Name	**layrecnet()** Layer recurrent neural network
Description	each layer has a recurrent connection with a tap delay associated with it. This allows the network to have an infinite dynamic response to time series input data This network is similar to the time delay (`timedelaynet`) and distributed delay (`distdelaynet`) neural networks, which have finite input responses.
Syntax	layrecnet(layerDelays,hiddenSizes,trainFcn) `layerDelays` Row vector of increasing 0 or positive delays (default = 1:2) `hiddenSizes` Row vector of one or more hidden layer sizes (default = 10) `trainFcn` Training function (default = `'trainlm'`)
Suggested	Distdelaynet | narnet | narxnet | preparets | removedelay | timedelaynet

narnet: NAR (nonlinear autoregressive) neural networks can be trained to predict a time series from that series past values.
narxnet: NARX (Nonlinear autoregressive with external input) networks can learn to predict one time series given past values of the same time series, the feedback input, and another time series, called the external or exogenous time series.
elmannet: Elman networks are feedforward networks (*`feedforwardnet`*) with the addition of layer recurrent connections with tap delays. no longer recommended except for historical and research purposes
For more accurate learning try time delay (*`timedelaynet`*), layer recurrent (*`layrecnet`*), NARX (*`narxnet`*), and NAR (*`narnet`*) neural networks

Name	**con2seq()** Convert cell array to ordinary array of the underlying data type
Description	Neural Network Toolbox™ software arranges concurrent vectors with a matrix, and sequential vectors with a cell array (where the second index is the time step). Con2seq and seq2con allow concurrent vectors to be converted to sequential vectors, and back again.
Syntax	`S = con2seq(b)` `S = con2seq(b,TS)`

Name	**preparets()** Prepare input and target time series data for network simulation or training
Description	This function simplifies the normally complex and error prone task of reformatting input and target time series. It automatically shifts input and target time series as many steps as are needed to fill the initial input and layer delay states
Syntax	`[Xs,Xi,Ai,Ts,EWs,shift] = preparets(net,Xnf,Tnf,Tf,EW)` input: `net` Neural network `Xnf` Non-feedback inputs `Tnf` Non-feedback targets `Tf` Feedback targets `EW` Error weights (default = `{1}`) return: `Xs` Shifted inputs `Xi` Initial input delay states `Ai` Initial layer delay states `Ts` Shifted targets `EWs` Shifted error weights `shift` The number of timesteps truncated from the front of `X` and `T` in order to properly fill `Xi` and `Ai`.
Suggested	`Adddelay` | `closeloop` | `narnet` | `narxnet` | `openloop` | `removedelay` | `timedelaynet`

Name	**trainml** Levenberg-Marquardt backpropagation
Description	`trainlm` is often the fastest backpropagation algorithm in the toolbox, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms. Validation vectors are used to stop training early if the network performance on the validation vectors fails to improve or remains the same for `max_fail` epochs in a row. Test vectors are used as a further check that the network is generalizing well, but do not have any effect on training

Syntax	`net.trainFcn = 'trainlm'`
	`[net,tr] = train(net,...)`
	
	Paramter: *(default)*
	`net.trainParam.epochs` `1000` Maximum number of epochs to train
	`net.trainParam.goal` `0` Performance goal
	`net.trainParam.max_fail` `6` Maximum validation failures
	`net.trainParam.min_grad` `1e-7` Minimum performance gradient
	`net.trainParam.mu` `0.001` Initial `mu`
	`net.trainParam.mu_dec` `0.1` `mu` decrease factor
	`net.trainParam.mu_inc` `10` `mu` increase factor
	`net.trainParam.mu_max` `1e10` Maximum `mu`
	`net.trainParam.show` `25` Epochs between displays (`NaN` for no displays)
	`net.trainParam.showCommandLine` `false` Generate command-line output
	`net.trainParam.showWindow` `true` Show training GUI
	`net.trainParam.time` `inf` Maximum time to train in seconds

Training Functions *trainFcn*
trainru - *Unsupervised random order weight/bias training*
trainc - *Cyclical order weight/bias training*
trainr - *Random order incremental training with learning functions*
trains - *Sequential order incremental training with learning functions*
trainb - *Batch training with weight and bias learning rules*
trainbu - *Batch unsupervised weight/bias training*
trainscg - *Scaled conjugate gradient backpropagation*
traingdx - *Gradient descent with momentum and adaptive learning rate backpropagation*
traingdm - *Gradient descent with momentum backpropagation*
traingd - *Gradient descent backpropagation*
trainrp - *Resilient backpropagation*
trainlm - *Levenberg-Marquardt backpropagation*
trainbr - *Bayesian regularization backpropagation*
traincgp - *Conjugate gradient backpropagation with Polak-Ribiére updates*
traincgb - *Conjugate gradient backpropagation with Powell-Beale restarts*
trainbfg - *BFGS quasi-Newton backpropagation*
trainoss - *One-step secant backpropagation*
traincgf - *Conjugate gradient backpropagation with Fletcher-Reeves updates*
traingda - *Gradient descent with adaptive learning rate backpropagation*

Name	**cell2mat()** Convert cell array to ordinary array of the underlying data type
Description	The elements of the cell array must all contain the same data type, and the resulting array is of that data type
Syntax	`A = cell2mat(C)`

nntraintool – Plots:

Name	Performance (*plotperform*)
Description	`plotperform(TR)` plots error vs. epoch for the training, validation, and test performances of the training record `TR` returned by the function `train`
Note	Generally, the error reduces after more epochs of training, but might start to increase on the validation data set as the network starts overfitting the training data. In the default setup, the training stops after six consecutive increases in validation error, and the best performance is taken from the epoch with the lowest validation error.

Name	Training State (*plottrainstate*)
Description	`plottrainstate(tr)` plots the training state from a training record `tr` returned by `train`.
Note	

Name	Error Histogramm (*ploterrhist*)
Description	The blue bars represent training data, the green bars represent validation data, and the red bars represent testing data. The histogram can give you an indication of outliers, which are data points where the fit is significantly worse than the majority of data
Note	It is a good idea to check the outliers to determine if the data is bad, or if those data points are different than the rest of the data set. If the outliers are valid data points, but are unlike the rest of the data, then the network is extrapolating for these points. You should collect more data that looks like the outlier points, and retrain the network

Name	Regression (*plotregression*)
Description	to validate the network performance regression plots display the network outputs with respect to targets for training, validation, and test sets
Note	For a perfect fit, the data should fall along a 45 degree line, where the network outputs are equal to the targets.

Name	(*plotresponse*)
Description	`plotresponse(t,y)` takes a target time series `t` and an output time series `y`, and plots them on the same axis showing the errors between them.
Note	

Name	
Description	
Syntax	
Suggested	