1. What is the function of a summation junction of a neuron? What is threshold activation

Function?

Ans; A **summation junction** for the input signals is weighted by the respective synaptic weight. Because it is a linear combiner or adder of the weighted input signals, the output of the summation junction can be expressed as follows:

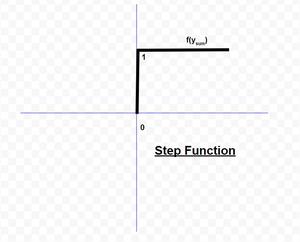
y sum = ∑=ni=1  = wi \*xi

A threshold **activation function** (or simply **the activation function,** also known as **squashing function**) results in an output signal only when an input signal exceeding a specific threshold value comes as an input.

2. What is a step function? What is the difference of step function with threshold function?

Answer :

It is a commonly used activation function. As depicted in the diagram, it gives **1 as output** of the input is either 0 or positive. If the input is negative, it gives **0 as output**. Expressing it mathematically,



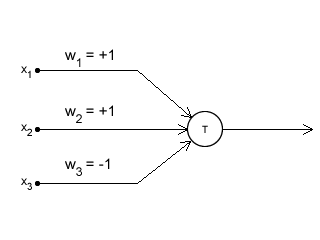
3. Explain the McCulloch–Pitts model of neuron.

Answer: The McCulloch-Pitts model was an extremely simple artificial neuron. The inputs could be either a zero or a one. And the output was a zero or a one. And each input could be either excitatory or inhibitory.

Now the whole point was to sum the inputs. If an input is one, and is excitatory in nature, it added one. If it was one, and was inhibitory, it subtracted one from the sum. This is done for all inputs, and a final sum is calculated.

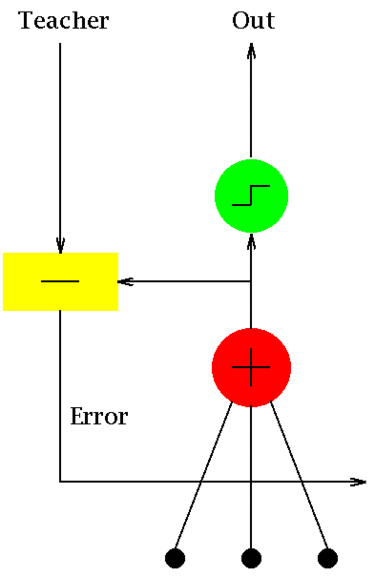
Now, if this final sum is less than some value (which you decide, say T), then the output is zero. Otherwise, the output is a one.

Here is a graphical representation of the McCulloch-Pitts model



4. Explain the ADALINE network model.

Ans: MADALINE (Many ADALINE) is **a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecture for classification that uses ADALINE units in its hidden and output layers**, i.e. its activation function is the sign function. The three-layer network uses memistors.It consists of a weight, a bias and a summation function.



5. What is the constraint of a simple perceptron? Why it may fail with a real-world data set?

Ans: Perceptron networks have several limitations.

First, the output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function.

Second, perceptrons can only classify linearly separable sets of vectors. If a straight line or a plane can be drawn to separate the input vectors into their correct categories, the input vectors are linearly separable. If the vectors are not linearly separable, learning will never reach a point where all vectors are classified properly

6. What is linearly inseparable problem? What is the role of the hidden layer?

Ans: This name is given to them, because if we were to represent them in the input space, we could classifythem using a straight line. The simplest examples are the logical AND or OR.

If you can draw a line or hyper plane that can separate those points into two classes, then

the data is separable.

In neural networks, a hidden layer is located between the input and output of the algorithm, in which the function applies weights to the inputs and directs them through an activation function as the output. In short, the **hidden layers perform nonlinear transformations of the inputs entered into the network**.

7. Explain XOR problem in case of a simple perceptron.

Ans : The XOr problem is that we need to build a Neural Network (a perceptron in our case) to produce the truth table related to the XOr logical operator. This is a binary classification problem

8. Design a multi-layer perceptron to implement A XOR B.

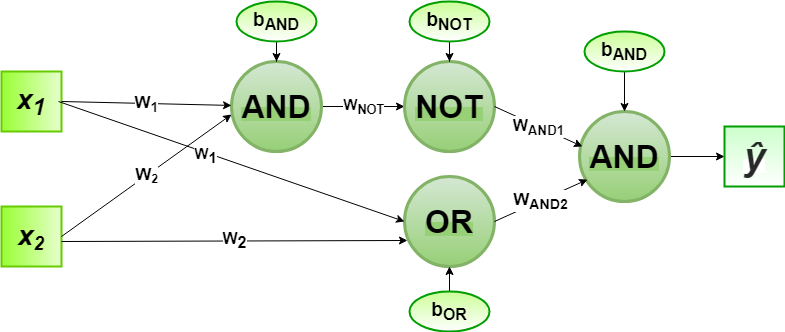
Ans:XOR logical function truth table for 2-bit binary variables. I.e. the input vector x: (x1,x2) ans corresponding output y

| x1 | x2 | y |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

XOR(x1,x2) = AND(NOT(AND(x1,x2)), OR(x1,x2))

Designing the Perceptron Network:

1. **Step1:** Now for the corresponding weight vector w:(w1,w2) of the input vector x:(x1,x2)to the AND and OR node, the associated Perceptron Function can be defined as:  
    Ŷ1 = (w1x1+w2x2+bAND)  
    Ŷ2 = (w1x1+w2x2+bOR)
2. **Step2:** The output Ŷ from the AND node will be input to the NOT node with weight wNOT and the associated Perceptron Function can be defined as:  
    Ŷ3 = (wNOT Ŷ1 +bNOT)
3. **Step3:** The output Ŷ2 from the OR node and the output Ŷ3 from NOT node as mentioned in Step2 will be input to the AND node with weight (wAND1, wAND2). Then the corresponding output Ŷis the final output of the XOR logic function. The associated
4. Perceptron Function can be defined as:  
    Ŷ = (wAND1Ŷ + wAND2Ŷ + bAND)



For the implementation, the weight parameters are considered to be

W1 =1, W2 =1, wNOT = -1, wAND1=1, wAND2 =1

and the bias parameters are

bAND= -1.5, bOR = -0.5, bNOT=0.5

9. Explain the single-layer feed forward architecture of ANN.

Ans: An [Artificial Neural Network (ANN)](https://www.geeksforgeeks.org/implementing-ann-training-process-in-python/) is an information processing paradigm that is inspired by the brain. ANNs, like people, learn by examples. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning largely involves adjustments to the synaptic connections that exist between the neurons.

The model of an artificial neural network can be specified by three entities:

* **Interconnections**
* [**Activation functions**](https://www.geeksforgeeks.org/activation-functions-neural-networks/)
* **Learning rules**

Interconnection can be defined as the way processing elements (Neuron) in ANN are connected to each other. Hence, the arrangements of these processing elements and geometry of interconnections are very essential in ANN. These arrangements always have two layers that are common to all network architectures, the Input layer and output layer where the input layer buffers the input signal, and the output layer generates the output of the network. The third layer is the Hidden layer, in which neurons are neither kept in the input layer nor in the output layer. These neurons are hidden from the people who are interfacing with the system and act as a black box to them. By increasing the hidden layers with neurons, the system’s computational and processing power can be increased but the training phenomena of the system get more complex at the same time.

10. Explain the competitive network architecture of ANN.

Answer:A neural network consists of three layers. The first layer is the input layer. It contains the input neurons that send information to the hidden layer. The hidden layer performs the computations on input data and transfers the output to the output layer. It includes weight, activation function, cost function.

The connection between neurons is known as weight, which is the numerical values. The weight between neurons determines the learning ability of the neural network. During the learning of artificial neural networks, weight between the neuron changes.

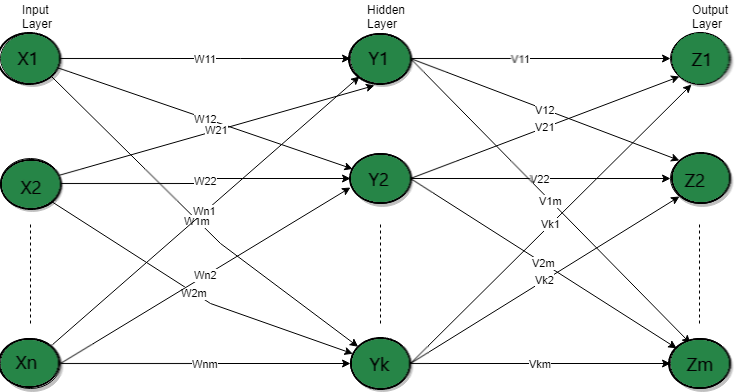
## Working of ANN

Firstly, the information is fed into the input layer. Which then transfers it to the hidden layers, and interconnection between these two layers assign weights to each input randomly at the initial point. Then bias is added to each input neuron and after this, the weight sum which is a combination of weights and bias is passed through the activation function. Activation Function has the responsibility of which node to fire for feature extraction and finally output is calculated. Therefore this whole process is known as Forward Propagation. After getting the output model to compare it with the original output the error is known and finally, weights are updated in backward propagation to reduce the error and this process continues for a certain number of epochs (iteration). Finally, model weights get updates and prediction is done.

11. Consider a multi-layer feed forward neural network. Enumerate and explain steps in the

backpropagation algorithm used to train the network.

Answer:This layer also has a hidden layer that is internal to the network and has no direct contact with the external layer. The existence of one or more hidden layers enables the network to be computationally stronger, a feed-forward network because of information flow through the input function, and the intermediate computations used to determine the output Z. There are no feedback connections in which outputs of the model are fed back into itself.



12. What are the advantages and disadvantages of neural networks?

Ans:

| Advantages | DisAvantages |
| --- | --- |
| A neural network can implement tasks that a linear program cannot. | The neural network required training to operate. |
| When an item of the neural network declines, it can continue without some issues by its parallel features. | The structure of a neural network is disparate from the structure of microprocessors therefore required to be emulated. |
| A neural network determines and does not require to be reprogrammed. | It needed high processing time for big neural networks. Heavy Hardware Requirement |
| It can be executed in any application. | Incomplete Results,Since ANNs are trained to adapt to the changing applications of neural networks, they are often left untrained for the whole process. |
| Effective Visual Analysis | Data Suitability |
| Processing of Unorganized Data | Minimal Control |
| Adaptive Structure |  |

13. Write short notes on any two of the following:

1. Biological neuron

**Biological neuron models**, also known as a **spiking neuron models**,[[1]](https://en.wikipedia.org/wiki/Biological_neuron_model#cite_note-Gerstner_2002-1) are mathematical descriptions of the properties of certain cells in the nervous system that generate sharp electrical potentials [across their cell membrane](https://en.wikipedia.org/wiki/Membrane_potential), roughly one millisecond in duration, called action potentials or spikes (Fig. 2). Since spikes are transmitted along the [axon](https://en.wikipedia.org/wiki/Axon) and [synapses](https://en.wikipedia.org/wiki/Synapse) from the sending neuron to many other neurons, spiking [neurons](https://en.wikipedia.org/wiki/Neuron) are considered to be a major information processing unit of the [nervous system](https://en.wikipedia.org/wiki/Nervous_system).Spiking neuron models can be divided into different categories: the most detailed mathematical models are biophysical neuron models (also called Hodgkin-Huxley models) that describe the membrane voltage as a function of the input current and the activation of ion channels. Mathematically simpler are integrate-and-fire models that describe the membrane voltage as a function of the input current and predict the spike times without a description of the biophysical processes that shape the time course of an action potential. Even more abstract models only predict output spikes (but not membrane voltage) as a function of the stimulation where the stimulation can occur through sensory input or pharmacologically.

2. ReLU function

The **rectified linear activation function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

* The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.
* The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks.

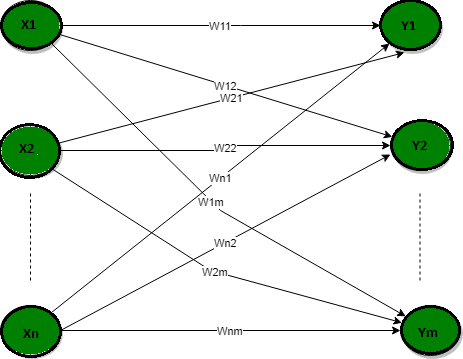
In order to use stochastic gradient descent with [backpropagation of errors](https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/) to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned.

The function must also provide more sensitivity to the activation sum input and avoid easy saturation.A node or unit that implements this activation function is referred to as a **rectified linear activation unit**, or ReLU for short. Often, networks that use the rectifier function for the hidden layers are referred to as rectified networks.The rectified linear activation function is a simple calculation that returns the value provided as input directly, or the value 0.0 if the input is 0.0 or less.

The function is linear for values greater than zero, meaning it has a lot of the desirable properties of a linear activation function when training a neural network using backpropagation. Yet, it is a nonlinear function as negative values are always output as zero.Because the rectified function is linear for half of the input domain and nonlinear for the other half, it is referred to as a [piecewise linear function](https://en.wikipedia.org/wiki/Piecewise_linear_function) or a hinge function

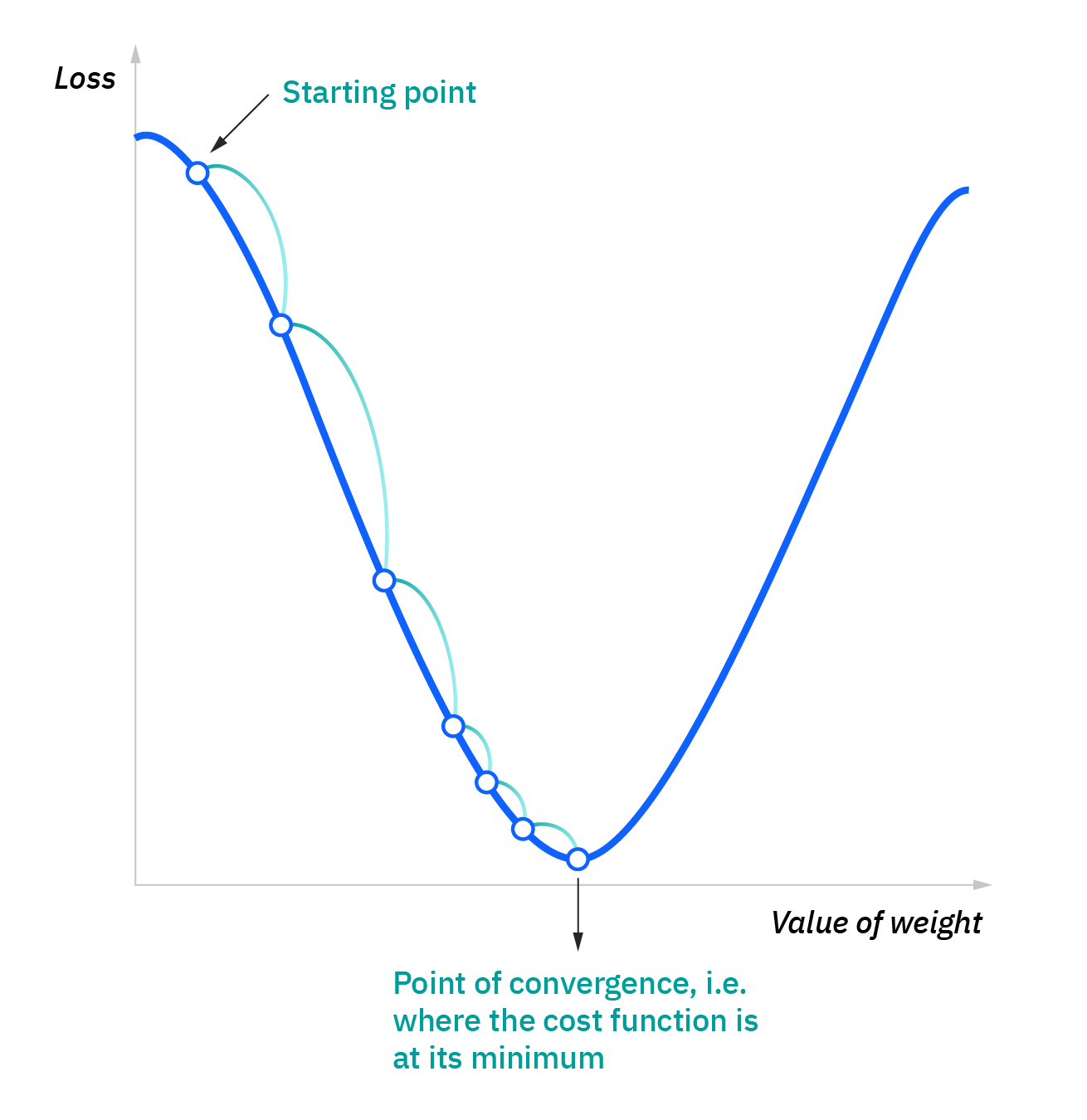
3. Single-layer feed forward ANN

In this type of network, we have only two layers input layer and the output layer but the input layer does not count because no computation is performed in this layer. The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken. After this, the neurons collectively give the output layer to compute the output signals.



4. Gradient descent

Gradient descent is an optimization algorithm which is commonly-used to train [machine learning](https://www.ibm.com/cloud/learn/machine-learning) models and [neural networks](https://www.ibm.com/cloud/learn/neural-networks). Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates. Until the function is close to or equal to zero, the model will continue to adjust its parameters to yield the smallest possible error. Once machine learning models are optimized for accuracy, they can be powerful tools for artificial intelligence (AI) and computer science applications.

plotting a scatterplot in statistics and finding the line of best fit, which required calculating the error between the actual output and the predicted output (y-hat) using the mean squared error formula. The gradient descent algorithm behaves similarly, but it is based on a convex function, such as the one below:

The starting point is just an arbitrary point for us to evaluate the performance. From that starting point, we will find the derivative (or slope), and from there, we can use a tangent line to observe the steepness of the slope. The slope will inform the updates to the parameters—i.e. the weights and bias. The slope at the starting point will be steeper, but as new parameters are generated, the steepness should gradually reduce until it reaches the lowest point on the curve, known as the point of convergence.

Similar to finding the line of best fit in linear regression, the goal of gradient descent is to minimize the cost function, or the error between predicted and actual y. In order to do this, it requires two data points—a direction and a learning rate. These factors determine the partial derivative calculations of future iterations, allowing it to gradually arrive at the local or global minimum (i.e. point of convergence). More detail on these components can be found below:

* Learning rate (also referred to as step size or the alpha) is the size of the steps that are taken to reach the minimum. This is typically a small value, and it is evaluated and updated based on the behavior of the cost function. High learning rates result in larger steps but risks overshooting the minimum. Conversely, a low learning rate has small step sizes. While it has the advantage of more precision, the number of iterations compromises overall efficiency as this takes more time and computations to reach the minimum
* The cost (or loss) function measures the difference, or error, between actual y and predicted y at its current position. This improves the machine learning model's efficacy by providing feedback to the model so that it can adjust the parameters to minimize the error and find the local or global minimum. It continuously iterates, moving along the direction of steepest descent (or the negative gradient) until the cost function is close to or at zero. At this point, the model will stop learning. Additionally, while the terms, cost function and loss function, are considered synonymous, there is a slight difference between them. It’s worth noting that a loss function refers to the error of one training example, while a cost function calculates the average error across an entire training set.

5. Recurrent networks

A **recurrent neural network** (**RNN**) is a class of [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. 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)[[2]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-3) This makes them applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition)[[4]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-4) or [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition).[[5]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-sak2014-5)[[6]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-liwu2015-6) Recurrent neural networks are theoretically [Turing complete](https://en.wikipedia.org/wiki/Turing_complete) and can run arbitrary programs to process arbitrary sequences of inputs.[[7]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-7)

The term "recurrent neural network" is used to refer to the class of networks with an [infinite impulse response](https://en.wikipedia.org/wiki/Infinite_impulse_response), whereas "[convolutional neural network](https://en.wikipedia.org/wiki/Convolutional_neural_network)" refers to the class of [finite impulse](https://en.wikipedia.org/wiki/Finite_impulse_response) response. Both classes of networks exhibit temporal [dynamic behavior](https://en.wikipedia.org/wiki/Dynamic_system).[[8]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-8) 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 (FNN).