# Recurrent neural network

A **recurrent neural network** (**RNN**) is a class of [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) where connections between nodes form a [directed graph](https://en.wikipedia.org/wiki/Directed_graph) along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_networks), RNNs can use their internal state (memory) to process variable length sequences of inputs.[[1]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-1) This makes them applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition)[[2]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-2) or [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition).[[3]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-sak2014-3)[[4]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-liwu2015-4)

The term “recurrent neural network” is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is [finite impulse](https://en.wikipedia.org/wiki/Finite_impulse_response) and the other is [infinite impulse](https://en.wikipedia.org/wiki/Infinite_impulse_response). Both classes of networks exhibit temporal [dynamic behavior](https://en.wikipedia.org/wiki/Dynamic_system).[[5]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-5) A finite impulse recurrent network is a [directed acyclic graph](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Directed_cyclic_graph) that can not be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of [long short-term memory](https://en.wikipedia.org/wiki/Long_short-term_memory) networks (LSTMs) and [gated recurrent units](https://en.wikipedia.org/wiki/Gated_recurrent_unit). This is also called Feedback Neural Network.



**Architectures**

RNNs come in many variants.

**Fully recurrent**

Unfolded basic recurrent neural network

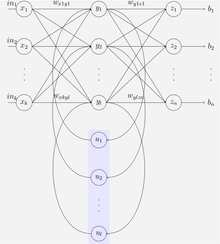
Basic RNNs are a network of [neuron-like](https://en.wikipedia.org/wiki/Artificial_neuron) nodes organized into successive layers. Each node in a given layer is connected with a [directed (one-way) connection](https://en.wikipedia.org/wiki/Directed_graph) to every other node in the next successive layer.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued [weight](https://en.wikipedia.org/wiki/Weighting). Nodes are either input nodes (receiving data from outside the network), output nodes (yielding results), or hidden nodes (that modify the data *en route* from input to output).

For [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) in discrete time settings, sequences of real-valued input vectors arrive at the input nodes, one vector at a time. At any given time step, each non-input unit computes its current activation (result) as a nonlinear function of the weighted sum of the activations of all units that connect to it. Supervisor-given target activations can be supplied for some output units at certain time steps. For example, if the input sequence is a speech signal corresponding to a spoken digit, the final target output at the end of the sequence may be a label classifying the digit.

In [reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning) settings, no teacher provides target signals. Instead a [fitness function](https://en.wikipedia.org/wiki/Fitness_function) or [reward function](https://en.wikipedia.org/wiki/Reward_function) is occasionally used to evaluate the RNN's performance, which influences its input stream through output units connected to actuators that affect the environment. This might be used to play a game in which progress is measured with the number of points won.

Each sequence produces an error as the sum of the deviations of all target signals from the corresponding activations computed by the network. For a training set of numerous sequences, the total error is the sum of the errors of all individual sequences.

**Elman networks and Jordan networks**

[](https://en.wikipedia.org/wiki/File:Elman_srnn.png)

The Elman network

An [Elman](https://en.wikipedia.org/wiki/Jeff_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.[[20]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-bmm615-20) At each time step, the input is fed forward and a [learning rule](https://en.wikipedia.org/wiki/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 such tasks as sequence-prediction that are beyond the power of a standard [multilayer perceptron](https://en.wikipedia.org/wiki/Multilayer_perceptron).

[Jordan](https://en.wikipedia.org/wiki/Michael_I._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 referred to as the state layer. They have a recurrent connection to themselves.[[20]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-bmm615-20)

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

**Independently RNN (IndRNN)**

The Independently recurrent neural network (IndRNN)[[28]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-auto-28) 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 the non-saturated nonlinear functions such as ReLU. Using skip connections, deep networks can be trained.

**Recursive**

Main article: [Recursive neural network](https://en.wikipedia.org/wiki/Recursive_neural_network)

A [recursive neural network](https://en.wikipedia.org/wiki/Recursive_neural_network)[[29]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-29) is created by applying the same set of weights [recursively](https://en.wikipedia.org/wiki/Recursion) over a differentiable graph-like structure by traversing the structure in [topological order](https://en.wikipedia.org/wiki/Topological_sort). Such networks are typically also trained by the reverse mode of [automatic differentiation](https://en.wikipedia.org/wiki/Automatic_differentiation).[[30]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-lin1970-30)[[31]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-grie2008-31) They can process [distributed representations](https://en.wikipedia.org/wiki/Distributed_representation) of structure, such as [logical terms](https://en.wikipedia.org/wiki/Mathematical_logic). 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](https://en.wikipedia.org/wiki/Natural_language_processing).[[32]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-32) The Recursive Neural Tensor Network uses a [tensor](https://en.wikipedia.org/wiki/Tensor)-based composition function for all nodes in the tree.[[33]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-33)

### Hierarchical

Hierarchical RNNs connect their neurons in various ways to decompose hierarchical behavior into useful subprograms.[[34]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-schmidhuber1992-34)[[54]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-54)

### Recurrent multilayer perceptron network

Generally, a Recurrent Multi-Layer Perceptron (RMLP) network consists of cascaded subnetworks, each of which contains multiple layers of nodes. Each of these subnetworks is feed-forward except for the last layer, which can have feedback connections. Each of these subnets is connected only by feed forward connections.[[55]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-55)

### 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 that depends on spatial connection between neurons and on distinct types of neuron activities, each with distinct time properties.[[56]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-56)[[57]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-57) 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](https://en.wikipedia.org/wiki/Memory-prediction_framework) theory of brain function by [Hawkins](https://en.wikipedia.org/wiki/Jeff_Hawkins) in his book [*On Intelligence*](https://en.wikipedia.org/wiki/On_Intelligence).[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

### Neural Turing machines

Main article: [Neural Turing machine](https://en.wikipedia.org/wiki/Neural_Turing_machine)

Neural Turing machines (NTMs) are a method of extending recurrent neural networks by coupling them to external [memory](https://en.wikipedia.org/wiki/Memory) resources which they can interact with by [attentional processes](https://en.wikipedia.org/w/index.php?title=Attentional_process&action=edit&redlink=1). The combined system is analogous to a [Turing machine](https://en.wikipedia.org/wiki/Turing_machine) or [Von Neumann architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture) but is [differentiable](https://en.wikipedia.org/wiki/Differentiable_neural_computer) end-to-end, allowing it to be efficiently trained with [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent).[[58]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-58)

### Differentiable neural computer

Main article: [Differentiable neural computer](https://en.wikipedia.org/wiki/Differentiable_neural_computer)

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

### Neural network pushdown automata

Neural network pushdown automata (NNPDA) are similar to NTMs, but tapes are replaced by analogue stacks that are differentiable and that are trained. In this way, they are similar in complexity to recognizers of [context free grammars](https://en.wikipedia.org/wiki/Context_free_grammar) (CFGs).[[59]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-59)

### Memristive Networks

Greg Snider of [HP Labs](https://en.wikipedia.org/wiki/HP_Labs) describes a system of cortical computing with memristive nanodevices.[[60]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-60) The [memristors](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/DARPA)'s [SyNAPSE project](https://en.wikipedia.org/wiki/SyNAPSE) has funded IBM Research and HP Labs, in collaboration with the Boston University Department of Cognitive and Neural Systems (CNS), to develop neuromorphic architectures which may be based on memristive systems. Memristive networks are a particular type of [physical neural network](https://en.wikipedia.org/wiki/Physical_neural_network) that have very similar properties to (Little-)Hopfield networks, as they have a continuous dynamics, have a limited memory capacity and they natural relax via the minimization of a function which is asymptotic to the Ising model. In this sense, the dynamics of a memristive circuit has the advantage compared to a Resistor-Capacitor network to have a more interesting non-linear behavior. From this point of view, engineering an analog memristive networks accounts to a peculiar type of [neuromorphic engineering](https://en.wikipedia.org/wiki/Neuromorphic_engineering) in which the device behavior depends on the circuit wiring, or topology. [[61]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-61)[[62]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-62)

**Global optimization methods**

Training the weights in a neural network can be modeled as a non-linear [global optimization](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Genetic_algorithm), especially in unstructured networks.[[77]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-77)[[78]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-78)[[79]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-79)

Initially, the genetic algorithm is encoded with the neural network weights in a predefined manner where one gene in the [chromosome](https://en.wikipedia.org/wiki/Chromosome_(genetic_algorithm)) 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 is:

* When the neural network has learnt a certain percentage of the training data or
* When the minimum value of the mean-squared-error is satisfied or
* When the maximum number of training generations has been reached.

The stopping criterion is evaluated by the fitness function as it gets the reciprocal of the mean-squared-error 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](https://en.wikipedia.org/wiki/Simulated_annealing) or [particle swarm optimization](https://en.wikipedia.org/wiki/Particle_swarm_optimization).