MathWorks®

Recurrent Neural Network (RNN)

What Is a Recurrent Neural Network?

3 things you need to know

A recurrent neural network (RNN) is a network architecture for deep learning that predicts on time-series or sequential data.

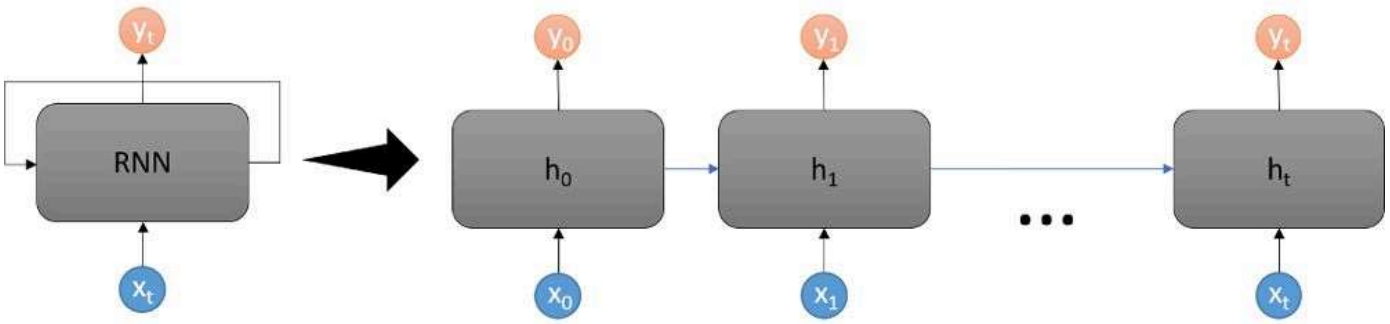
RNNs are particularly effective for working with sequential data that varies in length and solving problems such as natural signal classification, language processing, and video analysis.

How RNNs Work | Why RNNs Matter | RNNs with MATLAB

How RNNs Work

A recurrent neural network (RNN) is a deep learning structure that uses past information to improve the performance of the network on current and future inputs. What makes an RNN unique is that the network contains a hidden state and loops. The looping structure allows the network to store past information in the hidden state and operate on sequences.

How does the RNN know how to apply the past information to the current input? The network has two sets of weights: one for the hidden state vector and one for the inputs. During training, the network learns weights for both the inputs and the hidden state. When implemented, the output is based on the current input, as well as the hidden state, which is based on previous inputs.

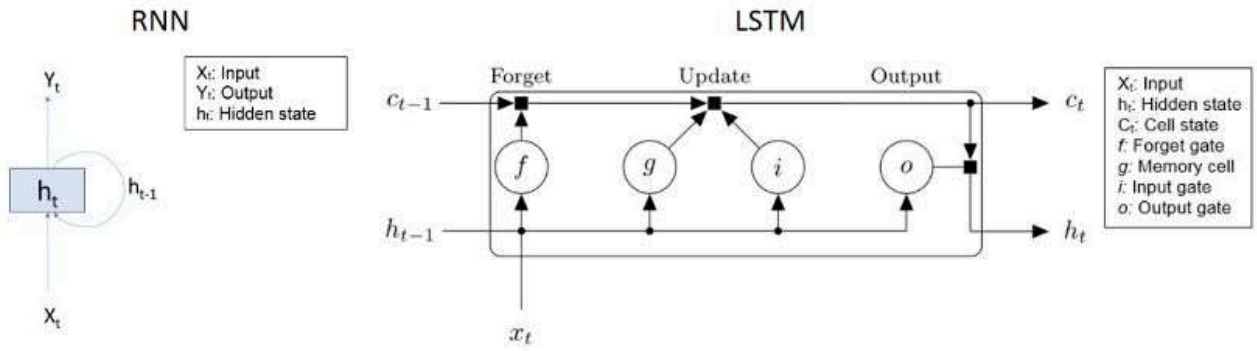


Unrolling a single cell of an RNN, showing how information moves through the network for a data sequence. Inputs are acted on by the hidden state of the cell to produce the output, and the hidden state is passed to the next time step.

LSTM

In practice, simple RNNs experience a problem with learning longer term dependencies. RNNs are commonly trained through backpropagation, where they can experience either a “vanishing” or “exploding” gradient problem. These problems cause the network weights to either become very small or very large, limiting the effectiveness of learning long-term relationships.

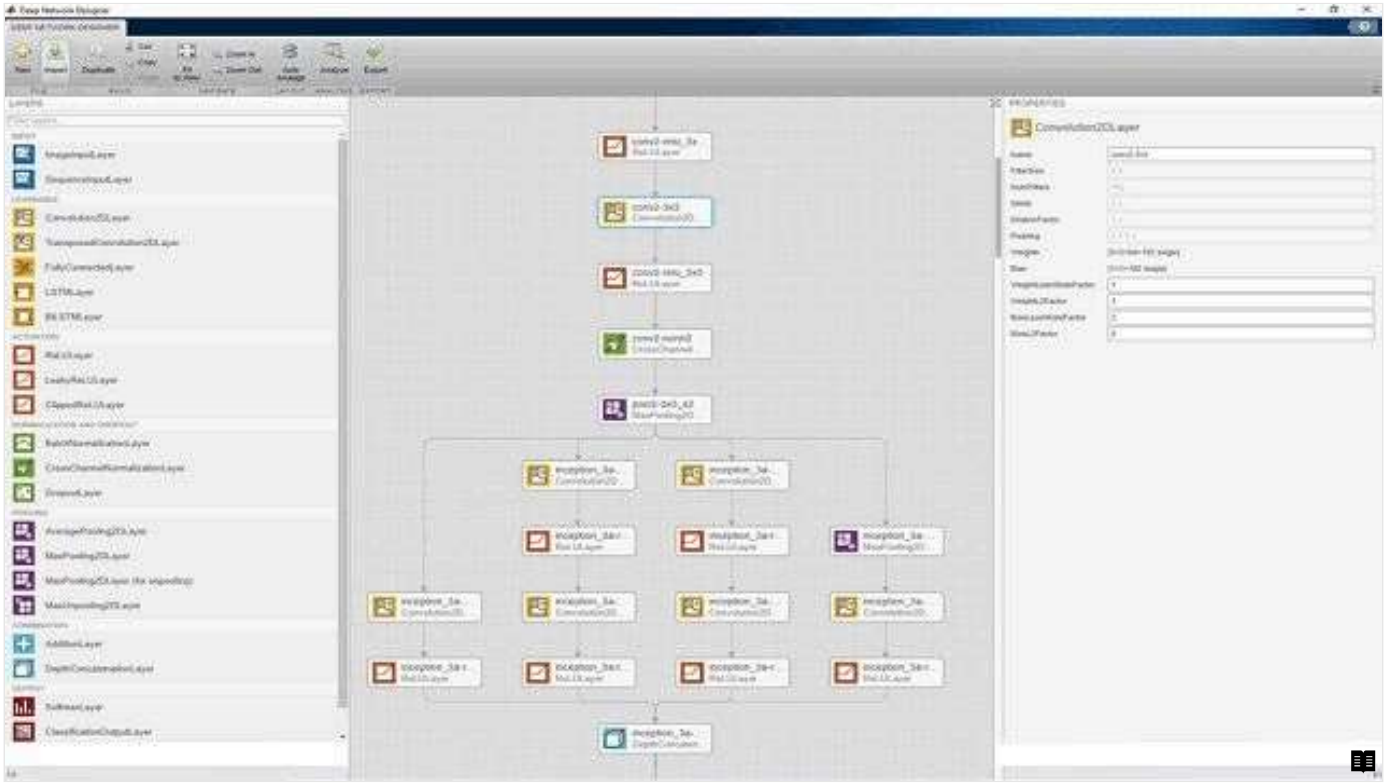
A special type of RNN that overcomes this issue is the long short-term memory (LSTM) network. LSTM networks use additional gates to control what information in the hidden state makes it to the output and the next hidden state. This allows the network to learn long-term relationships more effectively in the data. LSTMs are a commonly implemented type of RNN.



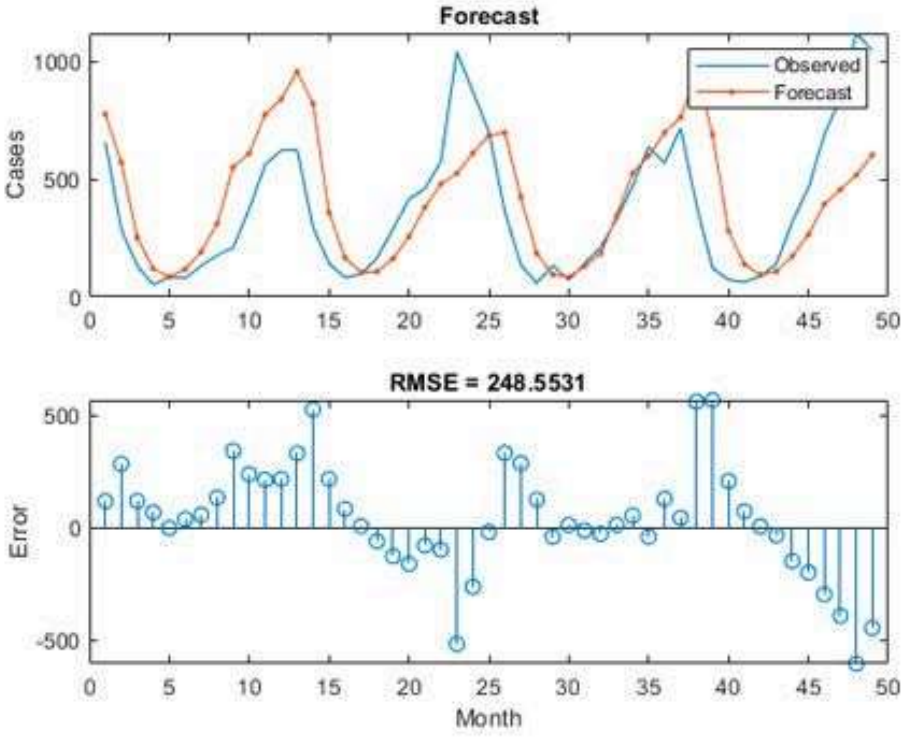
Comparison of an RNN (left) and LSTM network (right).

A bidirectional LSTM learns bidirectional dependencies between time steps of time-series or sequence data. These dependencies can be useful when you want the network to learn from the complete time series at each time step. Another RNN variant that learns longer term dependencies is the gated RNN. You can train and work with bidirectional LSTMs and gated RNNs in MATLAB®.

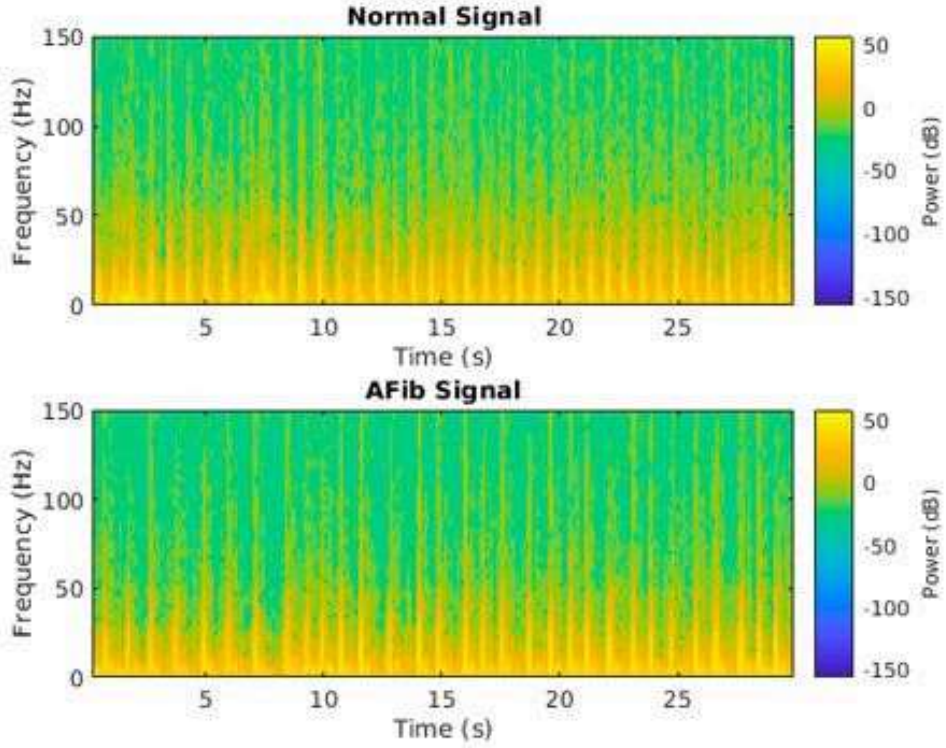
Get Started with RNN Examples in MATLAB



Create Simple Sequence Classification Network Using Deep Network Designer



Time Series Forecasting Using Deep Learning



Classify ECG Signals Using Long Short-Term Memory Networks

Why RNNs Matter

RNNs are a key technology in applications such as:

Signal Processing

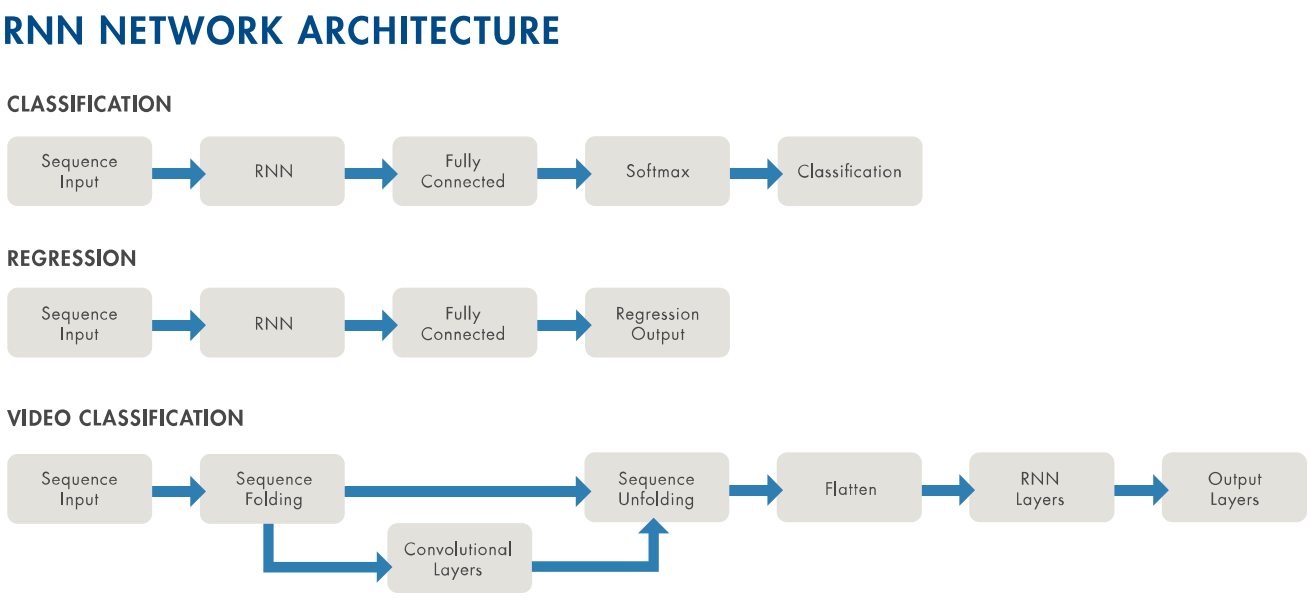
Signals are naturally sequential data, as they are often collected from sensors over time. Automatic classification and regression on large signal data sets allow prediction in real time. Raw signals data can be fed into deep networks or preprocessed to focus on specific features, such as frequency components. Feature extraction can greatly improve network performance.

Text Analytics

Language is naturally sequential, and pieces of text vary in length. RNNs are a great tool for natural language processing tasks, such as text classification, text generation, machine translation, and sentiment analysis (categorizing the meaning of words and phrases), because they can learn to contextualize words in a sentence.

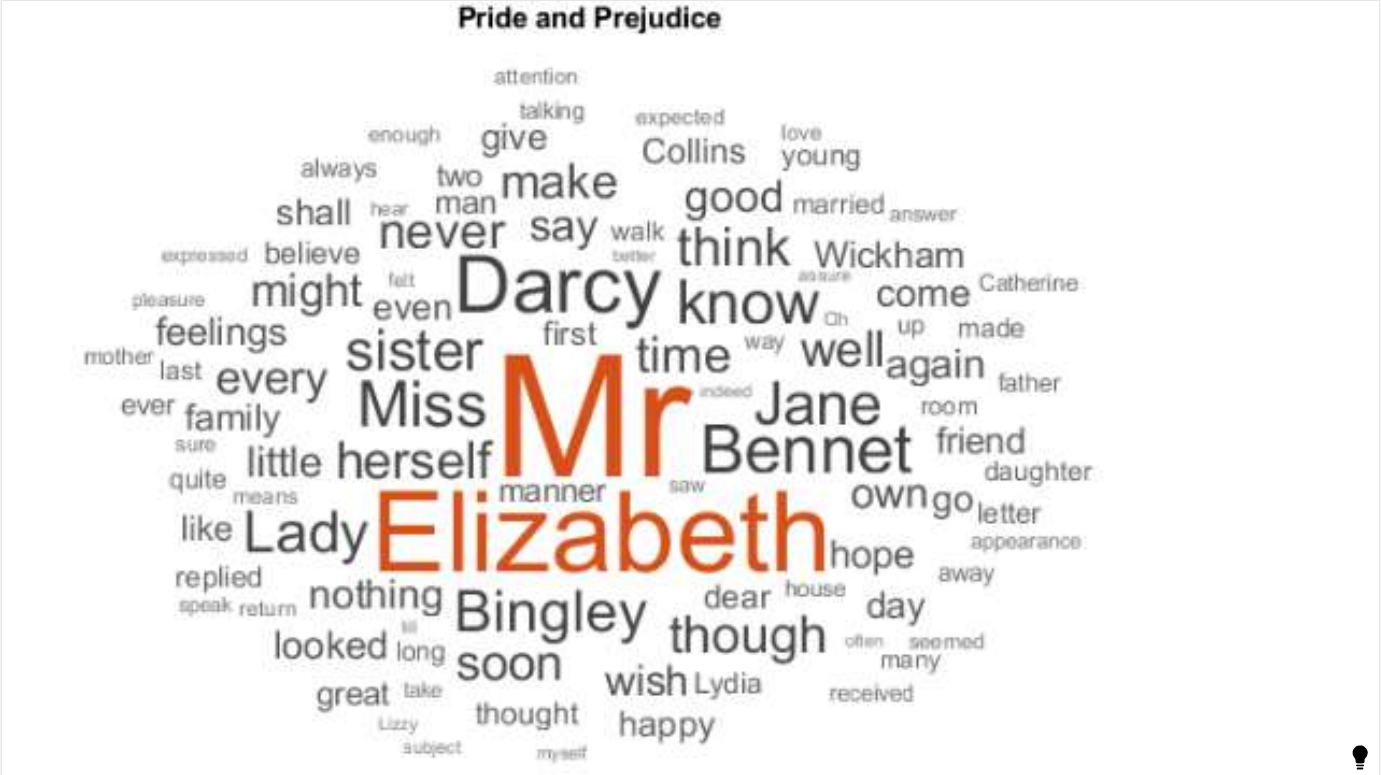
When Should You Use RNNs?

Consider using RNNs when you work with sequence and time-series data for classification and regression tasks. RNNs also work well on videos because videos are essentially a sequence of images. Similar to working with signals, it helps to do feature extraction before feeding the sequence into the RNN. Leverage CNNs (e.g., GoogLeNet) for feature extraction on each frame.

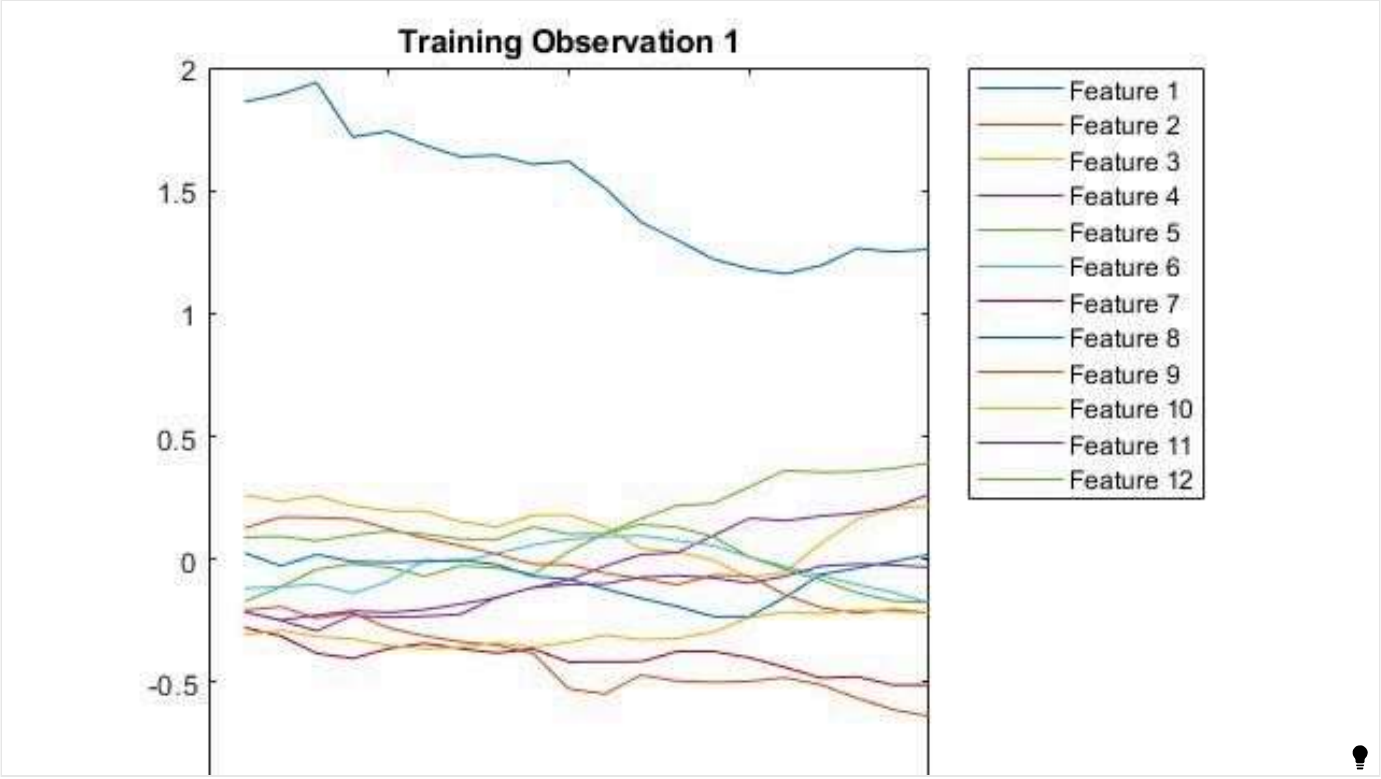


RNN network architecture for classification, regression, and video classification tasks.

Explore MATLAB examples using RNNs with text, signals, and videos.



Pride and Prejudice and MATLAB



Sequence-to-Sequence Classification Using Deep Learning

Classify Videos Using Deep Learning

RNNs with MATLAB

Using MATLAB with Deep Learning Toolbox™ enables you to design, train, and deploy RNNs. Using Text Analytics Toolbox™ or Signal Processing Toolbox™ allows you to apply RNNs to text or signal analysis.

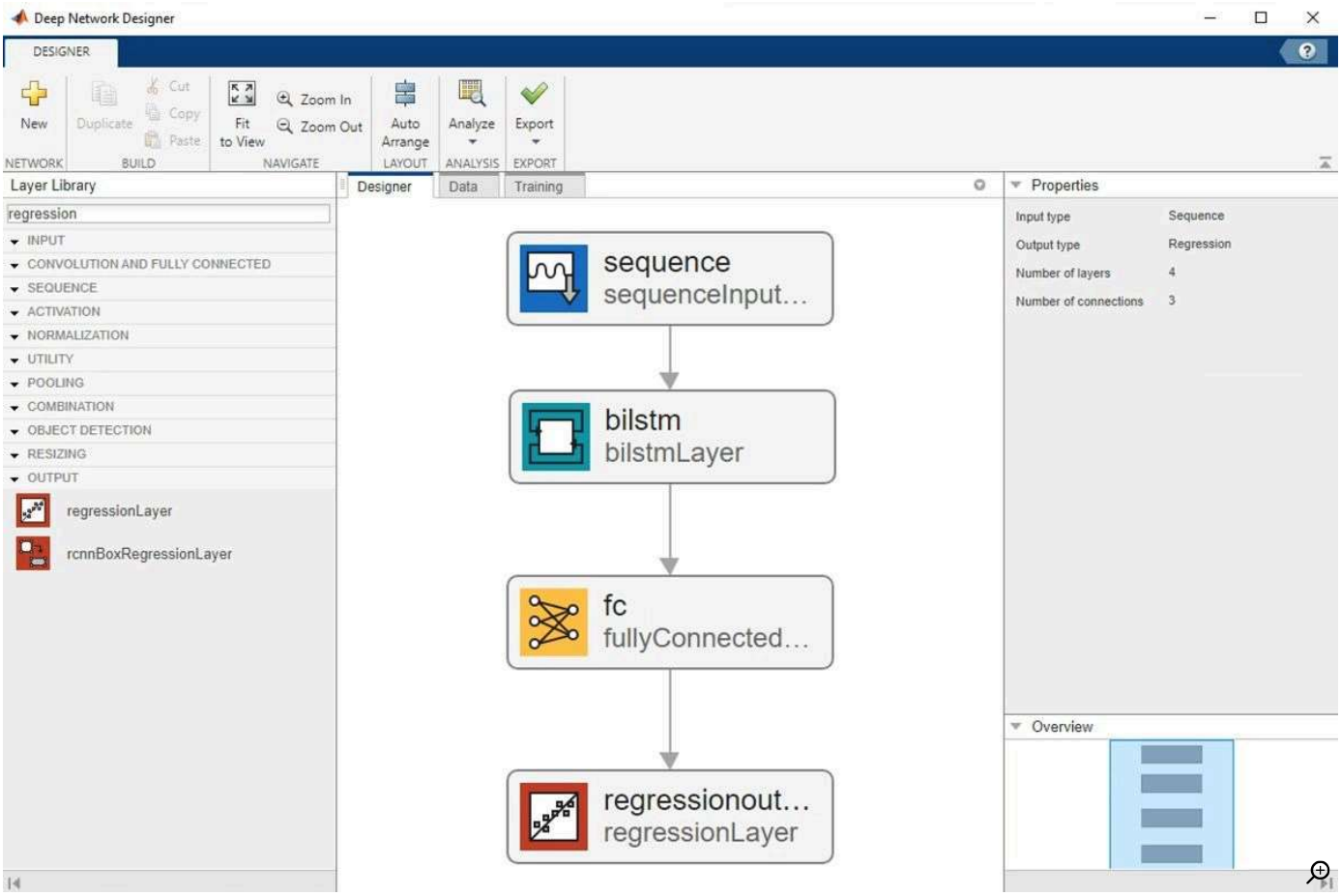
Design and Train Networks

You can create and train RNNs programmatically with a few lines of MATLAB code. Use recurrent layers (LSTM layer, bidirectional LSTM layer, gated recurrent layer, and LSTM projected layer) to build RNNs. Use a word embedding layer in an RNN network to map words into numeric sequences.

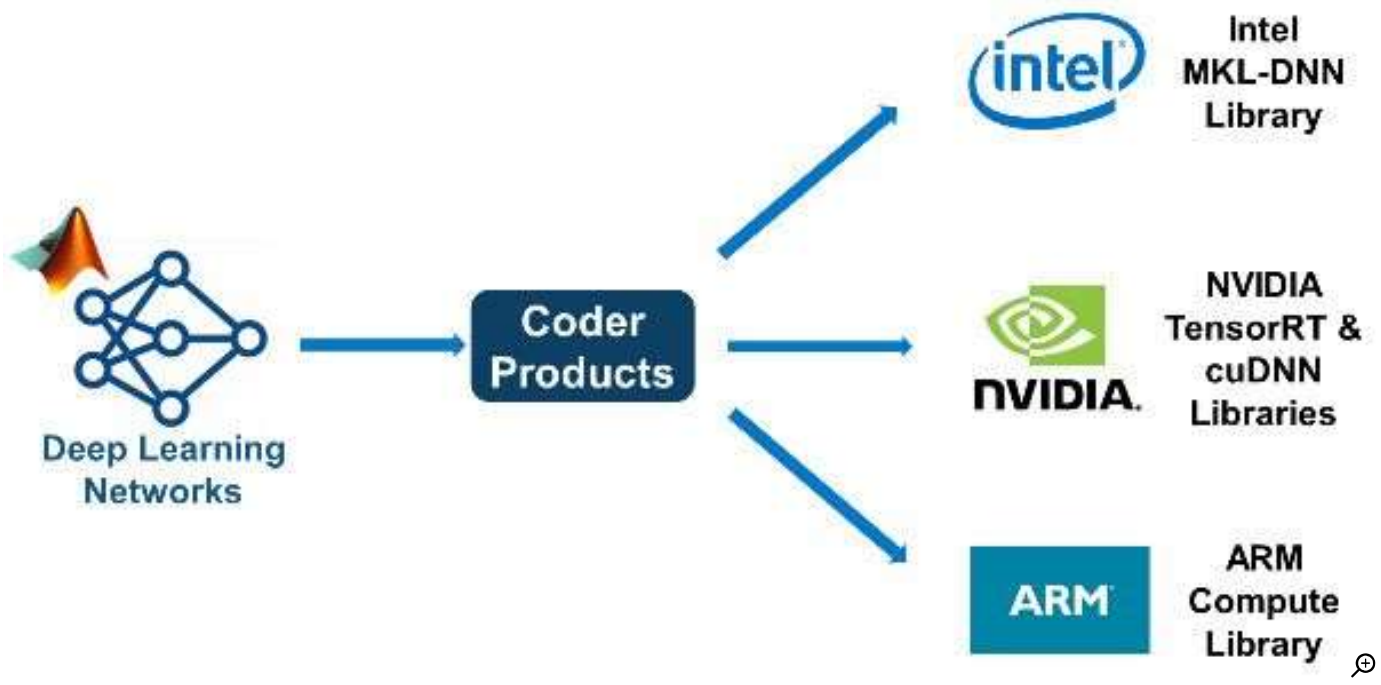
You can also create and train RNNs interactively using the Deep Network Designer app. Monitor training with plots of accuracy, loss, and validation metrics.

Deploy Networks

You can deploy your trained RNN on embedded systems, enterprise systems, FPGA devices, or the cloud. You can also generate code from Intel®, NVIDIA®, and ARM® libraries to create deployable RNNs with high-performance inference speed.



Deep Network Designer app for interactively building, visualizing, and editing RNN networks.

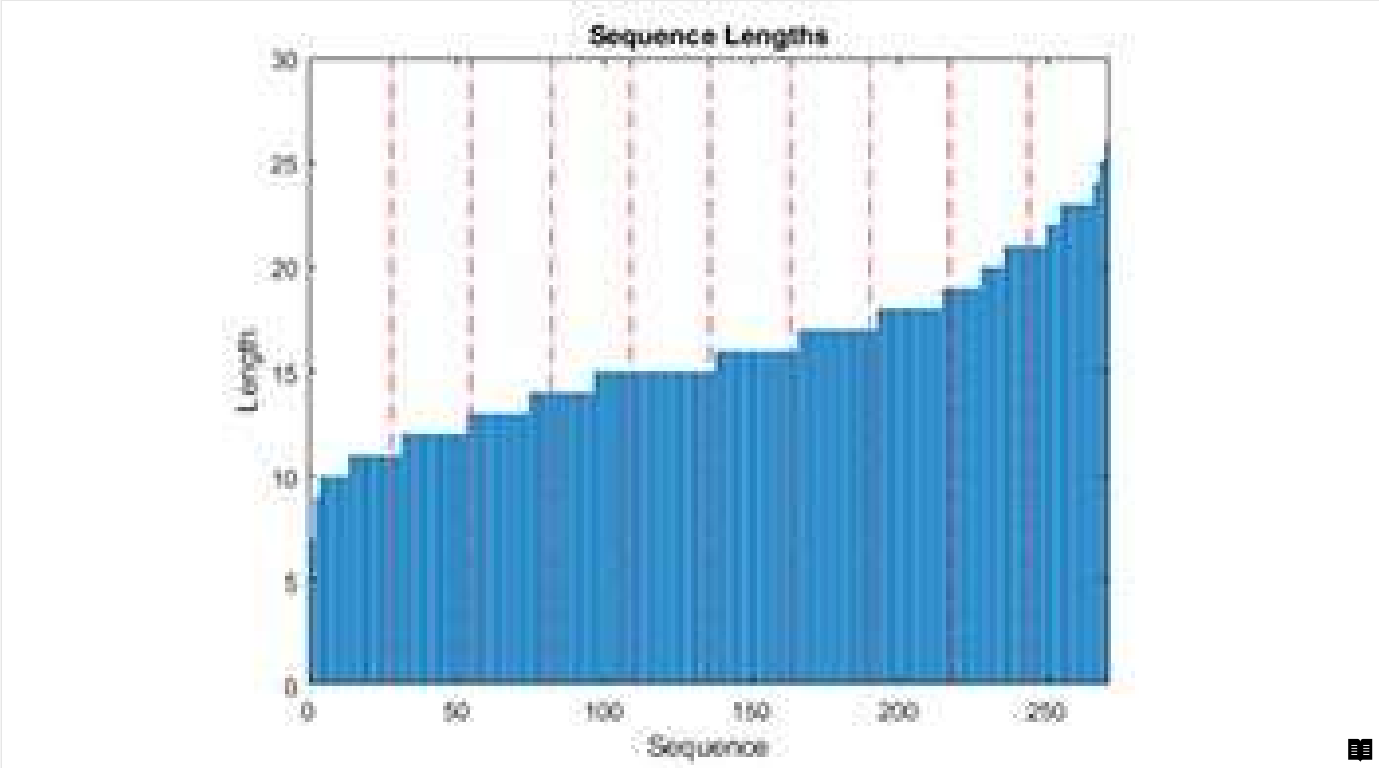


Quickly deploy trained deep learning networks into production.

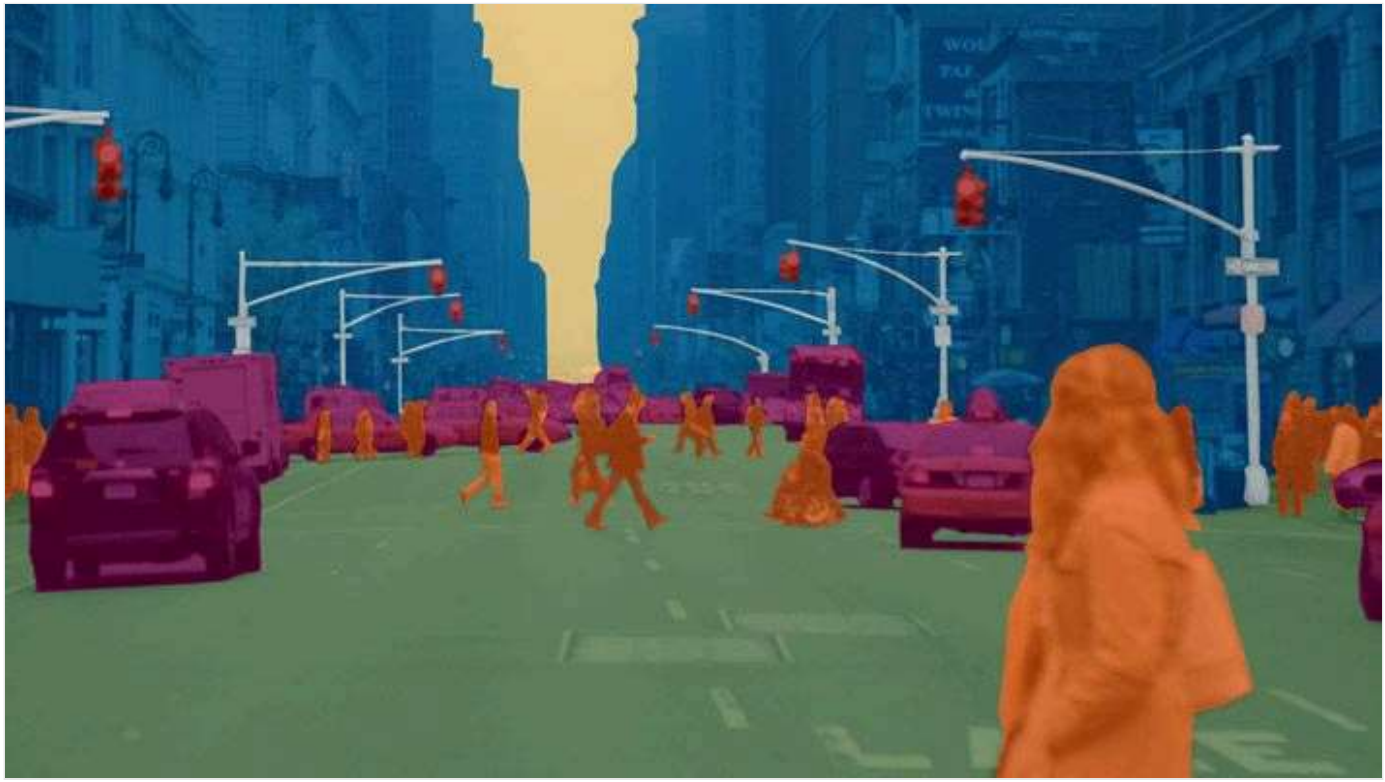
Keep Exploring RNNs

3 videos

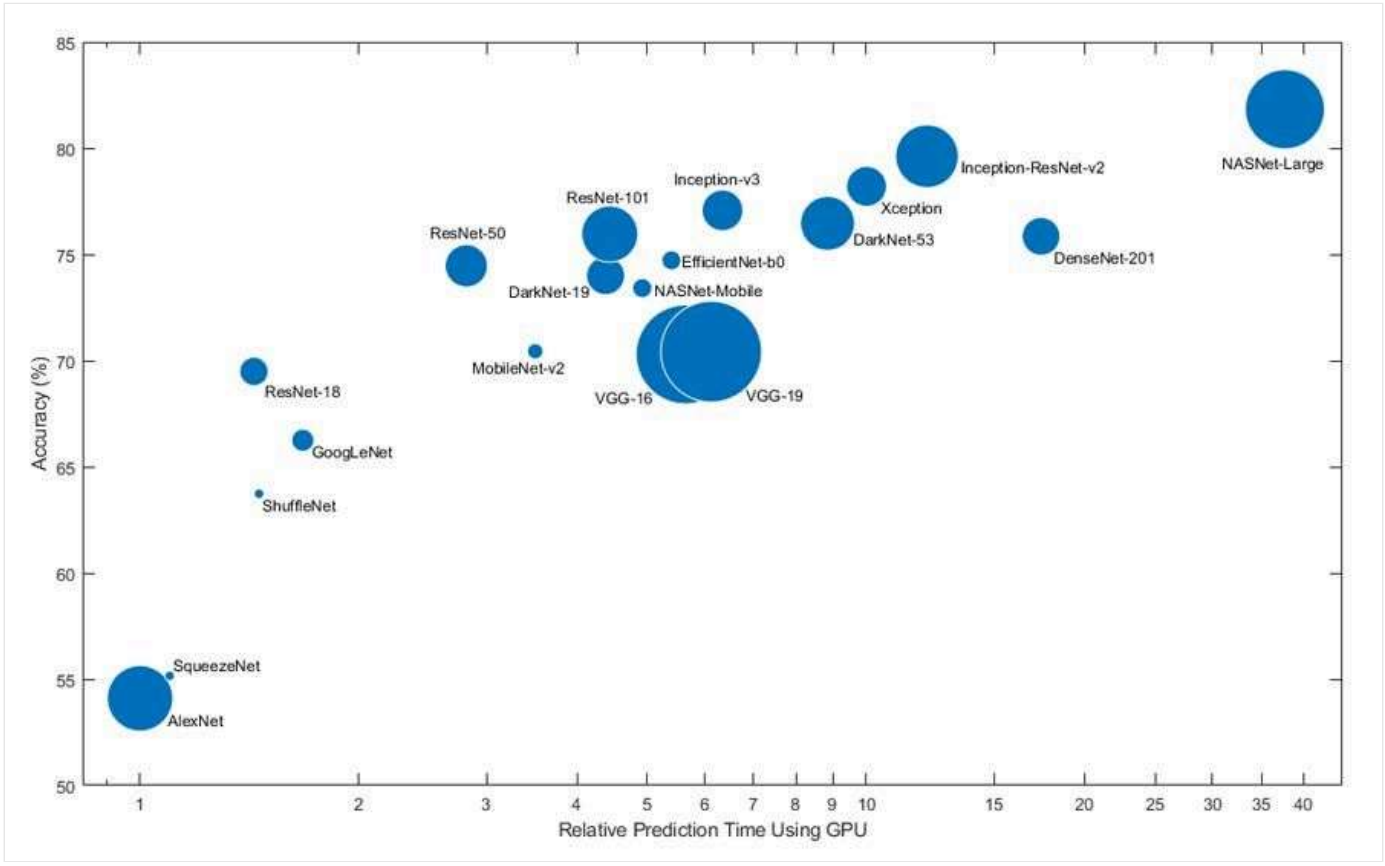
Introduction to Deep Learning (3 videos)



Long Short-Term Memory Neural Networks



Practical Deep Learning Examples with MATLAB



Pretrained Deep Learning Models

Related Topics

Artificial Intelligence

Deep Learning

Machine Learning

30-Day Free Trial

Download now

Questions?

Contact sales

Join the conversation