parallel fiber , Purkinje cells , and granule cells . [7] [8] In 1933, Lorente de Nó discovered "recurrent, reciprocal connections" by Golgi's method , and proposed that excitatory loops explain certain aspects of the vestibulo-ocular reflex . [9] [10] During 1940s, multiple people proposed the existence of feedback in the brain, which was a contrast to the previous understanding of the neural system as a purely feedforward structure. Hebb considered "reverberating circuit" as an explanation for short-term memory. [11] The McCulloch and Pitts paper (1943), which proposed the McCulloch-Pitts neuron model, considered networks that contains cycles. The current activity of such networks can be affected by activity indefinitely far in the past. [12] They were both interested in closed loops as possible explanations for e.g. epilepsy and causalgia . [13] [14] Recurrent inhibition was proposed in 1946 as a negative feedback mechanism in motor control. Neural feedback loops were a common topic of discussion at the Macy conferences . [15] See [16] for an extensive review of recurrent neural network models in neuroscience.

Frank Rosenblatt in 1960 published "close-loop cross-coupled perceptrons", which are 3-layered perceptron networks whose middle layer contains recurrent connections that change by a Hebbian learning rule. [18] : 73–75 Later, in Principles of Neurodynamics (1961), he described "closed-loop cross-coupled" and "back-coupled" perceptron networks, and made theoretical and experimental studies for Hebbian learning in these networks, [17] : Chapter 19, 21 and noted that a fully cross-coupled perceptron network is equivalent to an infinitely deep feedforward network. [17] : Section 19.11

Similar networks were published by Kaoru Nakano in 1971, [19] [20] Shun'ichi Amari in 1972, [21] and William A. Little [de] in 1974, [22] who was acknowledged by Hopfield in his 1982 paper.

Another origin of RNN was statistical mechanics . The Ising model was developed by Wilhelm Lenz [23] and Ernst Ising [24] in the 1920s [25] as a simple statistical mechanical model of magnets at equilibrium. Glauber in 1963 studied the Ising model evolving in time, as a process towards equilibrium (Glauber dynamics), adding in the component of time. [26]

The Sherrington–Kirkpatrick model of spin glass, published in 1975, [27] is the Hopfield network with random initialization. Sherrington and Kirkpatrick found that it is highly likely for the energy function of the SK model to have many local minima. In the 1982 paper, Hopfield applied this recently developed theory to study the Hopfield network with binary activation functions. [28] In a 1984 paper he extended this to continuous activation functions. [29] It became a standard model for the study of neural networks through statistical mechanics. [30] [31]

Modern

Modern RNN networks are mainly based on two architectures: LSTM and BRNN. [32]

At the resurgence of neural networks in the 1980s, recurrent networks were studied again. They were sometimes called "iterated nets". [33] Two early influential works were the Jordan network

(1986) and the Elman network (1990), which applied RNN to study cognitive psychology . In 1993, a neural history compressor system solved a "Very Deep Learning" task that required more than 1000 subsequent layers in an RNN unfolded in time. [34]

Long short-term memory (LSTM) networks were invented by Hochreiter and Schmidhuber in 1995 and set accuracy records in multiple applications domains. [35] [36] It became the default choice for RNN architecture.

Bidirectional recurrent neural networks (BRNN) use two RNNs that process the same input in opposite directions. [37] These two are often combined, giving the bidirectional LSTM architecture.

Around 2006, bidirectional LSTM started to revolutionize speech recognition , outperforming traditional models in certain speech applications. [38] [39] They also improved large-vocabulary speech recognition [3] [4] and text-to-speech synthesis [40] and was used in Google voice search , and dictation on Android devices . [41] They broke records for improved machine translation , [42] language modeling [43] and Multilingual Language Processing. [44] Also, LSTM combined with convolutional neural networks (CNNs) improved automatic image captioning . [45]

The idea of encoder-decoder sequence transduction had been developed in the early 2010s. The papers most commonly cited as the originators that produced seq2seq are two papers from 2014. [46] [47] A seq2seq architecture employs two RNN, typically LSTM, an "encoder" and a "decoder", for sequence transduction, such as machine translation. They became state of the art in machine translation, and was instrumental in the development of attention mechanisms and transformers .

Configurations

An RNN-based model can be factored into two parts: configuration and architecture. Multiple RNNs can be combined in a data flow, and the data flow itself is the configuration. Each RNN itself may have any architecture, including LSTM, GRU, etc.

Standard

RNNs come in many variants. Abstractly speaking, an RNN is a function f θ {\displaystyle f_{\theta }} of type (x t , h t) ■ (y t , h t + 1) {\displaystyle (x_{t},h_{t})\mapsto (y_{t},h_{t+1})} , where

x t {\displaystyle x_{t}} : input vector;

h t {\displaystyle h_{t}} : hidden vector;

y t {\displaystyle y_{t}} : output vector;

θ {\displaystyle \theta } : neural network parameters.

In words, it is a neural network that maps an input x t {\displaystyle x_{t}} into an output y t {\displaystyle y_{t}} , with the hidden vector h t {\displaystyle h_{t}} playing the role of "memory", a partial record of all previous input-output pairs. At each step, it transforms input to an output, and modifies its "memory" to help it to better perform future processing.

The illustration to the right may be misleading to many because practical neural network topologies are frequently organized in "layers" and the drawing gives that appearance. However, what appears to be layers are, in fact, different steps in time, "unfolded" to produce the appearance of layers .

Stacked RNN

A stacked RNN , or deep RNN , is composed of multiple RNNs stacked one above the other. Abstractly, it is structured as follows

Layer 1 has hidden vector h 1 , t {\displaystyle h_{1,t}} , parameters θ 1 {\displaystyle \theta _{1}} , and maps f θ 1 : (x 0 , t , h 1 , t) ■ (x 1 , t , h 1 , t + 1) {\displaystyle f_{\theta _{1}}:(x_{0,t},h_{1,t})\mapsto (x_{1,t},h_{1,t+1})} .

Layer 2 has hidden vector h 2 , t {\displaystyle h_{2,t}} , parameters θ 2 {\displaystyle \theta _{2}} , and maps f θ 2 : (x 1 , t , h 2 , t) ■ (x 2 , t , h 2 , t + 1) {\displaystyle f_{\theta _{2}}:(x_{1,t},h_{2,t})\mapsto (x_{2,t},h_{2,t+1})} .

...

Layer n {\displaystyle n} has hidden vector h n , t {\displaystyle h_{n,t}} , parameters θ n {\displaystyle \theta _{n}} , and maps f θ n : (x n – 1 , t , h n , t) ■ (x n , t , h n , t + 1) {\displaystyle f_{\theta _{n}}:(x_{n-1,t},h_{n,t})\mapsto (x_{n,t},h_{n,t+1})} .

Each layer operates as a stand-alone RNN, and each layer's output sequence is used as the input sequence to the layer above. There is no conceptual limit to the depth of stacked RNN.

Bidirectional

A bidirectional RNN (biRNN) is composed of two RNNs, one processing the input sequence in one direction, and another in the opposite direction. Abstractly, it is structured as follows:

The forward RNN processes in one direction: f θ (x 0 , h 0) = (y 0 , h 1) , f θ (x 1 , h 1) = (y 1 , h 2) , … {\displaystyle f_{\theta }(x_{0},h_{0})=(y_{0},h_{1}),f_{\theta }(x_{1},h_{1})=(y_{1},h_{2}),\dots }

The backward RNN processes in the opposite direction: f θ ′ ′ (x N , h N ′) = (y N ′ , h N – 1 ′) , f θ ′ ′ (x N – 1 , h N – 1 ′) = (y N – 1 ′ , h N – 2 ′) , … {\displaystyle f'_{\theta '}(x_{N},h_{N}')=(y'_{N},h_{N-1}'),f'_{\theta '}(x_{N-1},h_{N-1}')=(y'_{N-1},h_{N-2}'),\dots }

The two output sequences are then concatenated to give the total output: ((y 0 , y 0 ′) , (y 1 , y 1 ′) , … , (y N , y N ′)) {\displaystyle ((y_{0},y_{0}'),(y_{1},y_{1}'),\dots ,(y_{N},y_{N}'))} .

Bidirectional RNN allows the model to process a token both in the context of what came before it and what came after it. By stacking multiple bidirectional RNNs together, the model can process a token increasingly contextually. The ELMo model (2018) [48] is a stacked bidirectional LSTM which takes character-level as inputs and produces word-level embeddings.

Encoder-decoder

Two RNNs can be run front-to-back in an encoder-decoder configuration. The encoder RNN processes an input sequence into a sequence of hidden vectors, and the decoder RNN processes the sequence of hidden vectors to an output sequence, with an optional attention mechanism . This was used to construct state of the art neural machine translators during the 2014–2017 period. This was an instrumental step towards the development of transformers . [49]

PixelRNN

An RNN may process data with more than one dimension. PixelRNN processes two-dimensional data, with many possible directions. [50] For example, the row-by-row direction processes an n × n {\displaystyle n\times n} grid of vectors x i , j {\displaystyle x_{i,j}} in the following order: x 1 , 1 , x 1 , 2 , … , x 1 , n , x 2 , 1 , x 2 , 2 , … , x 2 , n , … , x n , n {\displaystyle x_{1,1},x_{1,2},\dots ,x_{1,n},x_{2,1},x_{2,2},\dots ,x_{2,n},\dots ,x_{n,n}} The diagonal BiLSTM uses two LSTMs to process the same grid. One processes it from the top-left corner to the bottom-right, such that it processes x i , j {\displaystyle x_{i,j}} depending on its hidden state and cell state on the top and the left side: h i – 1 , j , c i – 1 , j {\displaystyle h_{i-1,j},c_{i-1,j}} and h i , j – 1 , c i , j – 1 {\displaystyle h_{i,j-1},c_{i,j-1}} . The other processes it from the top-right corner to the bottom-left.

Architectures

Fully recurrent

Fully recurrent neural networks (FRNN) connect the outputs of all neurons to the inputs of all neurons. In other words, it is a fully connected network . This is the most general neural network topology, because all other topologies can be represented by setting some connection weights to zero to simulate the lack of connections between those neurons.

Hopfield

The Hopfield network is an RNN in which all connections across layers are equally sized. It requires stationary inputs and is thus not a general RNN, as it does not process sequences of patterns. However, it guarantees that it will converge. If the connections are trained using Hebbian learning , then the Hopfield network can perform as robust content-addressable memory , resistant to

connection alteration.

Elman networks and Jordan networks

An Elman network is a three-layer network (arranged horizontally as x , y , and z in the illustration) with the addition of a set of context units (u in the illustration). The middle (hidden) layer is connected to these context units fixed with a weight of one. [51] At each time step, the input is fed forward and a learning rule is applied. The fixed back-connections save a copy of the previous values of the hidden units in the context units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform tasks such as sequence-prediction that are beyond the power of a standard multilayer perceptron .

Jordan networks are similar to Elman networks. The context units are fed from the output layer instead of the hidden layer. The context units in a Jordan network are also called the state layer. They have a recurrent connection to themselves. [51]

Elman and Jordan networks are also known as "Simple recurrent networks" (SRN).

Variables and functions

x t {\displaystyle x_{t}} : input vector

h t {\displaystyle h_{t}} : hidden layer vector

s t {\displaystyle s_{t}} : "state" vector,

y t {\displaystyle y_{t}} : output vector

W {\displaystyle W} , U {\displaystyle U} and b {\displaystyle b} : parameter matrices and vector

σ {\displaystyle \sigma } : Activation functions

Long short-term memory

Long short-term memory (LSTM) is the most widely used RNN architecture. It was designed to solve the vanishing gradient problem . LSTM is normally augmented by recurrent gates called "forget gates". [54] LSTM prevents backpropagated errors from vanishing or exploding. [55] Instead, errors can flow backward through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier. Problem-specific LSTM-like topologies can be evolved. [56] LSTM works even given long delays between significant events and can handle signals that mix low and high-frequency components.

Many applications use stacks of LSTMs, [57] for which it is called "deep LSTM". LSTM can learn to recognize context-sensitive languages unlike previous models based on hidden Markov models (HMM) and similar concepts. [58]

Gated recurrent unit

Gated recurrent unit (GRU), introduced in 2014, was designed as a simplification of LSTM. They are used in the full form and several further simplified variants. [59] [60] They have fewer parameters than LSTM, as they lack an output gate. [61]

Their performance on polyphonic music modeling and speech signal modeling was found to be similar to that of long short-term memory. [62] There does not appear to be particular performance difference between LSTM and GRU. [62] [63]

Bidirectional associative memory

Introduced by Bart Kosko , [64] a bidirectional associative memory (BAM) network is a variant of a Hopfield network that stores associative data as a vector. The bidirectionality comes from passing information through a matrix and its transpose . Typically, bipolar encoding is preferred to binary encoding of the associative pairs. Recently, stochastic BAM models using Markov stepping are optimized for increased network stability and relevance to real-world applications. [65]

A BAM network has two layers, either of which can be driven as an input to recall an association and produce an output on the other layer. [66]

Echo state

Echo state networks (ESN) have a sparsely connected random hidden layer. The weights of output neurons are the only part of the network that can change (be trained). ESNs are good at reproducing certain time series . [67] A variant for spiking neurons is known as a liquid state machine . [68]

Recursive

A recursive neural network [69] is created by applying the same set of weights recursively over a differentiable graph-like structure by traversing the structure in topological order . Such networks are typically also trained by the reverse mode of automatic differentiation . [70] [71] They can process distributed representations of structure, such as logical terms . A special case of recursive neural networks is the RNN whose structure corresponds to a linear chain. Recursive neural networks have been applied to natural language processing . [72] The recursive neural tensor network uses a tensor -based composition function for all nodes in the tree. [73]

Neural Turing machines

Neural Turing machines (NTMs) are a method of extending recurrent neural networks by coupling them to external memory resources with which they interact. The combined system is analogous to a Turing machine or Von Neumann architecture but is differentiable end-to-end, allowing it to be efficiently trained with gradient descent . [74]

Differentiable neural computers (DNCs) are an extension of neural Turing machines, allowing for the usage of fuzzy amounts of each memory address and a record of chronology. [75]

Neural network pushdown automata (NNPDA) are similar to NTMs, but tapes are replaced by analog stacks that are differentiable and trained. In this way, they are similar in complexity to recognizers of context free grammars (CFGs). [76]

Recurrent neural networks are Turing complete and can run arbitrary programs to process arbitrary sequences of inputs. [77]

Training

Teacher forcing

An RNN can be trained into a conditionally generative model of sequences, aka autoregression .

Concretely, let us consider the problem of machine translation, that is, given a sequence (x 1 , x 2 , … , x n) {\displaystyle (x_{1},x_{2},\dots ,x_{n})} of English words, the model is to produce a sequence (y 1 , … , y m) {\displaystyle (y_{1},\dots ,y_{m})} of French words. It is to be solved by a seq2seq model.

Now, during training, the encoder half of the model would first ingest (x 1 , x 2 , … , x n) {\displaystyle (x_{1},x_{2},\dots ,x_{n})} , then the decoder half would start generating a sequence (y ^ 1 , y ^ 2 , … , y ^ l) {\displaystyle ({\hat {y}}_{1},{\hat {y}}_{2},\dots ,{\hat {y}}_{l})} . The problem is that if the model makes a mistake early on, say at y ^ 2 {\displaystyle {\hat {y}}_{2}} , then subsequent tokens are likely to also be mistakes. This makes it inefficient for the model to obtain a learning signal, since the model would mostly learn to shift y ^ 2 {\displaystyle {\hat {y}}_{2}} towards y 2 {\displaystyle y_{2}} , but not the others.

Teacher forcing makes it so that the decoder uses the correct output sequence for generating the next entry in the sequence. So for example, it would see (y 1 , … , y k) {\displaystyle (y_{1},\dots ,y_{k})} in order to generate y ^ k + 1 {\displaystyle {\hat {y}}_{k+1}} .

Gradient descent

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. In neural networks, it can be used to minimize the error term by changing each weight in proportion to the derivative of the error with respect to that weight, provided the non-linear activation functions are differentiable .

The standard method for training RNN by gradient descent is the " backpropagation through time " (BPTT) algorithm, which is a special case of the general algorithm of backpropagation . A more computationally expensive online variant is called "Real-Time Recurrent Learning" or RTRL, [78] [79] which is an instance of automatic differentiation in the forward accumulation mode with stacked tangent vectors. Unlike BPTT, this algorithm is local in time but not local in space.

In this context, local in space means that a unit's weight vector can be updated using only information stored in the connected units and the unit itself such that update complexity of a single unit is linear in the dimensionality of the weight vector. Local in time means that the updates take place continually (on-line) and depend only on the most recent time step rather than on multiple time steps within a given time horizon as in BPTT. Biological neural networks appear to be local with respect to both time and space. [80] [81]

For recursively computing the partial derivatives, RTRL has a time-complexity of O(number of hidden x number of weights) per time step for computing the Jacobian matrices , while BPTT only takes O(number of weights) per time step, at the cost of storing all forward activations within the given time horizon. [82] An online hybrid between BPTT and RTRL with intermediate complexity exists, [83] [84] along with variants for continuous time. [85]

A major problem with gradient descent for standard RNN architectures is that error gradients vanish exponentially quickly with the size of the time lag between important events. [55] [86] LSTM combined with a BPTT/RTRL hybrid learning method attempts to overcome these problems. [36] This problem is also solved in the independently recurrent neural network (IndRNN) [87] by reducing the context of a neuron to its own past state and the cross-neuron information can then be explored in the following layers. Memories of different ranges including long-term memory can be learned without the gradient vanishing and exploding problems.

The online algorithm called causal recursive backpropagation (CRBP), implements and combines BPTT and RTRL paradigms for locally recurrent networks. [88] It works with the most general locally recurrent networks. The CRBP algorithm can minimize the global error term. This fact improves the stability of the algorithm, providing a unifying view of gradient calculation techniques for recurrent networks with local feedback.

One approach to gradient information computation in RNNs with arbitrary architectures is based on signal-flow graphs diagrammatic derivation. [89] It uses the BPTT batch algorithm, based on Lee's theorem for network sensitivity calculations. [90] It was proposed by Wan and Beaufays, while its fast online version was proposed by Campolucci, Uncini and Piazza. [90]

Connectionist temporal classification

The connectionist temporal classification (CTC) [91] is a specialized loss function for training RNNs for sequence modeling problems where the timing is variable. [92]

Global optimization methods

Training the weights in a neural network can be modeled as a non-linear global optimization problem. A target function can be formed to evaluate the fitness or error of a particular weight vector as follows: First, the weights in the network are set according to the weight vector. Next, the network is evaluated against the training sequence. Typically, the sum-squared difference between the predictions and the target values specified in the training sequence is used to represent the error of the current weight vector. Arbitrary global optimization techniques may then be used to minimize this target function.

The most common global optimization method for training RNNs is genetic algorithms , especially in unstructured networks. [93] [94] [95]

Initially, the genetic algorithm is encoded with the neural network weights in a predefined manner where one gene in the chromosome represents one weight link. The whole network is represented as a single chromosome. The fitness function is evaluated as follows:

Each weight encoded in the chromosome is assigned to the respective weight link of the network.

The training set is presented to the network which propagates the input signals forward.

The mean-squared error is returned to the fitness function.

This function drives the genetic selection process.

Many chromosomes make up the population; therefore, many different neural networks are evolved until a stopping criterion is satisfied. A common stopping scheme can be:

When the neural network has learned a certain percentage of the training data.

When the minimum value of the mean-squared-error is satisfied.

When the maximum number of training generations has been reached.

The fitness function evaluates the stopping criterion as it receives the mean-squared error reciprocal from each network during training. Therefore, the goal of the genetic algorithm is to maximize the fitness function, reducing the mean-squared error.

Other global (and/or evolutionary) optimization techniques may be used to seek a good set of weights, such as simulated annealing or particle swarm optimization .

Other architectures

Independently RNN (IndRNN)

The independently recurrent neural network (IndRNN) [87] addresses the gradient vanishing and exploding problems in the traditional fully connected RNN. Each neuron in one layer only receives its own past state as context information (instead of full connectivity to all other neurons in this layer) and thus neurons are independent of each other's history. The gradient backpropagation can be regulated to avoid gradient vanishing and exploding in order to keep long or short-term memory. The cross-neuron information is explored in the next layers. IndRNN can be robustly trained with non-saturated nonlinear functions such as ReLU . Deep networks can be trained using skip connections .

Neural history compressor

The neural history compressor is an unsupervised stack of RNNs. [96] At the input level, it learns to predict its next input from the previous inputs. Only unpredictable inputs of some RNN in the hierarchy become inputs to the next higher level RNN, which therefore recomputes its internal state only rarely. Each higher level RNN thus studies a compressed representation of the information in the RNN below. This is done such that the input sequence can be precisely reconstructed from the representation at the highest level.

The system effectively minimizes the description length or the negative logarithm of the probability of the data. [97] Given a lot of learnable predictability in the incoming data sequence, the highest level RNN can use supervised learning to easily classify even deep sequences with long intervals between important events.

It is possible to distill the RNN hierarchy into two RNNs: the "conscious" chunker (higher level) and the "subconscious" automatizer (lower level). [96] Once the chunker has learned to predict and compress inputs that are unpredictable by the automatizer, then the automatizer can be forced in the next learning phase to predict or imitate through additional units the hidden units of the more slowly changing chunker. This makes it easy for the automatizer to learn appropriate, rarely changing memories across long intervals. In turn, this helps the automatizer to make many of its once unpredictable inputs predictable, such that the chunker can focus on the remaining unpredictable events. [96]

A generative model partially overcame the vanishing gradient problem [55] of automatic differentiation or backpropagation in neural networks in 1992. In 1993, such a system solved a "Very Deep Learning" task that required more than 1000 subsequent layers in an RNN unfolded in time. [34]

Second order RNNs

Second-order RNNs use higher order weights w i j k {\displaystyle w{}_{ijk}} instead of the standard w i j {\displaystyle w{}_{ij}} weights, and states can be a product. This allows a direct mapping to a

finite-state machine both in training, and representation. [98] [99] Long short-term memory is an example of this but has no such formal mappings or proof of stability.

Hierarchical recurrent neural network

Hierarchical recurrent neural networks (HRNN) connect their neurons in various ways to decompose hierarchical behavior into useful subprograms. [96] [100] Such hierarchical structures of cognition are present in theories of memory presented by philosopher Henri Bergson , whose philosophical views have inspired hierarchical models. [101]

Hierarchical recurrent neural networks are useful in forecasting , helping to predict disaggregated inflation components of the consumer price index (CPI). The HRNN model leverages information from higher levels in the CPI hierarchy to enhance lower-level predictions. Evaluation of a substantial dataset from the US CPI-U index demonstrates the superior performance of the HRNN model compared to various established inflation prediction methods. [102]

Recurrent multilayer perceptron network

Generally, a recurrent multilayer perceptron network (RMLP network) consists of cascaded subnetworks, each containing multiple layers of nodes. Each subnetwork is feed-forward except for the last layer, which can have feedback connections. Each of these subnets is connected only by feed-forward connections. [103]

Multiple timescales model

A multiple timescales recurrent neural network (MTRNN) is a neural-based computational model that can simulate the functional hierarchy of the brain through self-organization depending on the spatial connection between neurons and on distinct types of neuron activities, each with distinct time properties. [104] [105] With such varied neuronal activities, continuous sequences of any set of behaviors are segmented into reusable primitives, which in turn are flexibly integrated into diverse sequential behaviors. The biological approval of such a type of hierarchy was discussed in the memory-prediction theory of brain function by Hawkins in his book On Intelligence . [citation needed] Such a hierarchy also agrees with theories of memory posited by philosopher Henri Bergson , which have been incorporated into an MTRNN model. [101] [106]

Memristive networks

Greg Snider of HP Labs describes a system of cortical computing with memristive nanodevices. [107] The memristors (memory resistors) are implemented by thin film materials in which the resistance is electrically tuned via the transport of ions or oxygen vacancies within the film. DARPA 's SyNAPSE project has funded IBM Research and HP Labs, in collaboration with the Boston University Department of Cognitive and Neural Systems (CNS), to develop neuromorphic architectures that may be based on memristive systems. Memristive networks are a particular type of physical neural network that have very similar properties to (Little-)Hopfield networks, as they have continuous dynamics, a limited memory capacity and natural relaxation via the minimization of a function which is asymptotic to the Ising model . In this sense, the dynamics of a memristive circuit have the advantage compared to a Resistor-Capacitor network to have a more interesting non-linear behavior. From this point of view, engineering analog memristive networks account for a peculiar type of neuromorphic engineering in which the device behavior depends on the circuit wiring or topology.

The evolution of these networks can be studied analytically using variations of the Caravelli-Traversa-Di Ventra equation . [108]

Continuous-time

A continuous-time recurrent neural network (CTRNN) uses a system of ordinary differential equations to model the effects on a neuron of the incoming inputs. They are typically analyzed by dynamical systems theory . Many RNN models in neuroscience are continuous-time. [16]

For a neuron i {\displaystyle i} in the network with activation y i {\displaystyle y_{i}} , the rate of change of activation is given by:

Where:

τ i {\displaystyle \tau _{i}} : Time constant of postsynaptic node

y i {\displaystyle y_{i}} : Activation of postsynaptic node

y ■ i {\displaystyle {\dot {y}}_{i}} : Rate of change of activation of postsynaptic node

w j i {\displaystyle w{}_{ji}} : Weight of connection from pre to postsynaptic node

σ (x) {\displaystyle \sigma (x)} : Sigmoid of x e.g. σ (x) = 1 / (1 + e – x) {\displaystyle \sigma (x)=1/(1+e^{-x})} .

y j {\displaystyle y_{j}} : Activation of presynaptic node

Θ j {\displaystyle \Theta _{j}} : Bias of presynaptic node

I i (t) {\displaystyle I_{i}(t)} : Input (if any) to node

CTRNNs have been applied to evolutionary robotics where they have been used to address vision, [109] co-operation, [110] and minimal cognitive behaviour. [111]

Note that, by the Shannon sampling theorem , discrete-time recurrent neural networks can be viewed as continuous-time recurrent neural networks where the differential equations have transformed into equivalent difference equations . [112] This transformation can be thought of as occurring after the post-synaptic node activation functions y i (t) {\displaystyle y_{i}(t)} have been low-pass filtered but prior to sampling.

They are in fact recursive neural networks with a particular structure: that of a linear chain. Whereas recursive neural networks operate on any hierarchical structure, combining child representations into parent representations, recurrent neural networks operate on the linear progression of time, combining the previous time step and a hidden representation into the representation for the current time step.

From a time-series perspective, RNNs can appear as nonlinear versions of finite impulse response and infinite impulse response filters and also as a nonlinear autoregressive exogenous model (NARX). [113] RNN has infinite impulse response whereas convolutional neural network has finite impulse response . Both classes of networks exhibit temporal dynamic behavior . [114] A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

The effect of memory-based learning for the recognition of sequences can also be implemented by a more biological-based model which uses the silencing mechanism exhibited in neurons with a relatively high frequency spiking activity . [115]

Additional stored states and the storage under direct control by the network can be added to both infinite-impulse and finite-impulse networks. Another network or graph can also replace the storage if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated states or gated memory and are part of long short-term memory networks (LSTMs) and gated recurrent units . This is also called Feedback Neural Network (FNN).

Libraries

Modern libraries provide runtime-optimized implementations of the above functionality or allow to speed up the slow loop by just-in-time compilation .

Apache Singa

Caffe : Created by the Berkeley Vision and Learning Center (BVLC). It supports both CPU and GPU. Developed in C++ , and has Python and MATLAB wrappers.

Chainer : Fully in Python, production support for CPU, GPU, distributed training.

Deeplearning4j : Deep learning in Java and Scala on multi-GPU-enabled Spark .

Flux : includes interfaces for RNNs, including GRUs and LSTMs, written in Julia .

Keras : High-level API, providing a wrapper to many other deep learning libraries.

Microsoft Cognitive Toolkit

MXNet : an open-source deep learning framework used to train and deploy deep neural networks.

PyTorch : Tensors and Dynamic neural networks in Python with GPU acceleration.

TensorFlow : Apache 2.0-licensed Theano-like library with support for CPU, GPU and Google's proprietary TPU , [116] mobile

Theano : A deep-learning library for Python with an API largely compatible with the NumPy library.

Torch : A scientific computing framework with support for machine learning algorithms, written in C and Lua .

Applications

Applications of recurrent neural networks include:

Machine translation [42]

Robot control [117]

Time series prediction [118] [119] [120]

Speech recognition [121] [39] [122]

Speech synthesis [123]

Brain–computer interfaces [124]

Time series anomaly detection [125]

Text-to-Video model [126]

Rhythm learning [127]

Music composition [128]

Grammar learning [129] [58] [130]

Handwriting recognition [131] [132]

Human action recognition [133]

Protein homology detection [134]

Predicting subcellular localization of proteins [135]

Several prediction tasks in the area of business process management [136]

Prediction in medical care pathways [137]

Predictions of fusion plasma disruptions in reactors (Fusion Recurrent Neural Network (FRNN) code) [138]

References

Further reading

Mandic, Danilo P.; Chambers, Jonathon A. (2001). Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability . Wiley. ISBN 978-0-471-49517-8 .

Grossberg, Stephen (2013-02-22). "Recurrent Neural Networks" . Scholarpedia . 8 (2): 1888. Bibcode : 2013SchpJ...8.1888G . doi : 10.4249/scholarpedia.1888 . ISSN 1941-6016 .

Recurrent Neural Networks . List of RNN papers by Jürgen Schmidhuber 's group at Dalle Molle Institute for Artificial Intelligence Research .

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch

Walter Pitts

John von Neumann

Claude Shannon

Shun'ichi Amari

Kunihiko Fukushima

Takeo Kanade

Marvin Minsky

John McCarthy
Nathaniel Rochester
Allen Newell
Cliff Shaw
Herbert A. Simon
Oliver Selfridge
Frank Rosenblatt
Bernard Widrow
Joseph Weizenbaum
Seymour Papert
Seppo Linnainmaa
Paul Werbos
Geoffrey Hinton
John Hopfield
Jürgen Schmidhuber
Yann LeCun
Yoshua Bengio
Lotfi A. Zadeh
Stephen Grossberg
Alex Graves
James Goodnight
Andrew Ng
Fei-Fei Li
Alex Krizhevsky
Ilya Sutskever
Oriol Vinyals
Quoc V. Le
Ian Goodfellow
Demis Hassabis
David Silver
Andrej Karpathy
Ashish Vaswani
Noam Shazeer
Aidan Gomez
John Schulman
Mustafa Suleyman
Jan Leike
Daniel Kokotajlo

François Chollet

Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category

GND