# Neural Network Toolbox

## Create, train, and simulate neural networks

Neural Network Toolbox™ provides functions and apps for modeling complex nonlinear systems that are not easily modeled with a closed-form equation. Neural Network Toolbox supports supervised learning with feedforward, radial basis, and dynamic networks. It also supports unsupervised learning with self-organizing maps and competitive layers. With the toolbox you can design, train, visualize, and simulate neural networks. You can use Neural Network Toolbox for applications such as data fitting, pattern recognition, clustering, time-series prediction, and dynamic system modeling and control.

To speed up training and handle large data sets, you can distribute computations and data across multicore processors, GPUs, and computer clusters using Parallel Computing Toolbox™.

**Key Features**

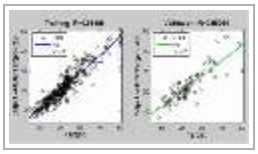
- Supervised networks, including multilayer, radial basis, learning vector quantization (LVQ), time-delay, nonlinear autoregressive (NARX), and layer-recurrent
- Unsupervised networks, including self-organizing maps and competitive layers
- Apps for data-fitting, pattern recognition, and clustering
- Parallel computing and GPU support for accelerating training (using Parallel Computing Toolbox)
- Preprocessing and postprocessing for improving the efficiency of network training and assessing network performance
- Modular network representation for managing and visualizing networks of arbitrary size
- Simulink® blocks for building and evaluating neural networks and for control systems applications


Getting Started with Neural Network Toolbox 4:20
Use graphical tools to apply neural networks to data fitting, pattern recognition, clustering, and time series problems.

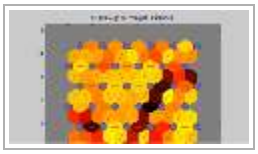
**Data Fitting, Clustering, and Pattern Recognition**

Like its counterpart in the biological nervous system, a neural network can learn and therefore can be trained to find solutions, recognize patterns, classify data, and forecast future events. The behavior of a neural network is defined by the way its individual computing elements are connected and by the strengths of those connections, or weights. The weights are automatically adjusted by training the network according to a specified learning rule until it performs the desired task correctly.

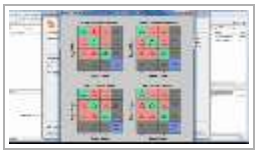
Neural Network Toolbox includes command-line functions and apps for creating, training, and simulating neural networks. The apps make it easy to develop neural networks for tasks such as data-fitting (including time-series data), pattern recognition, and clustering. After creating your networks in these tools, you can automatically generate MATLAB® code to capture your work and automate tasks.



House Pricing Estimation with Neural Net Fitting App 4:21
Estimate median house prices for neighborhoods based on various neighborhood attributes.



Iris Flower Clustering with Neural Net Clustering App 3:48
Cluster iris flowers based on petal and sepal size.



Wine Classification with Neural Net Pattern Recognition App 3:36
Identify the winery that particular wines came from based on chemical attributes of the wine.

**Network Architectures**

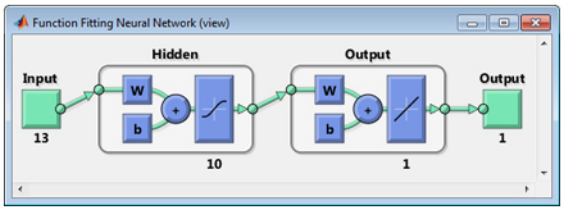
Neural Network Toolbox supports a variety of supervised and unsupervised network architectures. With the toolbox's modular approach to building networks, you can develop custom network architectures for your specific problem. You can view the network architecture including all inputs, layers, outputs, and interconnections.

Supervised Networks

Supervised neural networks are trained to produce desired outputs in response to sample inputs, making them particularly well-suited to modeling and controlling dynamic systems, classifying noisy data, and predicting future events.

Neural Network Toolbox includes four types of supervised networks: feedforward, radial basis, dynamic, and learning vector quantization.

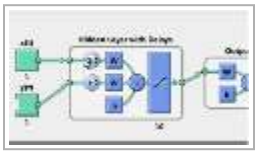
**Feedforward networks** have one-way connections from input to output layers. They are most commonly used for prediction, pattern recognition, and nonlinear function fitting. Supported feedforward networks include feedforward backpropagation, cascade-forward backpropagation, feedforward input-delay backpropagation, linear, and perceptron networks.




*A two-layer feedforward network with sigmoid hidden neurons and linear output neurons. This type of network can fit multidimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.*

**Radial basis networks** provide an alternative, fast method for designing nonlinear feedforward networks. Supported variations include generalized regression and probabilistic neural networks.

**Dynamic networks** use memory and recurrent feedback connections to recognize spatial and temporal patterns in data. They are commonly used for time-series prediction, nonlinear dynamic system modeling, and control systems applications. Prebuilt dynamic networks in the toolbox include focused and distributed time-delay, nonlinear autoregressive (NARX), layer-recurrent, Elman, and Hopfield networks. The toolbox also supports dynamic training of custom networks with arbitrary connections.



Maglev Modeling with Neural Time Series App 5:28
Model the position of a levitated magnet as current passes through an electromagnet beneath it.

**Learning vector quantization (LVQ) networks** use a method for classifying patterns that are not linearly separable. LVQ lets you specify class boundaries and the granularity of classification.

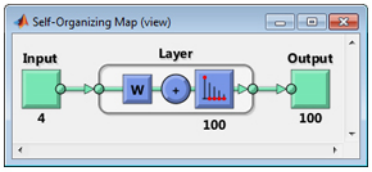
Unsupervised Networks

Unsupervised neural networks are trained by letting the network continually adjust itself to new inputs. They find relationships within data and can automatically define classification schemes.

Neural Network Toolbox includes two types of self-organizing, unsupervised networks: competitive layers and self-organizing maps.

**Competitive layers** recognize and group similar input vectors, enabling them to automatically sort inputs into categories. Competitive layers are commonly used for classification and pattern recognition.

**Self-organizing maps** learn to classify input vectors according to similarity. Like competitive layers, they are used for classification and pattern recognition tasks; however, they differ from competitive layers because they are able to preserve the topology of the input vectors, assigning nearby inputs to nearby categories.




*A self-organizing map consisting of a competitive layer that can classify a data set of vectors with any number of dimensions into as many classes as the layer has neurons.*

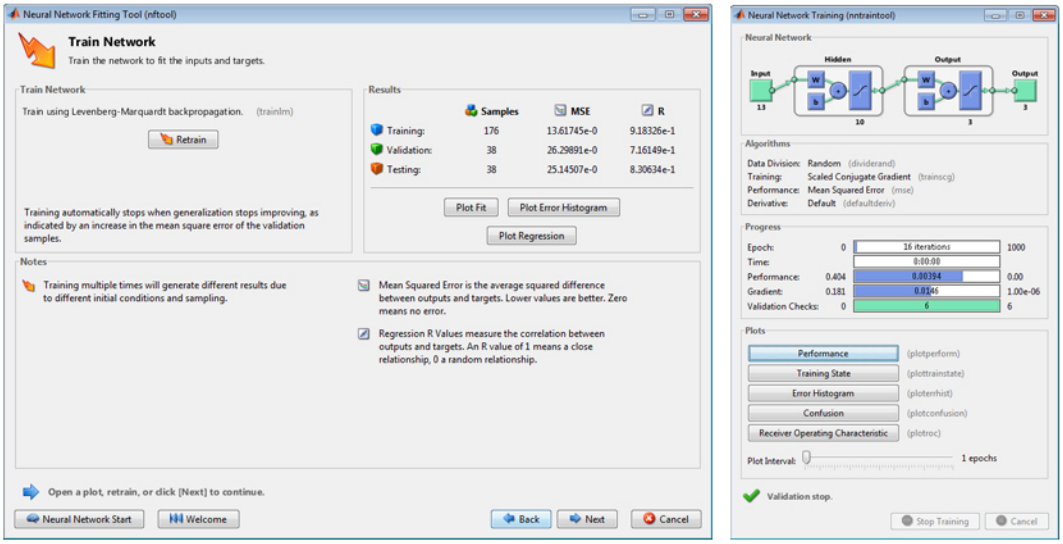
### Training Algorithms

Training and learning functions are mathematical procedures used to automatically adjust the network's weights and biases. The training function dictates a global algorithm that affects all the weights and biases of a given network. The learning function can be applied to individual weights and biases within a network.

Neural Network Toolbox supports a variety of training algorithms, including several gradient descent methods, conjugate gradient methods, the Levenberg-Marquardt algorithm (LM), and the resilient backpropagation algorithm (Rprop). The toolbox's modular framework lets you quickly develop custom training algorithms that can be integrated with built-in algorithms. While training your neural network, you can use error weights to define the relative importance of desired outputs, which can be prioritized in terms of sample, time step (for time-series problems), output element, or any combination of these. You can access training algorithms from the command

line or via apps that show diagrams of the network being trained and provide network performance plots and status information to help you monitor the training process.

A suite of learning functions, including gradient descent, Hebbian learning, LVQ, Widrow-Hoff, and Kohonen, is also provided.

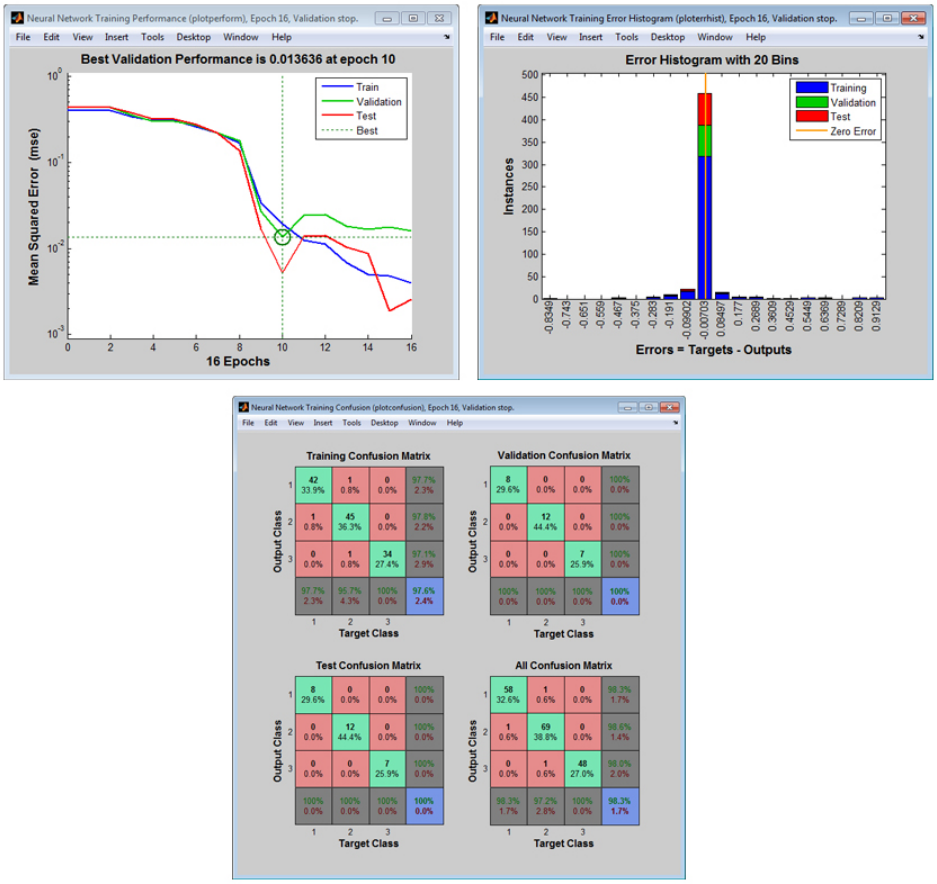



*Neural network apps that automate training your neural network to fit input and target data (left), monitor training progress (right), and calculate statistical results and plots to assess training quality.*

### Preprocessing and Postprocessing

Preprocessing the network inputs and targets improves the efficiency of neural network training. Postprocessing enables detailed analysis of network performance. Neural Network Toolbox provides preprocessing and postprocessing functions and Simulink blocks that enable you to:

- Reduce the dimensions of the input vectors using principal component analysis
- Perform regression analysis between the network response and the corresponding targets
- Scale inputs and targets so they fall in the range [-1,1]
- Normalize the mean and standard deviation of the training set
- Use automated data preprocessing and data division when creating your networks




*Postprocessing plots to analyze network performance, including mean squared error validation performance for successive training epochs (top left), error histogram (top right), and confusion matrices (bottom) for training, validation, and test phases.*

### Improved Generalization

Improving the network's ability to generalize helps prevent overfitting, a common problem in neural network design. Overfitting occurs when a network has memorized the training set but has not learned to generalize to new inputs. Overfitting produces a relatively small error on the training set but a much larger error when new data is presented to the network.

Neural Network Toolbox provides two solutions to improve generalization:

- **Regularization** modifies the network's performance function (the measure of error that the training process minimizes). By including the sizes of the weights and biases, regularization produces a network that performs well with the training data and exhibits smoother behavior when presented with new data.
- **Early stopping** uses two different data sets: the training set, to update the weights and biases, and the validation set, to stop training when the network begins to overfit the data.

### Simulink Blocks and Control Systems Applications

Simulink Support

Neural Network Toolbox provides a set of blocks for building neural networks in Simulink. All blocks are compatible with Simulink Coder™. These blocks are divided into four libraries:

- **Transfer function blocks**, which take a net input vector and generate a corresponding output vector

- **Net input function blocks**, which take any number of weighted input vectors, weight-layer output vectors, and bias vectors, and return a net input vector
- **Weight function blocks**, which apply a neuron's weight vector to an input vector (or a layer output vector) to get a weighted input value for a neuron
- **Data preprocessing blocks**, which map input and output data into the ranges best suited for the neural network to handle directly

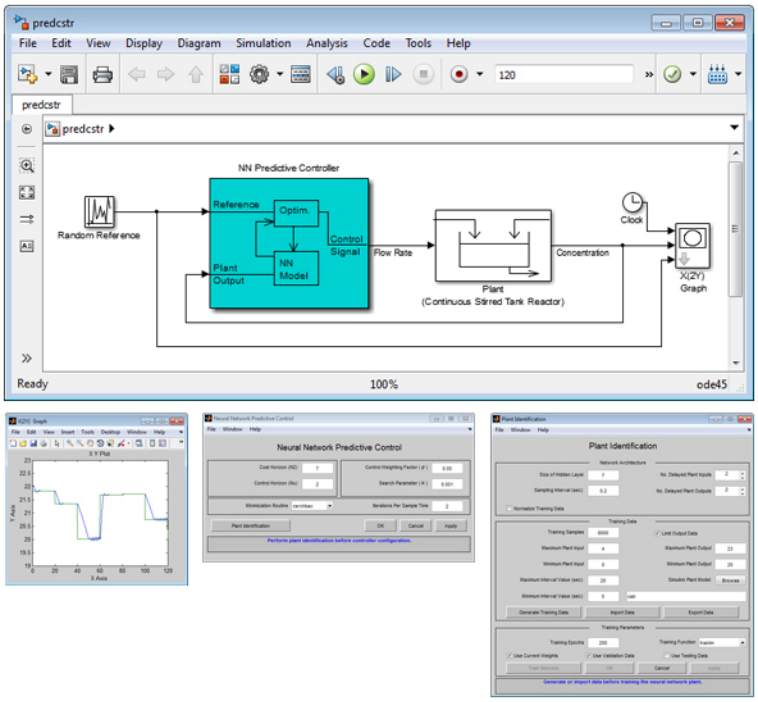
Alternatively, you can create and train your networks in the MATLAB environment and automatically generate network simulation blocks for use with Simulink. This approach also enables you to view your networks graphically.

Control Systems Applications

You can apply neural networks to the identification and control of nonlinear systems. The toolbox includes descriptions, examples, and Simulink blocks for three popular control applications:

- **Model predictive control**, which uses a neural network model to predict future plant responses to potential control signals. An optimization algorithm then computes the control signals that optimize future plant performance. The neural network plant model is trained offline and in batch form.
- **Feedback linearization**, which uses a rearrangement of the neural network plant model and is trained offline. This controller requires the least computation of these three architectures; however, the plant must either be in companion form or be capable of approximation by a companion form model.
- **Model reference adaptive control**, which requires that a separate neural network controller be trained offline, in addition to the neural network plant model. While the controller training is computationally expensive, the model reference control applies to a larger class of plant than feedback linearization.

You can incorporate neural network predictive control blocks included in the toolbox into your Simulink models. By changing the parameters of these blocks, you can tailor the network's performance to your application.




*A Simulink model that uses the Neural Network Predictive Controller block with a tank reactor plant model (top). You can visualize the dynamic response (bottom left) and manage the controller block (bottom center) and your plant identification (bottom right).*

**Accelerated Training and Large Data Sets**

You can speed up neural network training and simulation of large data sets by using Neural Network Toolbox with Parallel Computing Toolbox. Training and simulation involve many parallel computations, which can be accelerated with multicore processors, CUDA-enabled NVIDIA GPUs, and computer clusters with multiple processors and GPUs.

Distributed Computing

Parallel Computing Toolbox lets neural network training and simulation run across multiple processor cores on a single PC, or across multiple processors on multiple computers on a network using MATLAB Distributed Computing Server™. Using multiple cores can speed up calculations. Using multiple computers lets you solve problems using data sets too big to fit within the system memory of any single computer. The only limit to problem size is the total system memory available across all computers.

GPU Computing

Parallel Computing Toolbox enables Neural Network Toolbox simulation and training to be parallelized across the multiprocessors and cores of a general-purpose graphics processing unit (GPU). GPUs are highly efficient on parallel algorithms such as neural networks. You can achieve higher levels of parallelism by using multiple GPUs or by using GPUs and processors together. With MATLAB Distributed Computing Server, you can harness all the processors and GPUs on a network cluster of computers for neural network training and simulation.

Learn more about GPU computing with MATLAB.

## Resources

**Product Details, Examples, and System Requirements**
www.mathworks.com/products/neural-network

**Trial Software**
www.mathworks.com/trialrequest

**Sales**
www.mathworks.com/contactsales

**Technical Support**
www.mathworks.com/support

**Online User Community**
www.mathworks.com/matlabcentral

**Training Services**
www.mathworks.com/training

**Third-Party Products and Services**
www.mathworks.com/connections

**Worldwide Contacts**
www.mathworks.com/contact